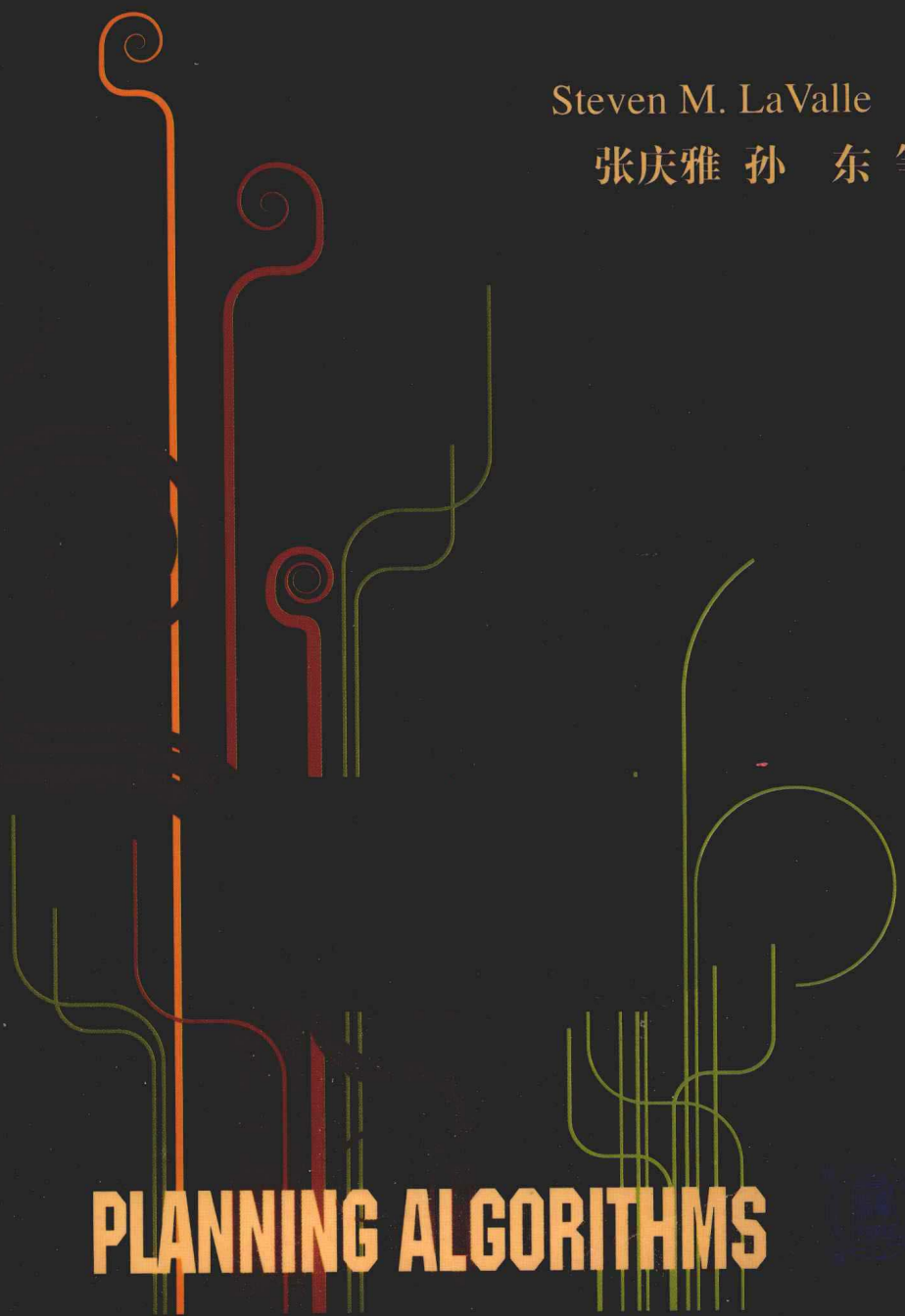
CAMBRIDGE

世界著名计算机教材精选

规划算法

Steven M. LaValle 著

张庆雅 孙 东 等译



PLANNING ALGORITHMS

清华大学出版社

规划算法 PLANNING ALGORITHMS

规划是人类智慧的结晶，规划问题广泛地存在于人们的日常工作和生活中。现在，规划已涉及计算机科学、人工智能、力学、机械学、控制论、对策论、概率论、图论、拓扑学、微分几何、代数几何等许多现代科学领域。本书是作者多年来教学和科研工作的总结，系统地介绍了规划领域中的基础知识和最新成果。作者将三个相对独立的学科：机器人学、人工智能和控制论巧妙地结合在一起。本书给出了大量内容详实的实例，使本来相对比较难以理解的数学问题变得生动起来，课后的阅读参考和练习题则能够进一步加深和扩展读者对相应内容的理解。

本书可以作为计算机类、控制类专业高年级本科学生或者研究生的教科书，也可以作为从事机器人学、控制科学、计算机科学研究的广大读者的参考书，其中的思想方法对社会科学工作者也大有益处。

世界著名计算机教材精选

- 数字图像处理：Java语言算法描述
- 逻辑设计基础（第3版）
- 计算机网络（第4版）
- TCP/IP协议族（第4版）
- 无线通信与网络（第2版）
- 密码学与网络安全（第2版）
- 网络安全基础：应用与标准（第4版）
- 编译器设计基础
- 数据结构基础（C语言版）（第2版）
- 数据结构基础（C++语言版）（第2版）
- 标准C程序设计（第4版）
- C++面向对象程序设计（第4版）
- Java面向对象程序设计（第2版）
- Java程序设计：一种跨学科的方法
- 计算理论基础（第2版）
- 数值分析与科学计算
- 数据挖掘教程
- Web数据挖掘
- 计算机图形学（OpenGL版）（第3版）
- 计算几何：算法与应用（第3版）
- Java软件结构与数据结构（第3版）
- TCP/IP协议原理与应用（第3版）
- 操作系统原理、设计与应用
- 计算群体智能基础
- 计算智能导论（第2版）
- 程序设计基础（第3版）
- MPI与Open MP并行程序设计：C语言版
- 离散数学与组合数学（第5版）
- 算法设计
- 算法基础
- 数据结构与算法分析（C++语言描述）（第2版）
- 数据结构与算法分析（Java语言描述）（第2版）
- 计算机组成和设计硬件／软件接口（第2版）
- 计算机组织与体系结构：性能设计（第7版）
- 操作系统原理
- 分布式系统原理与范型（第2版）
- 数据库管理系统原理与设计（第3版）
- 数据库系统基础教程
- 数据库设计与开发
- 面向对象系统分析与设计（第2版）
- 面向对象设计UML实践(第2版)
- 软件体系结构
- 面向对象软件工程：使用UML、模式与Java（第3版）
- 数字图像处理：原理与应用
- 程序设计语言概念（第9版）
- 规划算法

ISBN 978-7-302-23041-0
9 787302 230410 >
定价：99.00元

CAMBRIDGE
UNIVERSITY PRESS
www.cambridge.org

世界著名计算机教材精选

规 划 算 法

Steven M. LaValle 著
张庆雅 孙 东 等译

清 华 大 学 出 版 社
北 京

Planning Algorithms [0521862051] by **Steven M. LaValle** first published by Cambridge University Press **2006**. All rights reserved.

This **simplified Chinese** edition for the People's Republic of China is published by arrangement with the Press Syndicate of the University of Cambridge, Cambridge, United Kingdom.

© Cambridge University Press & Tsinghua University Press 2010.

The book is in copyright. No reproduction of any part may take place without the written permission of Cambridge University Press or Tsinghua University Press.

This edition is for sale in the mainland of China only, excluding Hong Kong SAR, Macao SAR and Taiwan, and may not be bought for export therefrom.

此版本仅限中华人民共和国境内销售,不包括香港、澳门特别行政区及中国台湾。不得出口。

北京市版权局著作权合同登记号　图字：01-2009-7995 号

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目（CIP）数据

规划算法/(英)拉瓦利(LaValle,S. M.)著;张庆雅,孙东等译. —北京：清华大学出版社,2011.1

(世界著名计算机教材精选)

书名原文：Planning Algorithms

ISBN 978-7-302-23041-0

Ⅰ. ①规…　Ⅱ. ①拉… ②张… ③孙…　Ⅲ. ①算法—教材　Ⅳ. ①O242.23

中国版本图书馆 CIP 数据核字(2010)第 113473 号

责任编辑：龙啟铭

责任校对：梁　毅

责任印制：杨　艳

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座

http://www.tup.com.cn　　**邮　　编**：100084

社　总　机：010-62770175　　**邮　购**：010-62786544

投稿与读者服务：010-62795954,jsjjc@tup.tsinghua.edu.cn

质　量　反　馈：010-62772015,zhiliang@tup.tsinghua.edu.cn

印 装 者：北京鑫海金澳胶印有限公司

经　　销：全国新华书店

开　　本：185×260　　**印　张**：44.25　　**字　数**：1099 千字

版　　次：2011 年 1 月第 1 版　　**印　次**：2011 年 1 月第 1 次印刷

印　　数：1～3000

定　　价：99.00 元

产品编号：032673-01

译 者 序

规划是人类智慧的结晶，规划问题广泛地出现在人们的日常工作和生活中。例如，奥运场馆建设、军事行动、交通运输、农作物耕种、航天飞机再入飞行、无人机航迹规划等不胜枚举，从孙子兵法到田忌赛马、从“运筹帷幄”到“事半功倍”，这些典故和词汇无不闪耀着规划的光辉。虽然规划问题是一个古老的问题，但是现代科学的发展为规划的发展注入了新的血液和强劲动力。现在，规划涉及计算机科学、人工智能、力学、机械学、控制论、对策论、概率论、图论、拓扑学、微分几何、代数几何等许多现代科学领域，所以说规划是一个既古老而又崭新的学科，它随着人类对客观世界的认识水平的发展而发展。

译者在从事航空航天飞行器的导航、制导与控制的科研工作中逐步接触并加深了对规划问题的认识和理解，也在工作中接触到《规划算法》这本书。《规划算法》是作者 Steven M. LaValle 教授多年来对教学和科研工作的总结，本书系统地介绍了规划领域中的基础知识和最新成果，把读者带入一个色彩斑斓的规划世界。作者将 3 个相对独立的学科——机器人学、人工智能和控制论——巧妙地结合在一起，使它们服务于规划这个主题。大量内容翔实的实例使本来难以理解的数学问题变得生动起来。课后的阅读参考和练习题则能够进一步加深和扩展读者对相应内容的理解。我们在运筹学等多个学科中已接触到规划的思想和算法，但是目前在国内很少有系统介绍规划问题的书籍，这使得译者产生将它翻译成中文介绍给我国读者的冲动和愿望。

本书涉及的许多数学知识超出了我国工科教育的数学教学内容，因此，阅读本书并深刻理解本书对读者的知识结构是一个挑战。迎接并战胜挑战，读者一定会受益匪浅。本书可以作为计算机类、控制类专业高年级本科学生或者研究生的教科书，也可以作为从事机器人学、控制科学、计算机科学研究人员的参考书，并且其中的一些思想方法对社会科学工作者也不无益处。

本书的翻译工作得到原作者 Steven M. LaValle 教授的大力支持和帮助，他的同事和他的研究生阅读了译稿并提出了许多意见和建议。在此对他们表示衷心的感谢。

在翻译过程中，译者本着“信”、“达”、“雅”的原则对原著进行了仔细的研究和推敲，但是由于译者水平有限，书中难免存在各种纰漏，敬请广大读者指正。

张庆雅　孙　东

前　言

“规划算法”的含义

由于对机器人、人工智能和控制等方面的研究取得了巨大的发展，三个曾经十分迥异的学科在当前情况下却处在一个具有某种相互冲突的课程中。在机器人领域，运动规划最初关心的是诸如如何将一架钢琴从一个房间移动到另外一个房间而不与任何其他物品发生碰撞等类似问题。但是该领域已经发展到包括不确定性、多体、动力学等问题。在人工智能方面，规划最初意味着搜索一个逻辑操作序列或者行动[1]（动作）序列，它们将一个初始状态转移到所需要的目标状态。目前规划扩展到包括一些决策论的思路，如马尔可夫决策过程、不完整状态信息和对策论平衡。尽管控制论通常关心的是诸如稳定性、反馈、最优性等问题，但是人们越来越对设计寻找非线性系统可行开环轨迹的算法产生兴趣。在这方面的工作中，大都用到了“运动规划”这个术语，但是它与机器人领域中的“运动规则”有所不同。虽然它们最初所考虑的问题不同，但是机器人学、人工智能以及控制论都已经对它们自身的研究范围进行了拓展以分享它们所共同感兴趣的领域。

本书从围绕该公共领域这个更加广泛的意义上使用“规划”这个术语。但是这并不意味着该术语能够覆盖机器人、人工智能和控制论中所有重要内容。论述着眼于与规划有关的算法问题。机器人技术的注意力集中在通过处理复杂几何模型来设计生成有用运动的算法。人工智能的注意力则集中在设计使用决策论模型来计算相应行动的系统。控制论则着眼于计算系统可行轨迹的算法，并考虑到反馈和最优性。在绝大多数控制论文献中用来进行阐述说明的分析技术并不是本书的重点内容。

“规划与控制”通常用来表示系统研究中密切配合、相互补充的一些问题。规划通常被认为是比控制更高一个层次的过程，但本书并不这么认为。忽略“规划”和“控制”的原始内涵，它们就可以相互交换使用。它们都涉及本书中的某种类型的决策，而与“高”或“低”层的概念无关。可以研究层次方法，每层都可以被称为“规划”或者“控制”，它们之间并没有什么不同。

本书面向的读者

本书最初的读者是计算机科学和工程学科的高年级本科生或者低年级硕士研究生。也可以作为机器人学、人工智能和控制论研究人员的入门书籍。对诸如计算生物学（药物设计和蛋白质折叠）、虚拟样机、制造、视觉游戏设计、计算机图形学等其他领域的研究人员来说，我也期望本书中的论述能够引起他们的兴趣，并且本书也是为通过设计和实现规划

1　译者注：在人工智能中通常称之为“动作”。但是由于规划问题并不局限于人工智能领域，在诸如抢险救灾、军事部署等涉及大量人员物质调度的规划问题中，“动作”这个概念似乎过于具体，涵盖的范围也略显狭小，所以在本书中将 action 翻译为“行动”。

方法来解决他们所关心的问题的研究人员而著的。

作者试图尽可能使本书具有完整性和可读性。书中介绍和解释了一些先进的数学概念（通常超出计算机科学与工程专业本科生已经学过的概念）。对于对数学具有浓厚兴趣的读者来说，本书给出了更多的研究和学习的方向。

本书所适合的领域

本书适合如下这些众所周知的学科领域。

机器人学：本书讨论了机器人规划问题，它包括运动规划、轨迹规划和存在不确定性下的规划。这仅仅是机器人技术中的一部分，许多机器人技术在本书中并没有被直接介绍，如机械设计、动力系统建模、反馈控制、传感器设计、计算机视觉、逆运动学、类人型机器人等。

人工智能：机器学习是目前人工智能的一个最大而且是最成功的分支。本书（可能与文献[384]一起）是对机器学习的重要补充，它可以被认为是“机器规划”。诸如强化学习、决策论等这样的学科处在两者的边缘上，本书也将予以介绍。一旦成功实现学习，那么应该产生什么决策？这就进入到规划领域。

控制理论：历史上控制理论已经将这里所考虑的内容当作有微分约束情况下连续空间中的规划问题来讨论。动态性、最优性和反馈在控制理论中是最重要的。对注意力主要集中在开环控制律，可行性（而不是最优性）和动态性可能很重要，也可能不重要，本书对此进行了补充。尽管如此，反馈、最优性和动态性的概念在本书中的许多位置都出现了。但本书中的内容更偏重于算法，这与在控制理论中发展起来的分析技术不同。

计算机图形学：动画是近年来计算机图形学中的一个热点。本书的许多技术或者已经应用，或者可以应用于制作视频游戏人物造型、虚拟人或者机械系统。规划算法允许用户在一个更高层次上确定任务，这就避免了原来不得不进行的插入关键科的烦琐工作。

算法：就像书名所说，本书可能适合算法，算法属于计算机科学。贯串本书，出现了来自于组合学的许多典型问题和复杂问题。在许多地方，作为算法的分支，计算几何和计算实代数几何中的技术变得非常重要。另一方面本书又不是纯粹的关于算法的书籍，因为其许多内容关心的是描述应用中出现的各种决策过程。本书也不只关注于复杂性和组合学。

其他领域：规划算法还触及周围的许多其他领域。例如运动规划算法构成了本书的主要部分，它对计算生物学、制造业中的虚拟样机、建筑设计、航空工程和计算地理学都有巨大的影响。

建议的使用方法

本书的思路是一个章节一个章节很自然地向前论述，但同时本书的安排也使得读者可以跳过部分章节而直接阅读其他章节。4 个部分之间的相互依赖性如图 1 所示。

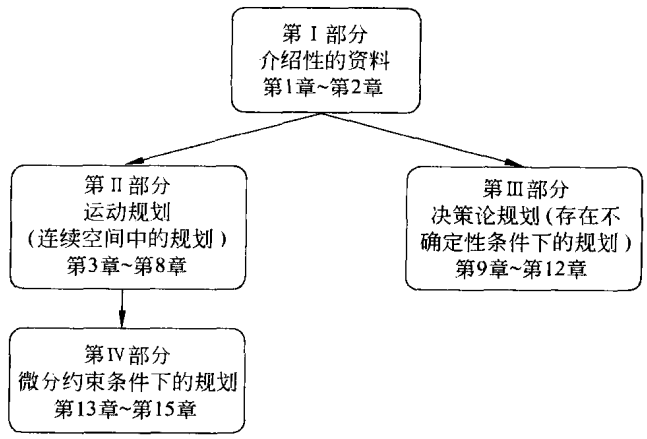


图 1　本书的 4 个主要部分之间的关系

如果读者仅对机器人运动规划感兴趣，那么需要阅读第 3 章～第 8 章，可能还需要阅读第 2 章中的某些离散规划算法，因为它们会在运动规划中出现。第 3 章和第 4 章将介绍基本机器人运动规划所需要的基础知识。第 5 章和第 6 章则提出解决问题的算法。第 7 章和第 8 章对基本问题进行了扩展。如果读者还对非完整性规划和其他涉及微分约束的问题感兴趣，那么阅读完第 II 部分以后就可以直接阅读第 13 章～第 15 章。

第 11 章和第 12 章将介绍存在感测不确定性的问题。这些问题建立在第 11 章介绍的信息空间上。第 12 章将介绍在信息空间中进行规划的算法。

如果读者主要对决策论规划感兴趣，那么阅读第 2 章以后可直接阅读第 9 章～第 12 章。因为第 9 章～第 12 章的内容不依赖于第 3 章～第 8 章所介绍的运动规划。因此如果读者对运动规划不感兴趣，那么可以跳过这些章节。

本书的内容可以作为一个学期的课程或者一个季度的课程。图 2 能够帮助你在核心内容和可选内容之间进行选择。对于一个高年级本科的课程来说，作者建议覆盖一个核心内容和某些可选内容。对于研究生课程来说，覆盖两个核心内容和某些可选内容，这要根据学生的具体情况来确定。如果包括所有 3 个核心内容和某些可选内容就可作为一个两学期课程，并且可以用许多不同方法来设立两个独立的课程。如果要避开连续空间，那么第 2.1

运动规划

核心内容：2.1—2.2，3.1—3.3，4.1—4.3，5.1—5.6，6.1—6.3

可选内容：3.4—3.5，4.4，6.4—6.5，7.1—7.7，8.1—8.5

存在不确定性条件下的规划

核心内容：2.1—2.3，9.1—9.2，10.1—10.4，11.1—11.6，12.1—12.3

可选内容：9.3—9.5，10.5—10.6，11.7，12.4—12.5

微分约束条件下的规划

核心内容：8.3，13.1—13.3，14.1—14.4，15.1，15.3—15.4

可选内容：13.4—13.5，14.5—14.7，15.2，15.5

图 2　在第 II 部分、第III部分和第IV部分，都有核心内容和可选内容

节～第 2.5 节、第 9.1 节～第 9.5 节、第 10.1 节～第 10.5 节、第 11.1 节～第 11.3 节、第 11.7 节和第 12.1 节～第 12.3 节可以构成一个关于离散规划的课程。如果有兴趣教授某些对策论内容，那么相关内容的大致章节有第 9.3 节～第 9.4 节、第 10.5 节、第 11.7 节和第 13.5 节。关于对将来研究的展望的一些材料将出现在第 7 章、第 8 章、第 11 章、第 12 章和第 14 章。关于信息空间的研究尚处在初始阶段。

为了有利于教学，本书提供了 500 多个的例子和练习题。每章中的习题分为书面解答问题和实现方案两种。对于运动规划，学生通常陷入底层实现中。一种解决的方法是用运动策略库（MSL）：http://msl.cs.uiuc.edu/msl/，以此为基础来进行研究。作者已经成功将此应用于本科和研究生教学中。

对于附加材料、更新和勘误表等内容，见与本书有关的网页 http://planning.cs.uiuc.edu/。读者可以免费下载本书的电子版以供个人使用。

更多阅读资料，可以参见每章结尾和书后所给出的参考文献。多数文献都可以在 Internet 上找到，但是这里未给出它们的地址，因为这些地址可能会发生变化。非常遗憾，Internet 上所列的文献比作者预计的少；因此在某些地方仅给出文献的列表，而且通常还不完整。作者尽可能平等对待本书所列文献，但是在某些地方的确存在对某些文献的偏爱。这是在所难免的，因为作者的研究工作和本书的写作有很大关系。

致谢

非常感谢我的学生和同事在本书撰写过程中所给予我的反馈和建议。这是多年来在斯坦福大学、爱荷华州立大学、伊利诺斯大学教学和研究的成果。这些大学对作者的工作给予了大量支持和帮助。

本书的许多内容是共同研究的成果。由于这个原因，所以要特别感谢合作者 Michael Branicky、Francesco Bullo、 Jeff Erickson、Emilio Frazzoli、Rob Ghrist、Leo Guibas、Seth Hutchinson、Lydia Kavraki、James Kuffner、Jean-Claude Latombe、Rajeev Motwani、Rafael Murrieta、Rajeev Sharma、Thierry Siméon 和 Giora Slutzki 的敏锐的洞察力以及与他们所进行的有益的探讨。在多年的相互交流中，他们的想法帮助作者逐渐形成本书的观点和思路。

许多很有价值的认识和观察来源于与学生之间的合作，特别是 Peng Cheng、Hamid Chitsaz、Prashanth Konkimalla、Steve Lindemann、Jason O'Kane、Stjepan Rajko、Shai Sachs、Boris Simov、Benjamin、Tovar、Jeff Yakey、Libo Yang 和 Anna Yershova。感谢与他们一起工作的机会，与他们之间的相互交流加深了作者对规划的认识和理解。

在撰写本书过程中的许多时候，作者会回忆起与同事之间进行的技术探讨对作者所产生的巨大影响。毋庸置疑，如下的名单是不完全的，尽管如此，作者仍愿意感谢他们所给予的有益的认识和具有激发性的探讨：Pankaj Agarwal、Srinivas Akella、Nancy Amato、Devin Balkcom、Tamer Basar、Antonio Bicchi、Robert Bohlin、Joel Burdick、Stefano Carpin、Howie Choset、Juan Cortés、Jerry Dejong、Bruce Donald、Ignacy Duleba、Mike Erdmann、 Roland Geraerts、Malik Ghallab、Ken Goldberg、Pekka Isto、Vijay Kumar、Andrew Ladd、Jean-Paul Laumond、Kevin Lynch、Matt Mason、Pascal Morin、David Mount、Dana Nau、Mark Overmars、Jean Ponce、Elon Rimon 和 Al Rizzi。

感谢 Karl Bohringer、Marco Bressan、Stefano Carpin、John Cassel、Peng Cheng、Hamid Chitsaz、 Ignacy Duleba、Claudia Esteves、Eleonora Fantini、Brian Gerkey、Ken Goldberg、Björn Hein、Sanjit Jhala、Marcelo Kallmann、James Kuffner、Olivier Lefebvre、Mong Leng、Steve Lindemann、Dennis Nieuwenhuisen、Jason O'Kane、Neil Petroff、Mihail Pivtoraiko、Stephane Redon、Gildardo Sanchez、Wiktor Schmidt、Fabian Schöfeld、Robin Schubert、Sanketh Shetty、Mohan Sirchabesan、James Solberg、Domenico Spensieri、Kristian Spoerer、Tony Stentz、Morten Strandberg、Ichiro Suzuki、Benjamin Tovar、Zbynek Winkler、Anna Yershova、Jing jin Yu、George Zaimes 和 Liang jun Zhang，他们对在线原稿中的许多错误进行了指正。还要感谢学习该课程的研究生，他们记录了以前某些部分的草稿。他们包括爱荷华州立大学和伊利诺斯大学的学生：Peng Cheng、Brian George、Shamsi Tamara Iqbal、Xiaolei Li、Steve Lindemann、Shai Sachs、Warren Shen、Rishi Talreja、Sherwin Tam 和 Benjamin Tovar。

感谢 Krzysztof Kozlowski 和他的课题组成员 Joanna Gawecka、Wirginia Król 和 Marek Lawniczak，以及作者在 Politechnika Poznańska（波兹南技术学院）所得到的巨大帮助。还要感谢作者在欧洲时，Heather Hall 替作者管理在美国的事务。感谢国家科学基金、海军研究办公室和 DARPA 对作者的资助，他们对作者撰写本书提供了支持。伊利诺斯大学计算机科学系也给作者提供了巨大的支持。

最后，感谢本书的责任编辑 Lauren Cowles、文字编辑 Elise Oranges 和剑桥大学出版社的其他人员，感谢他们在准备出版本书中所付出的努力和提出的建议。

Steven LaValle
伊利诺斯州，美国

目　录

第Ⅰ部分　介绍性的资料

第 1 章　绪论 2
1.1　从规划（的过程）到规划（的结果） 2
1.2　实例与应用 3
1.3　规划的基本组成 11
1.4　算法、规划器与规划 12
1.4.1　算法 12
1.4.2　规划器 13
1.4.3　规划 13
1.5　本书的组织安排 15
第 2 章　离散规划 18
2.1　离散可行规划简介 18
2.1.1　问题表述 18
2.1.2　离散规划的实例 19
2.2　可行规划的搜索 21
2.2.1　一般前向搜索 21
2.2.2　特殊前向搜索 23
2.2.3　其他搜索方案 26
2.2.4　搜索方法的统一描述 28
2.3　离散最优规划 29
2.3.1　最优定长规划 30
2.3.2　不指定长度的最优规划 34
2.3.3　再论 Dijkstra 算法 37
2.4　用逻辑来表示离散规划 38
2.4.1　类似 STRIPS 的表示 39
2.4.2　转换到状态空间表示 41
2.5　基于逻辑的规划方法 42
2.5.1　部分规划空间中的搜索 43
2.5.2　建立规划图 44
2.5.3　满足性规划 46
进一步阅读 48
习题 49

实现 50

第Ⅱ部分 运动规划

第 3 章 几何表示与变换 55
3.1 几何建模 55
3.1.1 多边形与多面体模型 56
3.1.2 半代数模型 59
3.1.3 其他模型 60
3.2 刚体变换 62
3.2.1 一般概念 62
3.2.2 二维变换 64
3.2.3 三维变换 65
3.3 物体运动链的变换 68
3.3.1 二维运动链 68
3.3.2 三维运动链 70
3.4 运动树的变换 77
3.5 非刚体的变换 82
进一步阅读 83
习题 83
实现 85
第 4 章 位形空间 86
4.1 拓扑的基本概念 86
4.1.1 拓扑空间 86
4.1.2 流形 90
4.1.3 路径与连通 94
4.2 位形空间 97
4.2.1 二维刚体：$SE(2)$ 97
4.2.2 三维刚体：$SE(3)$ 100
4.2.3 物体的链与树 103
4.3 位形空间障碍物 104
4.3.1 基本运动规划问题 104
4.3.2 显式建模 C_{obs}：平移情况 106
4.3.3 显式建模 C_{obs}：一般情形 110
4.4 闭运动链 112
4.4.1 数学概念 113
4.4.2 $\mathbb{R}^2$ 上的运动链 115
4.4.3 定义一般连杆组的簇 118
进一步阅读 121

习题 121
实现 124
第 5 章　基于采样的运动规划 125
5.1　*C*-空间中的距离与体积 126
5.1.1　度量空间 126
5.1.2　运动规划中重要的度量空间 127
5.1.3　基本测度理论的定义 129
5.1.4　使用正确的测度 130
5.2　采样理论 131
5.2.1　目的与基本概念 131
5.2.2　随机采样 133
5.2.3　低离散度采样 135
5.2.4　低差异采样 139
5.3　碰撞检测 141
5.3.1　基本概念 141
5.3.2　层次法 142
5.3.3　增量法 143
5.3.4　检验路径片段 144
5.4　递增采样与搜索 147
5.4.1　一般架构 147
5.4.2　改进离散搜索算法 149
5.4.3　随机势场 152
5.4.4　其他方法 153
5.5　快速探索稠密树 154
5.5.1　探索算法 155
5.5.2　有效地发现最近点 157
5.5.3　在规划中使用树 159
5.6　多疑问问题的路线图方法 160
5.6.1　基本方法 160
5.6.2　可视路线图 163
5.6.3　改进路线图的启发式方法 164
进一步阅读 166
习题 167
实现 168
第 6 章　组合运动规划 169
6.1　简介 169
6.2　多边形障碍区域 170
6.2.1　问题的表示 170
6.2.2　垂直胞腔剖分 172

6.2.3 最大间隙路线图 175
6.2.4 最短路径路线图 176
6.3 胞腔剖分 179
6.3.1 一般定义 179
6.3.2 二维剖分 181
6.3.3 三维垂直剖分 183
6.3.4 线段机器人的剖分 184
6.4 计算代数几何 190
6.4.1 基本定义与概念 190
6.4.2 圆柱代数剖分 193
6.4.3 Canny 的路线图算法 198
6.5 运动规划的复杂性 201
6.5.1 下界 202
6.5.2 Davenport-Schinzel 序列 204
6.5.3 上界 205
进一步阅读 207
习题 208
实现 209
第 7 章 基本运动规划的扩展 210
7.1 时变问题 210
7.1.1 问题的表述 210
7.1.2 直接解 212
7.1.3 速度调节方法 214
7.2 多机器人 215
7.2.1 问题的表述 215
7.2.2 解耦规划 218
7.3 离散与连续空间的混合空间 221
7.3.1 混合系统的架构 221
7.3.2 操作规划 225
7.4 闭运动链的规划 228
7.4.1 运动规划算法的适应性修改 229
7.4.2 主动-被动连杆的分解 232
7.5 机器人学和生物学中的折叠问题 235
7.6 覆盖规划 239
7.7 最优运动规划 241
7.7.1 一个机器人的最优性 241
7.7.2 多机器人的最优性 245
进一步阅读 246
习题 247

实现 248
第 8 章　反馈运动规划 250
8.1　目的 250
8.2　离散状态空间 251
8.2.1　反馈规划 251
8.2.2　将反馈规划作为导航函数 253
8.2.3　运动规划的基于栅格的导航函数 255
8.3　向量场与积分曲线 257
8.3.1　$\mathbb{R}^n$ 上的向量场 258
8.3.2　平滑流形 263
8.4　连续空间中的完备方法 269
8.4.1　反馈运动规划 269
8.4.2　胞腔复形上的向量场 272
8.4.3　最优导航函数 273
8.4.4　考虑动力学的向量场 275
8.5　连续空间中基于采样的方法 279
8.5.1　计算一个漏斗组合 279
8.5.2　带插值的动态规划 283
进一步阅读 290
习题 291
实现 292

第 III 部分　决策论规划

第 9 章　基本决策理论 295
9.1　基本概念 295
9.1.1　最优化 295
9.1.2　回顾概率论 298
9.1.3　随机策略 300
9.2　与大自然之间的对策 301
9.2.1　对大自然的建模 301
9.2.2　非确定模型和概率模型 302
9.2.3　利用观测 305
9.2.4　最优决策的例子 307
9.3　两人零和对策 310
9.3.1　对策表述 310
9.3.2　确定的策略 311
9.3.3　随机策略 314
9.4　非零和对策 317

9.4.1 两人对策 317
9.4.2 多人对策 322
9.5 对决策理论的进一步思考 323
9.5.1 效用理论与合理性 324
9.5.2 关于概率模型的问题 328
9.5.3 关于非确定模型的问题 330
9.5.4 关于对策论的有关问题 332
进一步阅读 332
习题 333
实现 335
第 10 章 序贯决策理论 336
10.1 与大自然之间的序贯对策 336
10.1.1 模型定义 336
10.1.2 前向投影与后向投影 340
10.1.3 一个规划及其执行 343
10.2 计算反馈规划的算法 345
10.2.1 值迭代 345
10.2.2 策略迭代 349
10.2.3 图搜索方法 351
10.3 无限范围问题 354
10.3.1 问题表述 354
10.3.2 求解技术 355
10.4 强化学习 358
10.4.1 一般原理 358
10.4.2 通过模拟来评估规划 359
10.4.3 Q-学习：计算最优规划 362
10.5 序贯对策论 363
10.5.1 对策树 364
10.5.2 状态空间上的序贯对策 369
10.5.3 其他序贯对策 372
10.6 连续状态空间 374
进一步阅读 377
习题 378
实现 379
第 11 章 传感器与信息空间 380
11.1 离散状态空间 381
11.1.1 传感器 381
11.1.2 历史信息空间 384
11.1.3 定义规划问题 385

11.2 衍生的信息空间 388
11.2.1 信息映射 388
11.2.2 非确定的信息空间 390
11.2.3 概率信息空间 392
11.2.4 有限记忆信息空间 394
11.3 离散状态空间的实例 394
11.3.1 基本的非确定性的实例 394
11.3.2 非确定的有限自动机 397
11.3.3 概率情形：POMDPs 400
11.4 连续状态空间 400
11.4.1 离散阶段信息空间 400
11.4.2 连续时间信息空间 401
11.4.3 衍生的信息空间 402
11.5 连续状态空间的例子 407
11.5.1 传感器模型 407
11.5.2 简单投影的例子 412
11.5.3 带大自然感测行动的例子 414
11.5.4 在没有传感器的情况下获得信息 416
11.6 计算概率信息状态 417
11.6.1 卡尔曼滤波 417
11.6.2 基于采样的方法 419
11.7 对策论中的信息空间 421
11.7.1 对策树中的信息状态 421
11.7.2 状态空间上对策的信息空间 424
进一步阅读 426
习题 427
实现 429
第 12 章 存在感测不确定性条件下的规划 430
12.1 一般方法 431
12.1.1 将信息空间作为一个大的状态空间 431
12.1.2 非确定的 *I*-空间中的算法 433
12.1.3 概率 *I*-空间中的算法（POMDPs） 434
12.2 定位 435
12.2.1 离散定位 435
12.2.2 连续定位的组合方法 440
12.2.3 定位的概率方法 442
12.3 环境不确定性与制图 444
12.3.1 栅格问题 445
12.3.2 Stentz 算法（D*） 450

12.3.3 未知连续环境中的规划 452
12.3.4 没有几何模型情况下的最优导航 456
12.3.5 概率定位与制图 460
12.4 基于可视性的追-逃问题 462
12.4.1 问题的表述 462
12.4.2 一个完备算法 465
12.4.3 其他类型 467
12.5 存在感测不确定性下的操作规划 468
12.5.1 原像规划 468
12.5.2 非抓握操作 473
进一步阅读 476
习题 477
实现 479

第Ⅳ部分 微分约束条件下的规划

第 13 章 微分模型 482
13.1 位形空间上的速度约束 482
13.1.1 隐含表示与参数表示 482
13.1.2 轮式系统的运动学 487
13.1.3 速度约束的其他实例 492
13.2 动力系统的相空间表示 495
13.2.1 通过增加维数来减小自由度 496
13.2.2 线性系统 498
13.2.3 非线性系统 499
13.2.4 通过增加积分器来对模型进行扩展 500
13.3 基本牛顿-欧拉力学 502
13.3.1 牛顿模型 503
13.3.2 质点运动 504
13.3.3 刚体运动 508
13.4 先进的力学概念 513
13.4.1 拉格朗日力学 514
13.4.2 一般拉格朗日表达式 518
13.4.3 欧拉-拉格朗日方程的扩展 521
13.4.4 哈密顿力学 524
13.5 多决策者 526
13.5.1 微分决策 526
13.5.2 微分对策论 527
进一步阅读 528

习题 529
实现 530
第 14 章　微分约束条件下基于采样的规划 531
14.1　简介 531
14.1.1　问题表述 531
14.1.2　不同类型的规划问题 533
14.1.3　相空间中的障碍物 535
14.2　可达性与完备性 538
14.2.1　可达集合 538
14.2.2　离散时间模型 540
14.2.3　运动本原 545
14.3　再论基于采样的运动规划 546
14.3.1　基本组成 546
14.3.2　系统模拟器 548
14.3.3　局部规划 551
14.3.4　微分约束下的一般框架 552
14.4　递增采样和搜索方法 553
14.4.1　在网格上进行搜索 554
14.4.2　结合状态空间离散化 559
14.4.3　RDT 方法 562
14.4.4　其他方法 565
14.5　微分约束条件下的反馈规划 566
14.5.1　问题的定义 566
14.5.2　带插值的动态规划 566
14.6　解耦规划法 568
14.6.1　解耦大问题的不同方法 568
14.6.2　规划与变换 569
14.6.3　具有路径约束的轨迹规划 571
14.7　基于梯度的轨迹优化 577
进一步阅读 579
习题 580
实现 580
第 15 章　系统理论与分析技术 582
15.1　系统的基本特性 582
15.1.1　稳定性 582
15.1.2　李亚普诺夫函数 584
15.1.3　可控性 586
15.2　连续时间动态规划 588
15.2.1　Hamilton-Jacobi-Bellman 方程 588

15.2.2 线性二次型问题 590
15.2.3 庞特里亚金最小值原理 591
15.3 一些轮式车辆的最优路径 595
15.3.1 Dubins 曲线 595
15.3.2 Reeds-Shepp 曲线 598
15.3.3 Balkcom-Mason 曲线 600
15.4 非完整性系统理论 602
15.4.1 仿射控制系统 602
15.4.2 确定一个系统是否是非完整的 604
15.4.3 确定可控性 611
15.5 非完整性系统的操控方法 616
15.5.1 使用 Philip Hall 基 616
15.5.2 采用正弦行动轨迹 621
15.5.3 其他操控方法 624
进一步阅读 625
习题 626
实现 627
参考文献 628
英汉词汇对照表 674

第Ⅰ部分　介绍性的资料

第1章 绪 论

1.1 从规划（的过程）到规划（的结果）[1]

规划对于不同的人来说意味着不同的含义。机器人学研究包含传感器、执行机构和计算能力的机械系统的自动化（也经常使用诸如自主系统这样的术语），它的一个基本需要是产生算法将人类的高层任务指标转化为如何进行运动这样的低层描述。运动规划和轨迹规划通常应用于这类问题。运动规划有时被称为“钢琴搬运工问题”。假设给出人和钢琴的精确的计算机辅助设计模型作为算法的输入，算法必须确定在不碰到任何其他物体的情况下如何将钢琴从一个房间移动到另外一个房间。大多数人在搬沙发或床垫上楼梯时都会遇到类似的问题。机器人运动规划通常忽略动力学和其他各种微分约束，而主要关注搬运钢琴所需要的平移和旋转。近年来的研究工作也考虑到了不确定性、微分约束、建模误差、最优性等其他方面因素的影响。轨迹规划通常是指应用机器人运动规划算法的解以及确定如何以遵守机器人机械限制的方式沿着规划解移动。

历史上，控制理论关心的是对由微分方程描述的物理系统的控制输入进行设计。这些包括机械系统（汽车或航空器等）、电子系统（噪声滤波器等），或者出现在化学、经济、社会学等领域中的系统。典型地，控制系统已经采用在执行过程中可以实现自适应响应的反馈策略，系统也关注稳定性问题，保证系统在动态情况下不至于失去控制。大量的研究着力于达到最优指标来最小化能量或时间等方面的消耗。在最近的控制理论文献中，运动规划有时是指构建一个非线性动力系统的输入，它驱动系统从初始状态运动到指定的目标状态。例如假设试图操作一架遥控飞机在冰面上滑翔，我们希望飞机离开目前的位置而运动到指定位置，那么是否能够设计一个算法来计算所需要的输入（即使在忽略模型不确定性的理想的仿真器上）？可能还会增加一些其他考虑方面，如不确定性、反馈和最优性；但是即使不考虑这些，问题也非常具有挑战性。

在人工智能方面，规划和人工智能规划这两个术语加入了更多的离散含义。其目的是解决魔方或拼图游戏这样的难题（见图 1.1），或者完成堆积木这样的离散任务，而不是将钢琴在连续空间中进行移动。尽管这些问题可以用连续空间进行建模，但是更自然的是定义一个有限行动集合，并将其应用于离散状态集合，通过给出相应的行动序列来构造一个解。历史上，认为规划不同于问题的求解，但是近年来它们之间的区别似乎正在变得模糊。本书并不刻意对它们加以区分，更多的工作体现在规划问题的描述语言上。尽管还对其他问题进行了论述，但那些问题不是本书的重点。一些决策理论的思想最近已经融入了人工智能规划问题、不确定性问题建模、对抗性游戏和最优性。这些问题是非常重要的，它们的详细内容将在第 III 部分进行介绍。

---

1 译者注：在翻译中将规划过程（Planning）及其结果（Plan）统称为规划。

规划的概念已经应用于人工智能、控制理论、机器人等领域，由于规划问题所涉及的范围非常广，你可能会有这样的疑问：规划问题是否有一个确定的含义？或者仅仅是关于那些可以认为是规划问题的实例的任何事情？稍后将简要介绍规划问题的一些公共组成部分。但首先我们将规划问题作为算法的分支来考虑，因此本书的书名为“规划算法”。本书重点介绍在几个学科中出现的算法以及规划问题的计算问题。另一方面，这并不意味着规划算法仅参考从事一般算法的研究机构中现有研究人员的成果。本书也不局限于单纯算法所主要关注的组合方法与渐近复杂性分析，本书的重点在于包括大量的概念，它们不一定是算法的，但它们有助于进行建模、求解和分析规划问题。

读者会很自然地提出以下问题：规划的结果是什么？如何表示规划结果？如何计算规划？规划的结果能完成什么？如何评估它的质量？什么人或者什么问题会使用规划的结果？本章将回答这些问题。关于规划结果的使用者，这依赖于具体的应用。在大多数应用中，算法执行规划结果，但是使用者甚至可以是人。例如规划算法可以给人们提供一个投资策略。

在本书中，规划的使用者通常指的是一个机器人或者一个决策者。在人工智能及其相关领域，近年来则普遍采用“智能体”这个名词，也可能是“软件智能体”。控制论通常将决策者称为控制器。这里有时将规划称为策略或者控制律（Control law）。在对策论中，决策者被称为对策者。不管这些具体学科中的术语怎样，本书关心的是能够为一个或者多个决策者提供一个策略的规划算法，因此机器人、智能体、控制器等这些术语是可以互换的。

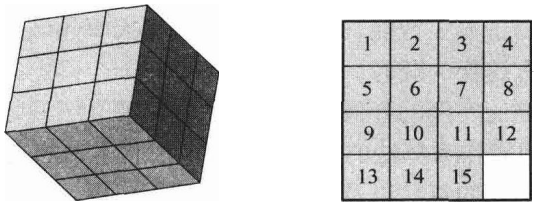
1.2 实例与应用

规划问题所涉及的内容非常丰富。本节提供一些规划的例子和应用来激发读者进一步阅读的兴趣。

为什么研究规划算法？这里至少有两个原因。首先，尝试用机器来解决那些人类难以解决的问题。这在规划问题建模、设计有效算法、开发鲁棒性强的实现等方面都有令人激动不已的挑战。其次，在理论和实践等许多方面，规划算法已经取得了巨大的成功，包括机器人、机械制造、药品设计、航空应用，等等。近年来的飞速发展预示着可能会出现更多的规划应用领域。这是一个致力于研究规划算法以及规划的应用与发展的时代。

离散的问题与操作 第2章涉及离散规划，离散规划可以解决如图1.1所示的那些我们所熟悉的难题。也可以应用于象棋、桥牌等游戏[899]。离散规划技术已经用于航天应用中，包括火星探测器和对地观测卫星[209,384,897]。当与连续空间中的规划结合在一起时，它们可以解决诸如确定如何将金属片折叠放入一个复杂的物体中这种复杂任务[422]。折叠硬纸盒的相关问题见第7.5节。

运动规划问题 因为移动零部件存在规则性与对称性，图1.1所示的难题很容易进行离散化。图1.2显示了一个缺乏这些特征且要求在连续空间中运动的问题。这些问题可以采用第II部分所介绍的运动规划技术来解决。这个难题设计用于挑战人类以及运动规划算法。在一台标准个人计算机上采用第5.5节中的技术能够在数分钟内解决该问题。一些其他难题将作为评估规划算法的标准得以研究和发展。



(a) (b)

图 1.1 魔方（a）拼图（b）以及其他相关游戏都是离散运动规划问题的例子

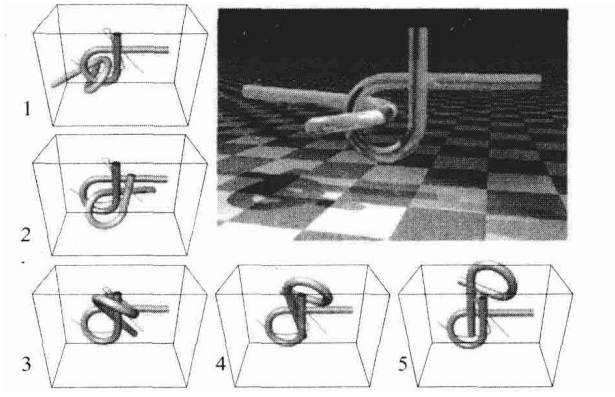


图 1.2 还记得这样的智力玩具吗？设想用算法来解决其中一个问题，目的是将它们两个分开；这个例子被称为 Alpha1.0 问题；它是由 Boris Yamrom 发明的，并被德克萨斯 A&M 大学的 Nancy Amato 当作一个研究的标准；James Kuffner 提供了它的解并制作了动画（见文献[561]）

自动装配问题 尽管图 1.2 中的问题以游戏的形式出现，但是在一些重要的应用中也会出现相似的问题。例如图 1.3 显示了一个自动化装配问题，其中需要软件来确定一个雨刷器电动机是否能够从车体的缝隙中插入（和移出）。传统地，这种问题可通过构造物理模型来解决。这个开销很大，且非常耗时的设计过程可以在软件虚拟环境中通过直接操作 CAD 模型来完成。

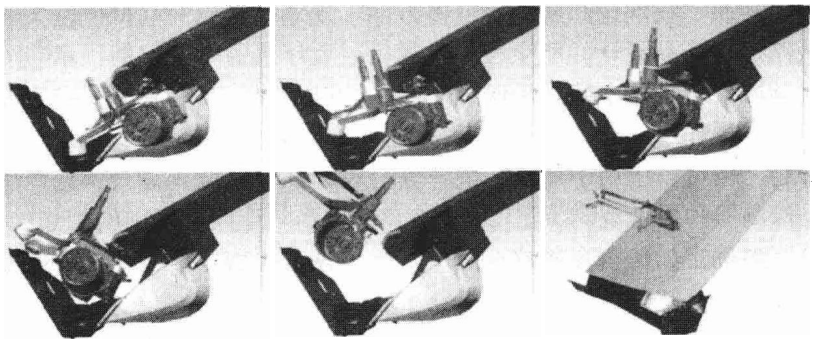


图 1.3 一个自动装配任务涉及从车体中插入或者移出一个挡风玻璃雨刷器电动机，这个问题可以用 Kineo CAM 的运动规划软件来解决

雨刷器仅仅是众多例子中的一个。对工业最广泛的冲击来自于 Kineo CAM 的运动规划软件。它已经从 Tecnomatix 集成到 Robcad（eM-workplace），这是一个专用于设计世界

上很多工厂中的机器人部件而开发的先进设计工具。它们的软件已经应用于 Renaut、Ford、Airbus、Optivus 以及其他一些世界知名公司的装配线上。其他公司和机构也已经参与了运动规划工具的开发（一些工程是秘密进行的，这里无法进行描述）。第一个应用于装配问题的运动规划的例子由文献[188]给出。

自动装配中的密封 图 1.4 显示了模拟在 Volvo 汽车厂完成密封任务的机器人。密封是一个使用机器人沿着车体焊缝喷射黏性材料的过程，这能够防止灰尘和水进入车体而引起的腐蚀。整个机器人部件用 CAD 工具设计，它能够为运动规划软件提供必要的几何模型。图 1.4 所示的是为 Volvo 汽车和其他使用 FCC 开发的运动规划软件所解决的许多问题之一。使用运动规划软件，工程师仅需要指定所要完成密封的高层任务，机器人的运动方式则自动进行计算。这可以在制造过程中节省大量的时间和财力。

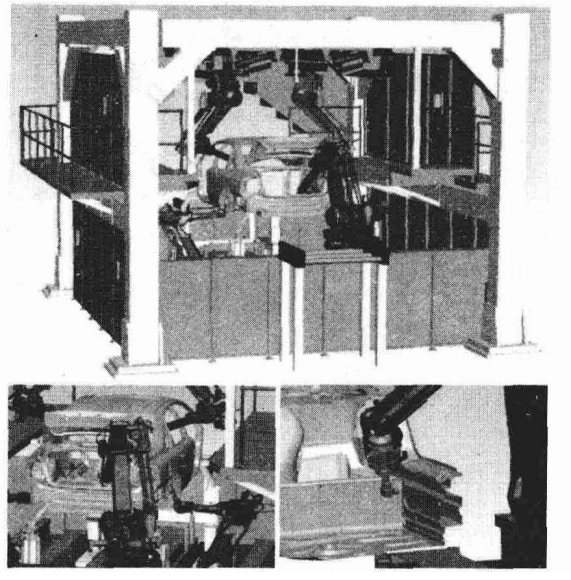


图 1.4 运动规划被应用于汽车制造中的焊接过程，Fraunhofer Chalmers Centre（FCC）开发的规划软件用在瑞典 Volvo 汽车生产厂

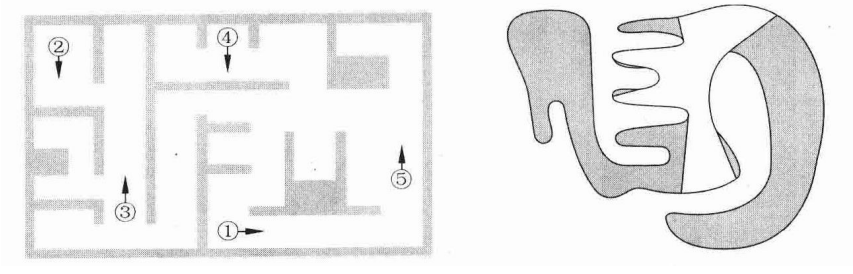
搬运家具 图 1.5 所示的问题涉及用 3 个安装有机械臂的可移动机器人搬运一个巨大的钢琴通过一个房间。这被称为钢琴搬运工问题。其中必须避免机器人与其他家具之间发生碰撞。这个问题非常复杂，因为机器人、钢琴和地面形成一个闭合运动链系，相关内容将在第 4.4 节和第 4.7 节论述。

可移动机器人导航 对可移动机器人的一个基本要求是能够在室内进行导航，如图 1.6(a)所示。一个机器人可能被要求完成诸如建立一个环境的地图、确定它在地图中的精确位置、到达指定地点等任务。获取和操作来自于传感器的信息是十分具有挑战性的，这将在第 11 章和第 12 章介绍。机器人的运行有较大的不确定性。在极端情况下，如果有一些传感器来精确估计环境以及机器人在环境中的位置和方向，那将是非常有利的。许多系统都是如此，如图 1.7 所示的机器人系统，它构建了一个环境的地图。在某些情况下，人们可能更倾向于发展低成本和高可靠性机器人，它们能够在不具备感测能力或者具备较少感测能力的条件下来完成特殊任务。第 11 章和第 12 章将对此进行权衡。存在不确定性

情况下的规划是第III部分的内容。

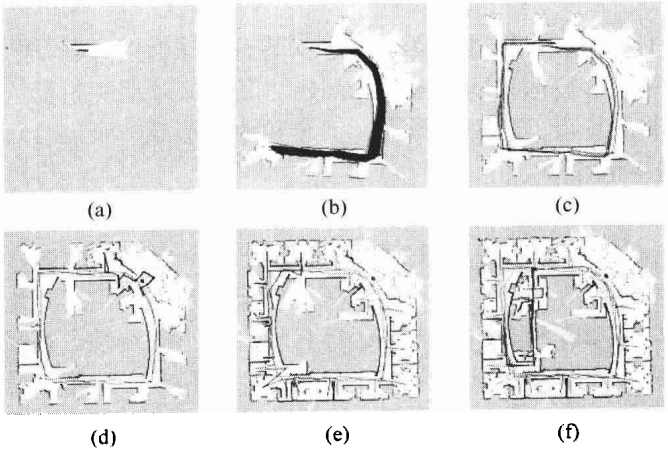


图 1.5 用可移动机器人搬运一架钢琴[246]



(a) (b)

图 1.6 (a) 几个机器人试图进入一个室内环境，并避免与墙和相互之间发生碰撞；(b) 假设在山洞中用灯笼搜索失踪人员



(a) (b) (c)

(d) (e) (f)

图 1.7 一个可移动机器人能够构建它所处环境的地图，同时确定其自身的位置；这是通过使用激光测距传感器并在信息空间上进行有效的贝叶斯计算来实现的

如果有多个机器人，那么会产生许多其他的问题。机器人之间如何进行通信？如何集成这些信息？它们之间的协调是集中的还是分散的？如何避免它们之间的碰撞？它们每个完成不同的任务，或者要求它们以某种方式进行合作？如果它们以某种方式进行竞争，那么可能会与对策论相关。因此一些对策论的内容会出现在第 9.3 节、第 9.4 节、第 10.5 节、第 11.7 节和第 13.5 节。

捉迷藏 可移动机器人的一个重要的任务是玩捉迷藏游戏。假设进入一个完全黑暗的山洞。你有一个灯笼，要求你搜索洞中的其他人，如图 1.6(b)所示。那么头脑中会出现几个问题：是否存在一个策略来保证我能够发现其他每个人？如果不存在，那么在任务完成之前，需要多少其他搜索者？我下一步向什么地方运动？我能避免多次搜索同一个地方吗？这种场景在许多机器人应用中出现。机器人放入一个监控系统中，监控系统使用带有多种传感器（运动传感器、热传感器、摄像机等）的可移动机器人。在有多个相互之间没有通讯或者较少通讯的机器人场景中，这种策略帮助一个机器人确定其他机器人的位置。一个机器人甚至可以试图定位另外一个故障的机器人。在机器人领域之外，软件工具的发展帮助人们系统地搜索或者遍历复杂环境，诸如搜索与营救、清理有毒物品、建筑物的安全设计等应用。这些问题非常复杂，因为必须仔细计算搜寻策略来避免逃避者逃回已经搜索过的地方。第 11 章的信息空间的概念在解决这个问题中就显得非常关键。一个求解捉迷藏游戏的算法见第 12.4 节。

视频游戏人物 图 1.6(b)中的问题可能使你想起视频游戏。在 Pacman 游戏中，幽灵就被设计成追逐游戏玩家。现代视频游戏涉及能够展现复杂行为的人物。规划算法使游戏设计者能够在更高层面上进行人物的编程，期望这些游戏人物能够以智能方式确定其自身的运动。

目前在规划算法和视频游戏之间存在巨大的距离。一些规划算法设计者最近才开始考虑视频游戏所特别关注的问题。视频游戏设计者目前不得不投入大量精力修改已有的技术来解决他们所关心的问题。提供给游戏设计者的最新书籍，见文献[154,373]。

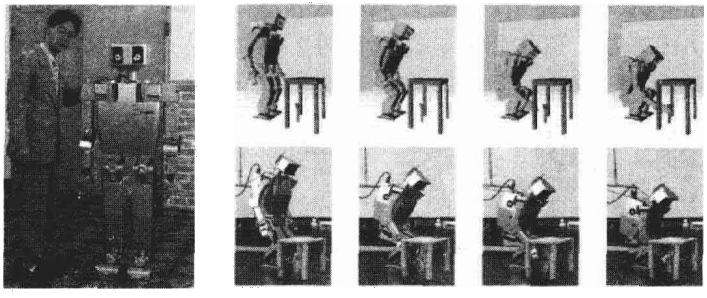
虚拟人与类人型机器人 除视频游戏以外，人们还对开发虚拟人有广泛的兴趣。如图 1.8 所示，计算机图形学关注于计算机生成的动画。动画绘制者希望开发一些具有人类



图 1.8 上部：一个规划算法计算数字演员到达一个电冰箱所要进行的运动[502]；左下：一个数字化演员（digital actor）与一个虚拟机器人下棋[547]；右下：一个规划算法计算 100 个数字演员通过有障碍物的地形的运动[594]

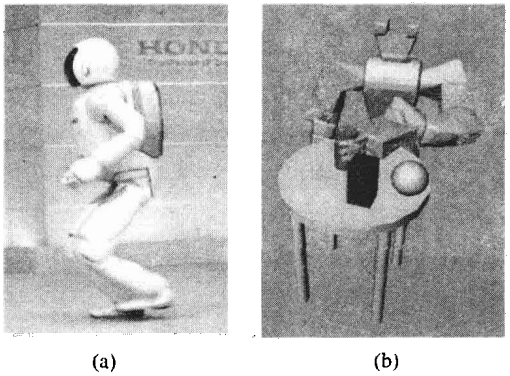
演员所具有的典型特征的数字演员，同时从更高层面上设计这些数字演员的运动。一帧一帧地指定所有的运动是非常乏味和费时的。这种领域的规划算法发展很快。

为什么止步于虚拟人呢？日本机器人界先进类人型机器人的发展已经对整个世界产生了巨大鼓舞。在 1997 年，本田公司的一种类人型机器人震惊了世界，该机器人能够上楼梯，能够从失去平衡的状态恢复过来。自那时起，许多公司和研究机构已经改进了类人型机器人的设计。尽管大多数机械问题已经解决，但是依然存在感测和规划这两个基本困难。如果不能进行编程以接收和自主执行高层命令，那么类人型机器人还有什么优势呢？图 1.9 展示了东京大学的研究工作，他们将仿真环境下计算类人型机器人的规划应用于真实机器人。图 1.10 展示了日本自动化工业领域的机器人工程。



(a) (b)

图 1.9 (a) H7 类人型机器人及其开发人员 S.Kagami；它是由东京大学的 JSK 实验室开发的；(b) 将虚拟现实和真实现实放在一起；规划算法计算一个类人型机器人的稳定运动来抓住地面上一个物体[564]



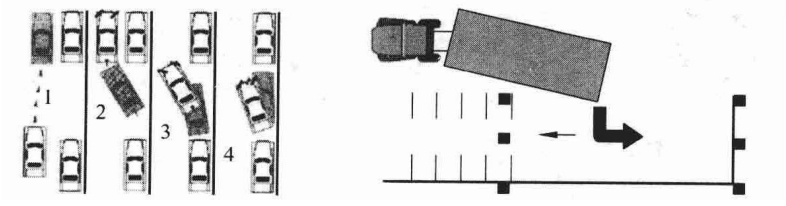
(a) (b)

图 1.10 日本汽车工厂的类人型机器人

(a) 本田公司最新的 Asimo 机器人能够以 3km/h 的速度前进；(b) 在丰田公司类人型机器人中，规划与视觉进行融合以使它能够规划抓握物体的过程[451]

停靠汽车和拖车 前面讨论的规划问题还没有涉及第Ⅳ部分所关注的微分约束。考虑慢速运动汽车的停靠问题，如图 1.11 所示。对多数人来说平行移位停车并不困难，但是停靠拖车就困难得多。想象平行移位停靠一辆机场行李车的困难。第 13 章有相关例子。什么使得这些问题如此具有挑战性？一辆汽车只能向车轮指向的方向运动。围绕障碍物所进行的移动因此变得具有挑战性。如果所有四个车轮能够转到任何方向，这个问题就不存在了。非完整性规划（nonholonomic planning）就是围绕停车问题和一些其他问题而展开的。

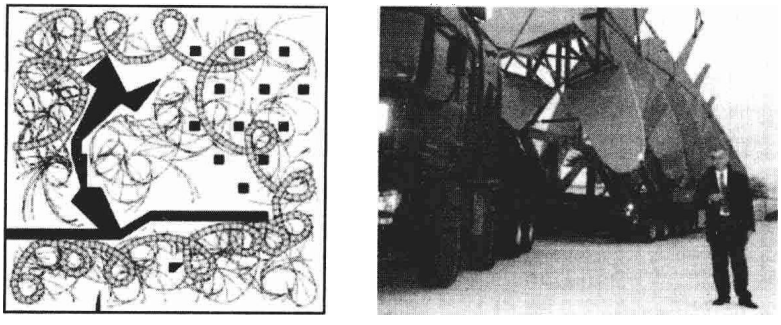
图 1.12(a)展示了一个这样的驾驶问题。非完整性规划算法在工业中得到充分的发展，图 1.12(b)展示了它在一个极其复杂车辆中的应用。



(a) (b)

图 1.11 一些停车的说明来源于官方的驾驶员考试手册

(a) 停泊汽车（来源于 2005 年 Missouri 驾驶指南）；
(b) 停靠一辆拖车；人和规划算法都能够解决这些问题

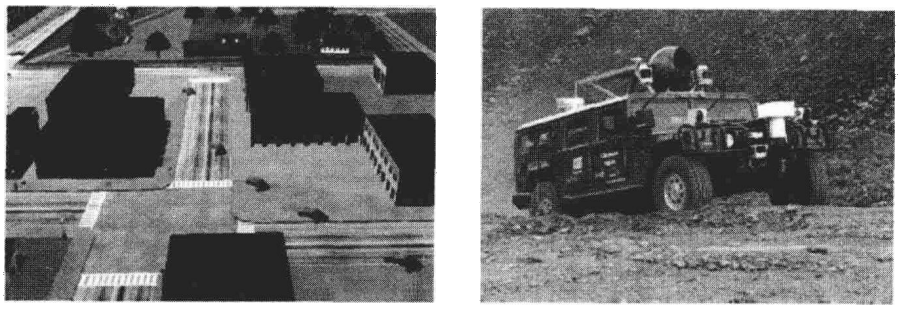


(a) (b)

图 1.12 (a) 图中显示了一辆必须向前行驶并只能左转的汽车的避碰路径；你能够自己发现这种解吗？对于多个规划算法来说，这是一个很容易的问题；(b) 这辆巨大的卡车设计用来运输空客 A380 的部件； Kineo CAM 开发了非完整性规划软件，它规划通过村庄的路线避开障碍物，并满足 20 个转向轴的微分约束；照片上还有非完整性规划的工程师 Jean-Paul Laumond

汽车驾驶 现在考虑高速行驶的汽车。当速度增加时，由于存在动量，所以汽车必须当作动力系统来对待。汽车不能够再瞬时进行起步与停止，但瞬时起步与停止对泊车问题来说是合理的。尽管有些规划算法讨论这些问题，但是仍然有一些尚未解决的研究问题。这些问题对工业的影响并不如图 1.3 和图 1.4 所示的普通运动规划对工业的影响大。在设计过程中考虑动力学问题，那么在构造汽车以前就可以实现性能和安全性的评估。图 1.13 展示了一个由一个规划算法计算而来的解，它确定如何驾驶一辆高速汽车通过一个城镇，同时避免与建筑物发生碰撞。规划算法甚至可以用来评定一辆运动型汽车当紧急停车时是否会发生侧翻。通过在设计阶段的仿真与规划来确定设计缺陷，可以节省大量的时间和成本。与此相关的一个问题是需要进行验证，机械系统必须进行测试来保证它的性能像期望的那样，不管在使用过程中发生什么样的问题。规划算法在这过程中是有帮助的。例如算法可以模拟猛烈地碰撞一辆汽车，从而确定是否需要修改设计方案。

除辅助设计过程以外，考虑动力学的规划算法可以直接植入机器人系统。如图 1.13(b)展示了一个涉及前面提到的大多数问题的复杂组合问题。在高低不平的未知地形上高速行驶时会涉及动态性、不确定性和避碰障碍等问题。本书中的许多研究问题到目前依然没有



(a) (b)

图 1.13 安全驾驶

(a) 用规划算法来驾驶汽车快速通过障碍路段[201]；(b) 在 DARPA 汽车挑战赛中由 Carnegie Mellon 大学研发的一辆汽车高速行驶在颠簸的路面上（由 Red Team 提供）

得到解决。

空间飞行 驾驶很自然地导致了飞行。规划算法可以导引自主飞行直升机通过障碍物。算法也可以计算航天器为避免与诸如国际空间站等大型结构发生碰撞所需要的推力。在第 14.1.3 节描述了可重复使用航天器再入轨迹的设计问题。诸如太阳帆等星际航天器的使命规划也可以通过规划算法来实现[439]。

药物设计 规划算法甚至对远离机器人的领域也产生了巨大影响。蛋白质折叠和药物设计是两个主要问题。在这两种情况下，科学家试图通过研究大组织分子之间的相互作用来解释生物体的行为。这些分子通常都是柔性的。药物分子很小，蛋白质通常有成千上万个原子。对接（docking）问题涉及确定一个柔性分子是否可以插入一个蛋白质孔中，如图 1.14 所示，而且还要满足诸如保持低能量等其他约束。一旦将几何模型应用于分子，问题就与图 1.3 所示的装配问题非常相似，那么就可以用运动规划算法来解决。相关内容见第 7.5 节以及第 7 章结尾的文献。

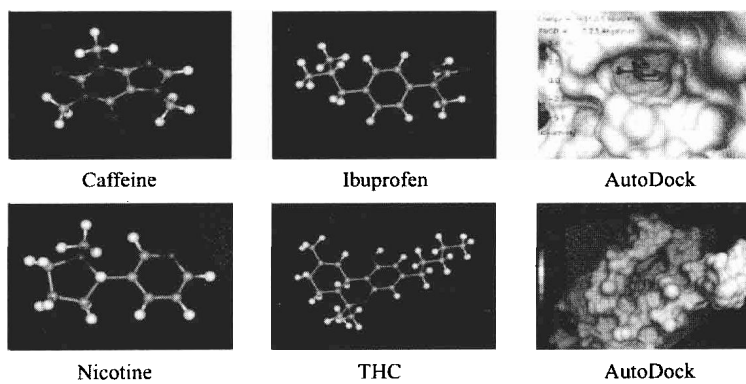


图 1.14 左边：几个熟悉的药物被描绘成球杆模型（纽约大学 MathMol 实验室提供[737]）；右边：AutoDock 软件包所显示的蛋白质—配位体对接的三维模型（Scripps 研究中心提供）

前景 规划算法的应用领域远远不止以上所介绍的问题。在一些情况下，规划的建模方法、算法、应用软件都得到了长足的发展。在其他一些情况下，大量的研究工作还将规划方法带到更加广阔的潜在领域。对参与规划算法研究发展的人员来说，未来充满了希望。

1.3 规划的基本组成

尽管本书的内容涉及大量的模型和问题，但是它们都有几个基本组成，这些基本组成贯串于关于规划的所有主题。

状态 规划问题涉及能够捕获所有可能状况的状态空间。状态可以表示诸如机器人的位置和方向、拼图玩具中碎片的位置，或者直升机的位置和速度等。它们可以是离散状态空间（有限或者可数无限）和连续状态空间（不可数无限）。状态空间通常被规划算法隐含表示。在多数应用中，状态空间的大小（即状态的个数或者组合复杂度）非常大以至于无法显式表示。然而在规划问题的表述中以及在解决规划问题的算法的设计和分析中，状态空间的定义都是重要组成部分。

时间 所有规划问题都涉及随时间进行的决策序列。时间可以显式建模，例如驾驶汽车尽快地通过一个障碍路段。也可描述行动一个接一个顺序地执行，这其中虽然没有明确表示时间，但却隐含着时间信息。特定时间并不重要，但是必须保持正确的顺序。隐含时间的另外一个例子是钢琴搬运工问题的解；搬运钢琴可以转换成一个随时间变化的动画，但是在规划中并不特别指定速度。就像在状态空间情况下一样，时间可以是离散的或者连续的。在连续情况下，假设规划产生了决策的一个连续统。

行动 规划产生对状态进行操作的行动。在人工智能中行动和算子是很常见的术语；在控制论和机器人中，相关的术语是输入和控制。在规划表述中的某些地方，必须指定当采取行动时，状态是如何变化的。对离散时间情况，这可以作为状态值函数来表达，对连续时间情况，则可以用常微分方程来表达。对多数运动规划问题来说，通过指定连续状态空间中的一条路径来避免显式地提及时间。对微分方程进行积分可以得到这样的路径，但这是不必要的。对某些情况，行动可能由大自然产生，大自然干扰输出结果并不受决策者的控制。这将不确定性引入规划问题，相关内容参见第 10 章。

初始和目标状态 规划问题通常涉及从某初始状态开始试图到达某特定目标状态或者目标状态集合中的任何状态。这种情况需要采取行动。

准则 准则根据状态以及所采取的行动来编码一个规划的期望结果。基于准则的类型，规划通常关注两个不同类型的问题：

1. 可行性：寻找一个能够引起到达目标状态的规划，而不管它的效率如何。
2. 最优性：规划不仅要实现到达目标状态之目的，而且还需满足某种最优性能指标。

对本书考虑的大多数问题来说，可行性已经具有足够的挑战性；实现最优性则更加困难。因此许多注意力关注于发现问题的可行解，而不是最优解。机器人学、控制论与相关领域的大多数文献都关注最优性，但是对许多我们感兴趣的问题来说，这是不必要的。在许多应用中，甚至很难对最优性准则进行表述。即使表述了满意的准则，也可能无法得到实际的计算最优规划的算法。在这种情况下，得到可行解总比无解好。非常幸运，一些算法得到的可行解离最优并不远。对于涉及概率不确定性的问题，则频繁出现优化问题。通常用概率来得到期望代价意义下的最优性能指标。可行性通常与不确定性的最坏情况分析有一定的联系。

规划 一般情况下，规划给决策者提供一个特定的策略或者行为。规划可能只是指定一个需要采取的行动序列；但是也可能复杂得多。如果不可能预测将来的状态，那么规划可以将行动作为状态的函数。在这种情况下，不管将来的状态如何，都可以确定相应的行动。采用其他领域的术语，这能够进行**反馈**或者**反应规划**。当然也可能有无法测量状态的情况。在这种情况下，相应的行动必须从目前可用的任何信息中产生。这通常称为信息状态，规划得到的行动以这些信息状态为条件。

1.4 算法、规划器与规划

1.4.1 算法

什么是规划算法？这是一个很难回答的问题，本书也没有给出它的精确数学定义。取而代之的是一个一般意义的解释以及规划算法的例子。一个基本问题是：什么是算法？一种答案是典型的图灵机模型，它用来定义理论计算机科学中一个算法。图灵机（Turing machine）是一个有特定开头的有限状态机，它可以沿着一个无限长的带子进行读和写，如图 1.15 所示。Church-Turing 的论文论述到一个算法就是一个图灵机（详细内容见文献[465,892]）。算法的输入是一串字符（通常是二进制串），然后写入带子。图灵机读取字符串、执行计算，然后决定是接受还是拒绝字符串。这个版本的图灵机仅能解决决策问题；对它进行扩展就可以产生其他所期望的输出，例如一个规划。

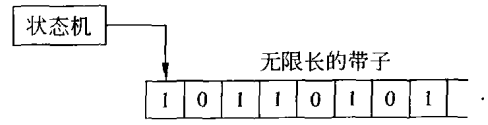


图 1.15 根据 Church-Turing 的论文，算法的概念等价于图灵机的概念

图灵机模型对本书中的许多算法来说都是合理的；但是对有些情况，图灵机模型并不合适。在某些情况下，采用图灵机的问题在于规划所得到的结果经常与物理世界互相影响。如图 1.16 所示，图灵机和环境之间的边界是一条随意的线，这条线随问题的不同而不同。一旦确定这条线，那么传感器提供关于环境的信息；这给图灵机提供了输入。图灵机执行这些行动，从而对环境产生刺激。刺激可能以传感器所能够测量的某种方式改变环境。因此图灵机与环境在行动执行过程中是紧密耦合的。这是机器人和其他一些使用规划的领域的基础。

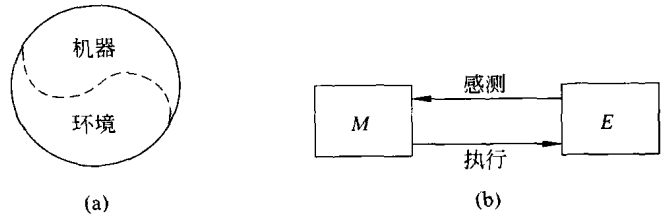


图 1.16 (a) 机器与环境的界限被认为是根据具体情况可以用许多方法绘制的一条任意的线；
(b) 一旦划定界限，那么假设机器 M 通过感测和刺激与环境 E 发生相互影响

使用图灵机作为算法的基础通常意味着在算法产生决策之前，物理世界必须首先仔细建模并写在带子上。如果在算法执行过程中，物理世界发生改变，那么不知道将会发生什么。例如一个可移动机器人能够在一个杂乱的环境中移动，这个环境中有人正在散步。另外一个例子，机器人可能将一个物体抛在桌子上，但无法精确预测该物体怎样停止。可以用传感器测量结果，但是确定有多少信息需要明确建模并写在带子上是很困难的。对这类问题来说在线算法模型是更适合的[513,771,893]，但是它依然无法获取对本书中所有内容来说足够广的算法的概念。

发生在物理世界中的过程比状态机与充满符号的一段带子之间的相互影响更加复杂。如图 1.17 所示，通过假设一个与一排开关相互影响的机器人来对带子进行模拟。开关与带子的作用相同，机器人载有一台可以模拟有限状态机的计算机[1]。机器人与环境之间的复杂的相互影响产生一些其他计算模型[2]。因此将以一种与计算理论中相比较而言不太正规的方式应用“算法”这个术语。本书中规划器和规划的结果都被认为是算法。

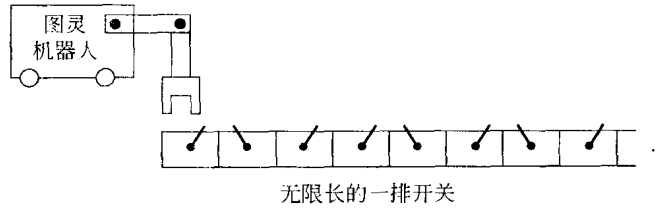


图 1.17 用一个机器人和开关的一个无限序列来模拟图灵机；但是通过操作，还可以得到图灵模型以外的许多其他类型的行为

1.4.2 规划器

规划器构造规划的结果，可能是一台机器或者一个人。如果规划器是一台机器，那么它通常被当作规划算法。在许多环境下，它是严格图灵机意义下的算法，但这是不必要的。在某些情况下，人因为进行规划而成为规划器。例如人可以设计一个与环境相联系的状态机（见第 12.3.1 节）。这种情况下没有其他的输入，因为人担当起了算法的角色。给定规划模型作为人的输入，人“计算”规划。

1.4.3 规划

一旦确定了一个规划，有三种方法使用规划：

1. 执行：可以在模拟情况下执行规划，也可以在物理世界中的机械装置（机器人）中执行。

2. 改进：改进使之成为更好的规划。

---

1 当然在物理世界中，无限长的带子似乎是不可能的。还有另外一种图灵机，其中虽然带子的长度有限，但是长度足够处理给定的输入。对于论述来说，这可能是合适的。

2 文献[817]中讨论了机械系统的计算。文献[120]介绍了实数上的计算模型。

3. 层次包含：将规划作为一个行动融入到更高级别的规划中。

现对它们进行介绍。

执行 规划通常由一台机器来执行。人也可以执行它，但是本书关注机器执行的情况。机器执行通常有两种类型。第一种如图 1.18(a)所示，其中规划器产生规划，规划以某种方式作为机器的输入。在这种情况下，机器被认为是可编程的，在执行以前，它可以从规划器那里接收可能的规划。通常假设一旦给定规划，那么机器变成自主的，它不再与规划器进行交互。当然这种模型可以扩展到允许机器当接收到更好的规划时加以改进提高；但是为讨论规划，我们希望一个严格的关于自主性的概念。在实际情况下，该方法并不禁止对规划进行更新；但是这并不必要，因为规划设计中已经考虑到了执行过程中的新的信息。

机器执行规划的第二种类型如图 1.18(b)所示。在这种情况下，由规划器产生的规划编码了整个机器。规划就是一个用来解决规划器特定任务的专用机器。在这种阐述下，我们可以设计一个足够完成任务的最简单的机器。如果规划编码为有限状态机，那么它有时可以被认为是图灵意义下的算法（依赖于机器连接到带子是否保护它的操作）。

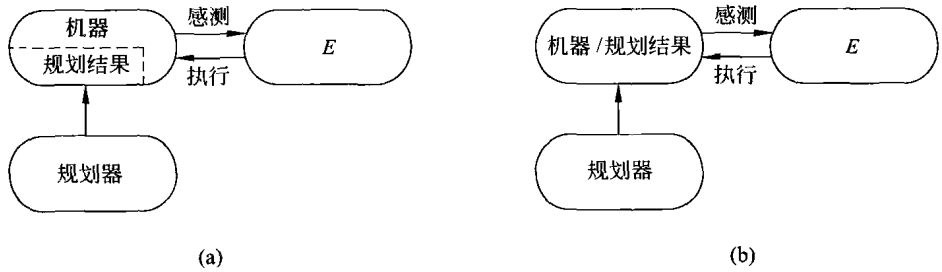


图 1.18 (a) 一个规划器产生机器可以执行的规划；规划器可能是机器本身或者一个人；(b) 规划器可以对整个机器进行设计

改进 如果对规划进行改进，那么规划器将它作为输入来得到一个改进的新规划。新规划可能考虑了问题的更多方面，或者只是更有效率。改进可以不断重复使用以产生一个改进后的规划序列，直到最后一个规划被执行。图 1.19 展示了一个在机器人中使用的改进方法。例如考虑移动一个室内可移动机器人。第一个规划产生了一个通过建筑物的无碰撞路径。第二个规划将路径变换成满足基于车轮运动的微分约束（回忆图 1.11）。第三个规划考虑如何沿着路径以不同的速度移动机器人，同时还要满足动量约束。第四个规划融合反馈来保证机器人距离规划的路径尽可能近，尽管存在不可预测的行为。关于这个方法进一步的内容以及对它的权衡折中出现在第 14.6.1 节。

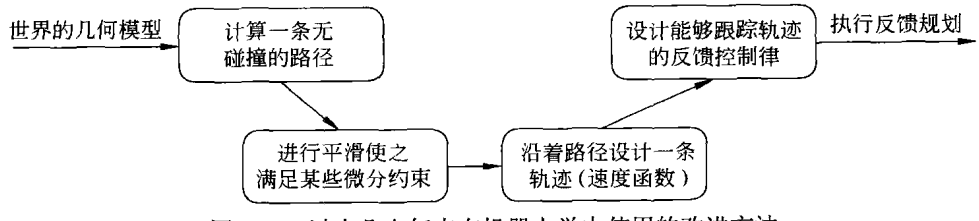


图 1.19 过去几十年来在机器人学中使用的改进方法

层次包含 在层次包含下，规划作为一个行动而融入到一个更大的规划中。原始规划

可以想象为一个更大规划的子规划。为了取得成功，对原始规划来说重要的是保证能够适时终止，以使更大规划能够执行更多所需要的行动。层次包含可以多次进行，这使规划成为一个有根的树。这产生了层次规划（hierarchical planning）的一般模型。树中的每个顶点就是一个规划。根顶点表示主规划。任何顶点的孩子都作为行动而融入到顶点的规划中。对树的深度或者每个顶点的孩子的数量没有限制。在层次规划中，机器与环境之间的分割线可以在多个位置进行划分。例如关于机器 M_1 的环境 E_1 可能包含其他机器 M_2，M_2 与其环境 E_2 相互影响，如图 1.20 所示。层次规划的例子见第 7.3.2 节和第 12.5.1 节。

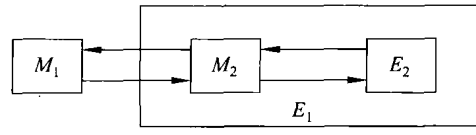


图 1.20 在一个层次模型中，一个机器的环境本身可能包含另外一个机器

1.5 本书的组织安排

这里简单介绍本书的内容。

第 I 部分：介绍性的资料

这部分为本书的其他部分提供基础背景材料。

第 1 章：绪论

本章提供一些应用前景，包括一些规划算法的例子和应用。

第 2 章：离散规划

本章介绍规划的最简单形式，可以作为进一步阅读本书其余部分的出发点。从这里，你可以进入第 II 部分，或者直接到第III部分。第 2.1 节和第 2.2 节对第 II 部分非常重要，第 2.3 节对第III部分非常有用。

第 II 部分：运动规划

本部分问题和算法来自于机器人学。但是这些方法非常具有一般性，所以可以应用于其他领域，比如计算生物学、计算机辅助设计、计算机图像学。它还有一个更能够精确反映规划类型的名称，那就是“连续状态空间中的规划”。

第 3 章：几何表示与变换

第 3 章为表达运动规划问题而给出重要背景。第 3.1 节将描述如何构造几何模型，其余部分介绍如何变换这些模型。第 3.1 节和第 3.2 节对后续章节来说是很重要的。

第 4 章：位形空间

本章介绍拓扑概念以及采用拓扑来表述位形空间，位形空间就是运动规划中出现的状态空间。第 4.1 节、第 4.2 节和第 4.3.1 节将对理解后续章节非常重要。除了以前提到的部分，整个第 4.3 节为第 6 章中的组合方法提供了非常有用的背景。

第 5 章：基于采样的运动规划

本章介绍的运动规划算法在近年来的文献中出现得非常多，并在机器人领域内外得到广泛应用。如果你理解了用位形空间表示连续状态空间的基本思路，那么大多数概念都应

该是可以理解的。除了运动规划和机器人学以外，它们甚至应用于其他一些连续状态空间问题。第 14 章再次讨论基于采样的规划问题，但是在微分约束条件下。

第 6 章：组合运动规划

本章介绍的算法有时称为**精确算法**，因为它们建立起离散表示，却不损失任何信息。它们是完备的，即如果解存在，那么它们必须发现问题的解；否则，它们报告无解。基于采样的算法在实际中非常有用，但是它们仅能够实现较弱的完备性。

第 7 章：基本运动规划的扩展

本章介绍的一些问题和算法是对第 5 章和第 6 章方法的扩展，它们都可以沿着这些章节的思路来理解。第 7.4 节讨论闭合运动链系的规划；这要求对第 4.4 节中的附加材料进行理解。

第 8 章：反馈运动规划

这是一个过渡章节，它将反馈引入运动规划，但是仍然没有引入微分约束，微分约束在第Ⅳ部分进行介绍。第 II 部分以前的章节关注开环规划的计算，这意味着在规划执行过程中所发生的任何误差都予以忽略，规划依然照常进行。使用反馈产生的闭环规划对规划执行过程中不可预知的事件产生反应。

第Ⅲ部分：决策论规划

第III部分的另外一个名称是“存在不确定性条件下的规划”。第III部分大多数内容讨论离散状态空间，可以从第 I 部分直接跳到这里。但是，一些内容扩展到连续空间；为理解这些部分，学习第 II 部分的部分内容是有益的。

第 9 章：基本决策理论

本章的主要思路是为面临来自于其他决策者干扰的决策者设计最好的决策。其他的决策者可能是真正的对手，也可能是为了建模不确定性而假想的。本章关注一步决策，为第III部分提供了一个构造模块，因为不确定性条件下的规划可以当作多步决策。

第 10 章：序贯决策理论

本章采用第 9 章的概念，并将一系列基本决策问题连接起来。来自第 2.3 节的动态规划概念在这里非常重要。对本章中的所有问题，假设目前的状态总是已知的。所有不确定性都是关于对未来状态的预测，这与测量目前状态不同。

第 11 章：传感器与信息空间

本章将第 10 章的表述扩展到当规划执行时目前状态未知的情况。关于状态的信息是从传感器感测和从以前所采取的行动的记忆中而得到的。信息空间对于具有感测不确定性的问题来说就像位形空间对于运动规划一样。

第 12 章：存在感测不确定性条件下的规划

本章中的几个规划问题和算法涉及感测不确定性。这包括诸如定位、地图绘制、追逃以及操作等问题。所有这些问题在信息空间中的规划思路下是统一的。

第Ⅳ部分：微分约束条件下的规划

这可以看作是第 II 部分的继续。在运动规划中出现的连续状态空间中存在全局（障碍物）和局部（微分）约束。也考虑动力系统，它产生包含位置、速度等信息的状态空间（这与控制论中的状态空间或者物理和微分方程中的空间相一致）。

第 13 章：微分模型

本章引入大量的涉及微分约束的模型，这是第Ⅳ部分的基础，包括来自车轮滚动的约束、机械系统的动力学约束等。

第 14 章：微分约束条件下基于采样的规划

本章提出解决第 13 章模型条件下的规划问题的算法。一些算法是第 5 章方法的扩展。所有方法是基于采样的方法，因为在微分约束条件下组合技术几乎无所作为。

第 15 章：系统理论与分析技术

本章主要介绍在控制论中发展起来的概念和工具。作为对第 14 章的算法的补充，它们提供了对微分约束条件下规划算法的发展的重要认识。

第2章 离散规划

本章将介绍基本概念，这里考虑的规划问题是最简单的，因为在大多数情况下状态空间是有限的。如果系统不是有限的，那么也至少是可数无限的，因此不需要几何模型或微分方程来描述离散规划问题，并且不考虑不确定性，这样可以避免使问题复杂化，所有的模型都是完全可知的和可预测的。

本章包含3个主要部分：第2.1节和第2.2节为可行规划定义搜索方法，这两节只关心目标状态的到达，这些搜索方法将在后续章节中用到；第2.3节论述最优规划问题，《最优化原理》或《动态规划原理》[86]提供的关键技术能够在许多规划算法中减少计算量，动态规划中的值迭代方法是第2.3节的重点，这一节也将讨论Dijkstra算法与值迭代方法的关系；第2.4节和第2.5节描述了规划的逻辑表示以及利用这些表示使问题求解更容易的方法。

尽管本章讨论的是一种形式的规划，但是它还包括“问题求解”。在人工智能的研究历史上，很难描述“问题求解”[738]与“规划”之间的区别。Russell和Norvig合著的、应用广泛的教科书[842]对人工智能领域进行了具有代表性的调查研究。书中6个主要部分中的两个就是“问题求解”和“规划”；但是，它们的定义十分相似。问题求解部分是从论述“问题求解是通过发现导致期望状态的行动序列来决定做什么”开始的，规划部分是从“提出能够完成目标的行为序列的任务被称为规划”开始的，并且STRIPS系统[339]被公认为一种基本的规划算法，“PS”就代表“问题求解器”，因此问题求解与规划就是同义词，但“规划”可能有将来时间的含义，而“问题求解”则似乎更具有一般意义。问题求解这个任务可能像从一个犯罪现场提取线索并将疑犯的行为拼接起来。称之为“规划”似乎有点奇怪，因为它发生在过去。

因为很难清楚地区分“问题求解”和“规划”之间的差别，所以我们简单地把它们统称为规划。这样有助于保持本书的主题。当然读者也应该注意到：一些概念可以应用于比规划的含义更广泛的问题中。

2.1 离散可行规划简介

2.1.1 问题表述

离散可行规划模型的定义采用状态空间模型，这些模型将在本书中不断出现。它们是这里介绍的模型的自然延伸。世界中每个明确的情形称为状态，用 x 表示，所有可能的状态的集合称为状态空间 X。对于离散规划，重要的是该集合是可数的；在大多数情况下它是有限的。在给定应用中，应该仔细定义状态空间以使无关信息不会成为状态（例如涉及在法国控制一个机器人的规划问题不应该包括关于在中国的电灯泡是否亮着这样的信息）。

融入无关信息会使本来很容易求解的问题变得难以求解，另一方面，状态空间 X 必须足够大，以包含所有与解决任务有关的信息。

通过应用规划器所选择的行动，世界能够发生变换。每个行动 u 作用于当前状态 x 后就产生了新的状态 x'，它们的关系由状态转移函数 f 确定。用 f 能够方便地表示状态转移方程：

$$x'=f(x,u) \tag{2.1}$$

用 $U(x)$表示每个状态 x 的行动空间，它表示能够从 x 所应用的全部行动的集合。对于不同的 x, $x'\in X$, $U(x)$与 $U(x')$不一定是没有交集的；同一个行动可以应用于多个状态，因此很容易定义所有可能行动的集合 U：

$$\mathrm{U}=\bigcup_{x\in X}U(x) \tag{2.2}$$

作为规划问题的一部分，定义目标状态的集合 $X_G\subset X$。规划算法的任务就是发现一个有限行动序列将初始状态 x_I 变换到 x_G 中的某状态。

表述 2.1（离散可行规划）

1. 一个非空状态空间 X，它是状态的有限或可数无限集合。
2. 对于每个状态 $x\in X$，有一个有限行动空间 $U(x)$。
3. 对于每个 $x\in X$ 和 $u\in U(x)$，一个状态转移函数 f 产生状态 $f(x,u)\in X$。状态转移方程 $x'=f(x,u)$从 f 产生而来。
4. 一个初始状态 $x_I\subset X$。
5. 一个目标集合 $X_G\subset X$。

通常很容易将表述 2.1 表示成一个有向**状态转移图**（state transition graph）。顶点的集合就是状态空间 X。在图中存在从 $x\in X$ 到 $x'\in X$ 的一条有向边，当且仅当存在一个行动 $u\in U(x)$使 $x'=f(x,u)$。初始状态和目标集合被指定为图中的特殊顶点，这就以图的形式对表述 2.1 进行了表示。

2.1.2 离散规划的实例

例2.1（二维栅格上运动） 假设一个机器人在栅格上运动，每个格点的坐标为整数(i,j)。机器人向 4 个方向（上、下、左、右）中的 1 个方向前进，坐标每次加 1 或者减 1。运动和相应的状态转移图如图 2.1 所示，它可以被想象为在有无限多个格的地板上从一个格步进到另外一个格。

这可以用表述 2.1 来进行表达。令 X 为所有(i,j)形式的整数对，其中 $i,j\in\mathbb{Z}$（$\mathbb{Z}$ 表示所有整数的集合）。令 $U=\{(0,1),(0,-1),(1,0),(-1,0)\}$。令对于所有 $x\in X$，有 $U(x)=U$。状态转移方程为 $f(x,u)=x+u$，其中为了进行加法运算，$x\in X$ 和 $u\in U$ 被当作二维向量。如果 $x=(3,4)$和 $u=(0,1)$，那么 $f(x,u)=(3,5)$。为方便起见，假设初始状态为 $x_I=(0,0)$，可能有许多有趣的目标集合。假设 $X_G=\{(100,100)\}$，就很容易找到一个输入序列来将状态从(0,0)转移到(100,100)。

如果用阴影的格来表示机器人必须避开的障碍物（见图 2.2），那么问题就更有趣了。在这种情况下，任何阴影格的相应顶点和有关的边将被从状态转移图中删除。这将构造出

一个外部边界以形成一个有界区域使 X 变成有限的。这样可以构造非常复杂的迷宫。 ■

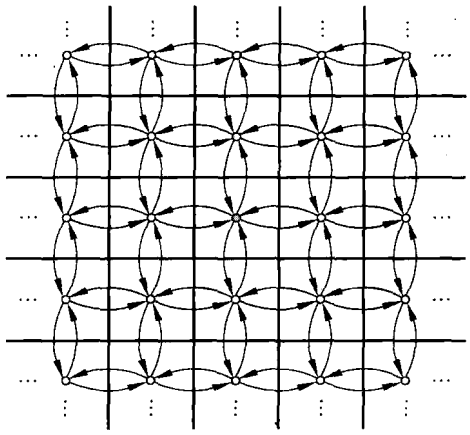


图 2.1 一个实例的状态转移图，该例涉及在一个有无限多个方格的地板上行走的问题

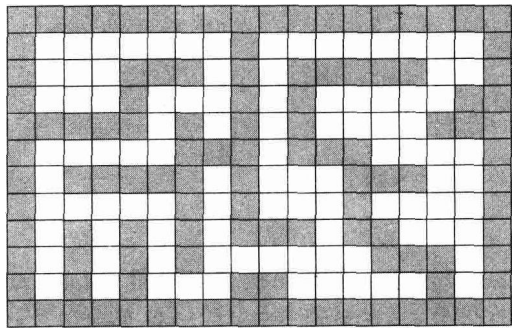


图 2.2 有趣的规划问题，它涉及探索一个用阴影方格构造起来的迷宫

例 2.2（魔方） 许多问题都可以表示成离散规划问题。例如魔方看起来像一个有 3×3×3 个小立方体构成的阵列，它们形成了一个大的立方体，如第 1.2 节中的图 1.1(a)所示。大立方体的每个面涂有 6 种颜色之一。应用一个行动将立方体的一个 3×3 的层旋转 90°。经过多次行动以后，每个面的颜色逐渐变得混乱。状态空间是立方体的位形的集合（与整个立方体的定向无关）。对于每个状态来说，有 12 个可能的行动。对于魔方的某任意位形来说，规划任务是找到一个行动序列使之返回到原来的一个面是一种颜色的位形。 ■

应该注意到，规划问题通常并没有明确表示出整个状态转移图。它是在规划过程中逐步显现出来的。在例 2.1 中，实际上只需要给定很少的信息就可确定一个尺寸为无限的图。如果一个规划问题作为算法的输入，那么当进行一个复杂度分析时必须注意编码问题。对于 X 是无限的这样的一个问题，输入长度必须仍然是有限的。对于某些问题，也许能够简洁地确定一个等价于表述 2.1 的模型。现在已经有许多不同的表示逻辑，这些表示问题在过去的几十年中一直都是人工智能中许多研究的基础，参见第 2.4 节和文献[384]。从某种意义上来看，这些表示可以看作是输入压缩方案。

有计算机工程经验的读者可能意识到，当 X 是有限时，表述 2.1 等同于一个有限状态机或者 Mealy/Moore 机的定义。将这两个模型联系起来，行动可以被阐述为状态机的输入，状态机的输出只是报告它的状态。因此可行规划问题（如果 X 是有限的）可以被阐述为确

定是否存在一个输入序列使得一个有限状态机最终报告一个所需要的输出。从规划的观点来看，假设规划算法有一个完整的机器转移指标，并且能够在任意时刻读出它目前的状态。

有理论计算机科学经验的读者可能观察到，这与确定性的有限自动机（deterministic finite automaton，DFA）有相似的联系，DFA 是一种特殊类型的有限状态机，它读入一个输入字符串并决定是接受（accept）还是拒绝（reject）它。输入字符串只是输入的一个有限序列，与有限状态机中的输入有着相同的意义。DFA 的定义包含可接受的状态集合，该集合在规划中被重新命名为目标集合。这使得可行规划问题（如果 X 是有限的）等价于确定是否存在一个被给定的 DFA 所接受的输入字符串。通常，语言（language）与 DFA 有关，它是 DFA 接受的所有字符串的集合。DFA 在计算理论中是非常重要的，因为它们的语言对应于规则的表达式。规划问题等同于确定空的语言是否与 DFA 有关。

因此，有多种方法来表示和阐述离散可行规划问题，这些方法有时能够对问题进行非常简洁的、隐式的编码。这个问题将在第 2.4 节再次进行讨论。在此之前，第 2.2 节将介绍基本规划算法，第 2.3 节将介绍离散最优规划。

2.2 可行规划的搜索

本节讲述的方法只是图搜索算法，但读者也应该知道，状态转移图是应用行动后逐渐显现出来的，而不是预先全部指定的。所以本节的内容可以认为是从规划的角度访问图搜索算法。搜索算法必须是系统的（systematic）。如果图是有限的，这意味着算法将访问每个可达状态，它在有限时间内应正确地报告是否存在一个解。“系统的”是指算法应该跟踪已经访问的状态；否则搜索可能在同一状态永远循环进行。通过保证不发生任何多余的探索过程，那么就足以保证搜索过程是系统的。

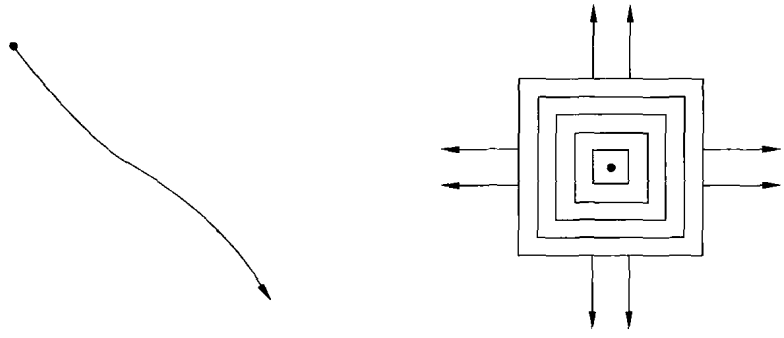
如果图是无限的，我们愿意容忍关于“系统的”的一个较弱的定义。如果解存在，那么搜索算法必须在有限时间内报告结果；如果解不存在，搜索算法永远进行下去是可接受的。“系统的”这个要求是这样实现的，在极限情况下，当搜索迭代次数趋近无穷时，保证图中每个可达顶点都被探索过。因为假设顶点是可数的，这一定总是可能的。

作为这种要求的一个例子，考虑例 2.1 中在无限多个地板方格上的情况。如果搜索算法仅沿一个方向进行探索（见图 2.3(a)），那么大部分空间并没有被覆盖到，尽管也没有状态被重复访问。如果搜索过程从波前原点向外进行（见图 2.3(b)），那么它是系统的。实际上必须仔细分析每个搜索算法。虽然搜索算法能够向多个方向进行，但仍然可能不是系统的。如果图是有限的，那么任何搜索算法都可以是系统的，只要它能够标记已经访问过的状态，以免无限重复访问相同状态。

2.2.1 一般前向搜索

图 2.4 给出用状态空间表达的搜索算法通用模板。在搜索过程中有 3 种状态。

（1）未访问的：还没有被访问的状态。初始时除初始状态以外，每个状态均未被访问。



(a) (b)

图 2.3 (a) 一些算法过于集中在一个方向上，这妨碍了它在无限图上是系统的；(b) 如果搜索沿波前仔细进行，那么它就会变成系统的；要求“系统的”就是在极限情况下，当迭代次数趋近无穷时能够到达所有可达顶点

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x←Q.GetFirst()
4       if x∈ X_G
5           return SUCCESS
6       forall u∈ U(x)
7           x'←f(x,u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```

图 2.4 前向搜索的一个通用模板

(2) 不活动的：已经访问的状态以及对该状态来说下一个可能的状态都已经被访问过。x 的下一个状态是 x'，对于它来说存在 $u \in U(x)$ 使 $x' = f(x,u)$。在某种意义上，这些状态是不活动的，因为它们对于搜索来说不会再有贡献；不会对发现可行规划的更新的信息有任何帮助。第 2.3.3 节讨论了一种变种情况，在获得最优规划过程中不活动状态能够再次变成活动的。

(3) 活动的：已经遇见的状态和一些还没有被访问的相邻状态。它们被认为是活动的，在初始时刻只有初始状态 x_I 是活动的。

活动状态的集合保存在优先级队列 Q 中，优先级由优先级函数确定。各种搜索算法之间的明显区别在于采用特定的函数来对 Q 进行排序。后面还会论述一些变化形式，但是现在暂时选择其中之一。因此现在假设队列 Q 是一个普通的 FIFO（First-In First-Out，先进先出）队列；当调用 *Q.GetFirst*()时，将会选择已经等待最长时间的那个状态。一般搜索算法的其余部分十分简单。初始时 Q 包含初始状态 x_I。然后执行 **while** 循环，仅当 Q 为空时跳出循环。这种情况发生在已经探索整个图而没有发现任何目标状态时，即导致 FAILURE（除非状态空间 X 的可达部分是无限的，在这种情况下算法将永不停止）。在每次 **while** 迭

代中，Q 中最靠前的元素 x 将被删除。如果 x 在 X_G 中，那么就报告 SUCCESS，并终止搜索过程，否则算法会尝试每个可能的行动 $u \in U(x)$。对下一个状态 $x' = f(x,u)$ 来说，必须确定是不是第一次访问 x'。如果它还没有被访问过，那么则将它插入队列 Q，否则就不需要考虑它，因为它要么是不活动的，要么已经在队列 Q 中。

图 2.4 所描述的算法删除了几个细节，这些细节在实际应用中通常非常重要。例如第 4 行（Line 4）用于确定 $x \in X_G$ 所进行的测试是否有效。这取决于状态空间的大小以及对 x 和 X_G 的表示方法。在这里并未指定特定的方法，因为还未给出表示。

一个重要的细节是现有的算法仅指出解是否存在，而并没有产生一个规划，规划是指完成目标所需要的行动序列。通过在第 7 行后插入一行将 x' 与前一个状态 x 联系起来，就可以解决上述问题。如果每次都进行这个运算，那么就很容易跟踪从最终状态到初始状态的指针线索来得到规划。为方便起见，除了跟踪从 x' 到 x 的指针线索以外，还需要保存已经应用过的行动。

第 8 行与第 9 行从表面上看是很简单的，但是如何判断 x' 是否被访问过？对一些问题，状态转移图的形状可能是一棵树，这意味着没有重复的状态。尽管这并不经常发生，但如果它发生，那就太好了，因为没有必要检查状态是否被访问过。如果 X 中的所有状态都在栅格上，那么可以制作一个表格，在固定时间内查询该表来判断某种状态是否被访问过。但在通常情况下很难这样做，因为状态 x' 必须与队列中所有其他状态进行比较，也必须与已经访问过的状态进行比较。如果每个状态很长，那么这样做所花费的代价是很大的。一个简练的数据结构能够大大减小这种代价，但是在实际情况下，计算时间依然很长。另外一种方法是允许有重复状态，但这又会导致计算量的增加。即使图很小，搜索算法耗费的时间也会与状态转移图的大小呈指数关系，或者即使图是有限的，搜索过程也可能根本不会终止。

一个细节是，一些搜索算法要求计算与每个状态有关的代价。如果一些状态被重复访问，那么代价必须更新，如果搜索算法需要（第 12 行进行了这种更新）。这些代价用来以某种方式对优先级队列进行排序，或者它们能够产生实现算法的规划。如果不保存指针，则应将回到初始状态的最优代价与每个状态一起保存。这个代价足以确定导致任何已访问状态的行动序列。从一个访问过的状态开始，通过以在每步尽可能快地减小代价的方式向后遍历状态转移图，这样就可以得到返回初始状态 x_I 的路径。为此，代价必须具有单调性，这可以通过 Dijkstra 算法和 A^* 搜索来得到，这部分内容将在第 2.2.2 节介绍。在一般情况下，代价必须形成导航函数并融入离散规划，在第 8.2.2 节中，我们将导航函数当作反馈来考虑。

2.2.2 特殊前向搜索

本节提出几种搜索算法，其中每个都可构造出一棵搜索树。它们都是图 2.4 所述算法的特殊情况，只是给 Q 定义了不同的排序函数。大多数算法都是经典图搜索算法[245]。

广度优先 第 2.2.1 节给出的算法将 Q 指定为一个先入先出（FIFO）队列，即采用先来先用的原则选择状态。这引起搜索边界一致（uniform）增长，所以称之为广度优先搜索（breadth-first search），包含 k 步的所有规划在进行有 $k+1$ 步的规划之前都已经全部用完。因此广度优先保证发现的第一个解使用最少步数。在检测状态是否重复访问时，在第 12

行中不做任何工作。因为搜索过程在一系列波前中进行，所以广度优先搜索是系统的。实际上如果它不跟踪重复状态，它甚至将一直保持是系统的（但是它将耗费时间来考虑无关的循环）。

广度优先搜索的渐近运行时间是 $O(|V|+|E|)$，其中$|V|$和$|E|$分别是状态转移图中顶点和边的数量（图通常不是输入；例如输入可能是魔方的规则）。这意味着诸如确定一个状态是否已经访问过等所有这样的基本运算可在固定时间内完成。实际上这些运算通常要求更多的时间，并且必须作为算法复杂性的一部分而进行考虑。运行时间能以其他形式进行表达。请注意，$|V|=|X|$是状态的数量。如果在每个状态中，相同的行动 U 是可行的，那么$|E|=|U||X|$。如果行动集合 $U(x_1)$和 $U(x_2)$对于任何 $x_1, x_2 \in X$ 是两两不相交的，那么$|E|=|U|$。

深度优先 将 Q 作为一个栈（后进先出，LIFO），那么对状态转移图的探索将是非常迅速的，这不同于广度优先搜索的一致扩展。这被称为深度优先搜索（depth-first search），因为搜索快速深入图中。特性是，尽早地优先选择更长的规划。尽管这种积极迅速的行为似乎是我们想要的，但是应该注意到选择更长的规划具有随意性。行动正好以它们定义的顺序应用于 forall 循环。如果一个状态被再次访问，那么第 12 行将不做任何工作。对任何有限 X，深度优先搜索都是系统的，对于无限 X 则不是这样，因为它的行为将如图 2.3(a)所示那样。搜索过程很容易集中于一个方向，当迭代过程趋近无穷时，搜索过程可能错过大部分的搜索空间。深度优先搜索的运行时间也是 $O(|V|+|E|)$。

Dijkstra 算法 到目前为止，没有理由认为一种搜索过程比其他过程更优越。第 2.3 节将最优离散规划用数学的形式加以描述，并提出几种发现最优规划的算法。在此之前，我们先提出一种寻找最优规划的系统算法，因为它对寻找可行规划也非常有用。它是著名的 Dijkstra 算法，它用于寻找图中单源最短路径[275]，这是动态规划的一种特殊形式。第 2.3 节将介绍贯穿本书的更一般的动态规划计算。

假设在离散规划问题的图论表示中，每条边 $e \in E$ 都有一个相应的非负代价 $l(e)$，它是应用行动所付出的代价。代价 $l(e)$可以用状态空间的形式表示为 $l(x, u)$，表示从状态 x 应用行动所产生的代价 $l(x, u)$。规划的总代价只是从初始状态到达目标状态的路径上边代价（edge costs）的总和。

优先级队列 Q 按照被称为已付代价（cost-to-come）的函数 $C: X \rightarrow [0, \infty]$进行排序。对每个状态 x，值 $C^*(x)$被称为从初始状态 x_I 产生的最优已付代价[1]。在从 x_I 到 x 所有可能路径上对边的代价 $l(e)$求和，并取产生最小累积代价的路径，这样就可以得到最优代价。如果代价不是最优的，那么可被表示为 $C(x)$。

可在如图 2.4 所示的搜索算法执行过程中递增地计算已付代价。初始时 $C^*(x_I)=0$。每次生成状态 x'后，计算代价指标 $C(x') = C^*(x) + l(e)$，其中 e 是从 x 到 x'的边（等价地，也可以记为 $C(x') = C^*(x) + l(x,u)$）。这里 $C(x')$表示目前已知的最好已付代价。但是它并没有被记为 C^*，因为不知道是否已经最优地到达 x'。基于此，需要改进第 12 行，如果 x'已经在队列 Q 中，那么新发现的通向 x'的路径有可能更有效。如果是这样的话，那么必须降低 x'的已付代价值 $C(x')$，Q 必须被重新排序。

那么 $C(x)$什么时候变成 $C^*(x)$呢？一旦用 $Q.GetFirst()$将 x 从 Q 中删除，那么该状态将

1 如同关于优化的文献那样，我们用*来表示最优的。

变成不活动状态，并且意味着不能以更低的代价到达 x。这可以用归纳法加以证明。对于初始状态，$C^*(x_I)$是已知的，这将作为进行归纳的基础。假设每一个不活动状态的最优已付代价都已经确定。这意味着它们的已付代价值不再改变。对于队列中的第一个元素 x，值必须是最优的，因为任何具有更低总代价的路径将不得不经过 Q 中另一个状态，但是这些状态已经有更高的代价。在产生 $C(x)$的过程中已经考虑了所有只经过不活动状态的路径。一旦探索了所有离开 x 的边，那么可以认定 x 是不活动的，归纳过程继续进行。这并不足以证明最优性；更多论据将出现在第 2.3.3 节和文献[245]中。运行时间为 $O(|V|\lg|V|+|E|)$，其中$|V|$和$|E|$分别是离散规划问题的图论表示中的顶点和边的数量。这假设优先级队列用 Fibonacci 堆来实现，并且诸如确定一个状态是否已经访问过等所有其他运算均将在固定时间内完成。如果采用其他数据结构来实现优先级队列，那么可能需要更长的运行时间。

A^*算法 A^*搜索算法是 Dijkstra 算法的扩展，通过对从给定状态到目标状态的代价进行启发式估计，它试图减少所需探索状态的数量。令 $C(x)$表示从 x_I 到 x 的已付代价，$G(x)$表示从 x 到 x_G 中某状态的尚需代价（come-to-go）。那么很容易用动态规划法递增地计算 $C^*(x)$，但是无法提前知道真正的最优尚需代价 G^*。幸运的是在一些应用中有可能构造一个合理的低估代价。作为一个典型的低估的例子，考虑如图 2.2 所示的迷宫中的规划。假设代价是规划中的总步数。如果一个状态的坐标为(i, j)，另一个状态为(i', j')，那么 $|i'-i|+|j'-j|$就是一个低估值，因为它是不考虑障碍物而直接规划的长度。一旦包含障碍物，那么当机器人试图到达障碍物附近时代价就会增加（也许甚至是不可能的）。当然 0 也可以作为一个低估值，但那样不会给算法提供任何有用的信息。我们的目的是计算一个尽可能接近最优尚需代价值的估计值，并且还保证不大于最优尚需代价值。可用 $\hat{G}^*(x)$ 表示该估计值。

A^*算法的运行方式与 Dijkstra 算法几乎相同，只是用于对 Q 进行排序的函数不同。在 A^*算法中，采用 $C^*(x')+\hat{G}^*(x)$，这意味着用从 x_I 到 X_G 的最优代价的估计值来对优先级队列进行排序。对所有 $x \in X$，如果 $\hat{G}^*(x)$ 是真正最优尚需代价的一个低估值，那么 A^*算法保证能够发现最优规划[339,780]。当 $\hat{G}^*(x)$ 接近 G^*时，与 Dijkstra 算法比较而言，这种算法搜索较少的顶点。这似乎很有利，但是在一些问题中，很难或者根本不可能发现一个既能有效地计算，又可提供良好搜索引导的启发式方法。注意到对于所有 $x \in X$，如果 $\hat{G}^*(x)=0$，那么 A^*算法将退化为 Dijkstra 算法。在任何情况下，搜索过程总是系统的。

最好优先 对于最好优先搜索（best-first search），优先级队列按照最优尚需代价的估计值进行排序。这种方式得到的解不一定是最优的；因此估计值是否超过真正的最优尚需代价并不重要，而这对于保持 A^*搜索的最优性却是非常重要的。尽管没有得到最优解，但在一些情况下，探索较少的顶点能加快运算速度。但这种情况并不是一定能够发生。最好优先搜索在最坏情况下的性能比 A^*搜索和动态规划在最坏情况下的性能差。该算法经常非常“贪婪”，因为它偏爱那些在搜索中“看起来好的”状态，这种“贪婪”有时是要付出代价的。图 2.5 显示了一个人为的例子，其中涉及在三维世界中花费较少步数的规划问题。对于任何特定的步数 k，容易构造一个螺旋，与 Dijkstra 算法比较而言它至少需要 k 步。需要注意的是，最好优先搜索过程并不是系统的。

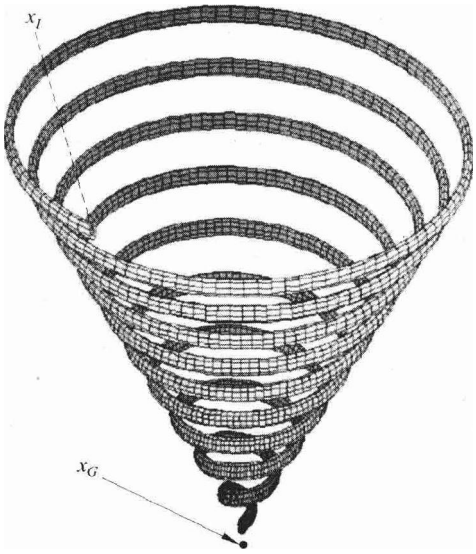


图 2.5 这是最好优先搜索的一个令人感到棘手的例子；假设想要到达螺线管正下方的一个状态；如果初始状态从管子的顶部开口内开始，那么搜索将沿着螺线管进行，而不是离开管子直接朝目标前进

迭代深化 当搜索树有大量的分支时迭代深化方法通常是更可取的（例如在下一层比当前层有更多的顶点）。如果每个状态有许多行动并且只有一些状态被重复访问，这时就会发生这种情况。它采用深度优先搜索的思路，并且寻找所有与 x_I 的距离为 i 或者小于 i 的状态。如果没有发现目标，那么则放弃以前的工作，采用深度优先寻找所有与 x_I 的距离为 i+1 或者小于 i+1 的状态。这通常从 i=1 开始迭代，搜索过程不断进行直到发现目标。迭代深化方法可以被认为是将深度优先搜索转换为一种系统搜索的方法。放弃以前迭代工作的目的是到达距离 i+1 的状态的数量大大超过了到达距离 i 的状态的数量（例如 10 倍）。因此一旦到达 i+1 级，那么以前所有迭代的代价是可以忽略的。

对许多问题而言，迭代深化方法具有比广度优先搜索更好的最坏情况性能，并且还会减少所需要的空间，因为广度优先搜索中的队列通常比深度优先搜索的队列大得多。如果最近的目标状态与 x_I 的距离为 i，那么在最坏情况下广度优先搜索在搜索过程终止之前可能到达所有距离为 i+1 的状态。每次到达与 x_I 的距离为 i 的状态 $x \notin X_G$ 时这种情况就会发生，因为所有一步能够到达的新状态均被放置在队列 Q 中。A^* 的思路与迭代深化方法进行组合可以产生 IDA^* 算法，其中 i 被 $C^*(x')+\hat{G}^*(x')$ 代替。在 IDA^* 算法的每次迭代中，允许的总代价将逐渐增大[780]。

2.2.3 其他搜索方案

本节涉及两个一般意义的搜索算法，第一个只是图 2.4 中树搜索算法的“后向”版本。第二个是使两棵搜索树进行双向生长的方法，一棵树从初始状态开始生长，另一棵从目标状态生长。

后向搜索 第 2.2.2 节中的任何前向搜索算法均可产生其后向版本。例如 Dijkstra 算法

的后向版本可从 x_G 开始。为了创建后向搜索算法，这里假设仅有一个目标状态 x_G。对于一些规划问题，从 x_I 开始的分支可能很多。在这种情况下，从目标状态开始进行搜索也许更有效，这样就会遇到从 x_G 开始直到初始状态的后向工作。这种方法的一个通用模板由图 2.6 给出。对于前向搜索，从 $x \in X$ 应用行动 $u \in U(x)$ 得到一个新的状态 $x' = f(x,u)$。对于后向搜索，频繁计算的是确定某 x'、前续状态 $x \in X$ 以及行动 $u \in U(x)$，以使 $x' = f(x,u)$。在第一行通过将所有 $x_G \in X_G$ 插入队列 Q，并将其标记为已访问的，通过扩展图 2.6 中的模板可以处理一个目标区 X_G。

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x←Q.GetFirst()
4       if x∈ X_G
5           return SUCCESS
6       forall u∈U(x)
7           x'←f(x,u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```

图 2.6 后向搜索的一个通用模板

对大多数问题，人们可能更愿意预先计算“后向”搜索算法的状态转移函数 f。这里将构造 f 后向版本中的一些便捷符号。令 $U^{-1} = \{(x,u) \in X \times U \mid x \in X, u \in U(x)\}$，它表示所有状态-行动对的集合，它们是 f 的定义域。假设从给定的状态 $x' \in X$，用 f 将所有 $(x,u) \in U^{-1}$ 的集合映射到 x'。这可以认为是一个反向行动空间，对任何 $x' \in X$，该空间被定义为：

$$U^{-1}(x') = \{(x,u) \in U^{-1} \mid x' \in f(x,u)\} \tag{2.3}$$

为方便起见，用 u^{-1} 表示属于某 $U^{-1}(x')$ 的状态-行动对 (x,u)。对于任何 $u^{-1} \in U^{-1}(x')$，都存在唯一 $x \in X$。因此用 f^{-1} 表示从 x' 和 $u^{-1} \in U^{-1}(x')$ 产生 x 的后向状态转移函数。这定义了一个后向状态转移方程 $x = f^{-1}(x', u^{-1})$，它看起来与前向版本 $x' = f(x,u)$ 非常相似。

采用状态转移图很容易对 f^{-1} 进行阐述：将每条边反向。在反向图中采用后向搜索进行规划与在原始图中采用前向搜索进行规划是等价的。将所有边反向后得到的后向状态转移函数是 f 的变形。每个 u^{-1} 是反向以后的边。因为是关于第 2.2.1 节前向搜索的对称形式，所以第 2.2.2 节中的任何搜索算法的变形都能够适应图 2.6 的模板，只要提前定义了 f^{-1}。

双向搜索 既然已经介绍了前向与后向搜索方法，那么接下来就应介绍双向搜索。图 2.7 给出的通用搜索模板可以被认为是图 2.4 和图 2.6 中两种搜索的组合。一棵树从初始状态开始生长，而另一棵树从目标状态开始生长（仍然假设 X_G 是单元素集合 $\{x_G\}$）。当两棵树相遇时搜索成功终止。当其中一个优先级队列耗尽时搜索就失败了。对一些问题，双向搜索能够显著减少需要进行探测的数量。有双向搜索的 Dijkstra 算法和 A^* 算法，它们可产生最优解。对于最好优先搜索和其他种类的搜索，保证两棵树快速相遇是很有挑战性的。

它们可能非常接近但不会相遇。额外的启发式方法在一些情况下有助于将一棵树引向另外一棵树，甚至可以将这种框架扩展到任何数量的搜索树。一些应用希望如此，但是树之间的连接是相当复杂和昂贵的。

```
BIDIRECTIONAL_SEARCH
1   Q_I.Insert(x_I) and mark x_I as visited
2   Q_G.Insert(x_G) and mark x_G as visited
3   while Q_I not empty and Q_G not empty do
4       if Q_I not empty
5           x←Q_I.GetFirst()
6           if x=x_G or x∈Q_G
7               return SUCCESS
8           forall u∈ U(x)
9               x'←f(x,u)
10              if x' not visited
11                  Mark x' as visited
12                  Q_I.Insert(x')
13              else
14                  Resolve duplicate x'
15      if Q_G not empty
16          x'←Q_G.GetFirst()
17          if x'=x_I or x'∈ Q_I
18              return SUCCESS
19          forall u^-1∈ U^-1(x')
20              x ←f^-1(x',u^-1)
21              if x not visited
22                  Mark x as visited
23                  Q_G.Insert(x)
24              else
25                  Resolve duplicate x
26  return FAILURE
```

图 2.7 双向搜索的一个通用模板

2.2.4 搜索方法的统一描述

很容易将所有搜索方法的行为以几个基本步骤的形式概括起来。这些步骤的变化形式将在以后更加复杂的规划问题中出现。例如第 5.4 节中一族基于采样的运动规划算法就是对这里提出的步骤的扩展。这种情况是从离散状态空间扩展到连续状态空间（称为位形空间）。每个方法递增地构造一个搜索图$G(V,E)$，它是到目前已经探索过的状态转移图的子图。

本节介绍的所有规划方法均采用同一个基本模板。

1. **初始化**：用 E 为空、V 包含一些开始状态来初始化搜索图$G(V,E)$。对于前向搜索 $V=\{x_I\}$；对于后向搜索 $V=\{x_G\}$；如果采用双向搜索，那么 $V=\{x_I, x_G\}$。也可以在搜索过程中生长更多的树。在这种情况下 V 初始化为更多的状态。可递增地生长搜索图来显现越来越多的状态转移图。

2. **选择顶点**：为扩展而选择一个顶点 $n_{cur} \in V$，这通常通过保持优先级队列来完成。用 x_{cur} 表示与 n_{cur} 有关的状态。

3. **应用行动**：在前向或后向中，得到一个新状态 x_{new}。对于 $u \in U(x)$（前向），$x_{new}=f(x,u)$；对于 $u \in U(x_{new})$（后向），$x=f(x_{new},u)$。

4. **向图中插入一条有向边**：如果算法特有的测试得以通过，那么对于前向搜索情况就会产生一条从 x 到 x_{new} 的边，或者对于后向搜索情况就会产生一条从 x_{new} 到 x 的边。如果 x_{new} 不在 V 中，那么则将其插入 V。[1]

5. **检查解**：确定 G 是否编码了一条从 x_I 到 x_G 的路径。如果只有一棵单独的搜索树，那么这个过程很容易。如果有两棵或两棵以上的搜索树，那么该步骤的花费可能很大。

6. **返回步骤 2**：迭代继续进行，除非得到规划解或者终止条件得到满足，在这种情况下算法报告失败。

注意到在这个概括中，需要几次迭代来生成一个用以前的公式表达的迭代。图 2.4 所示的前向搜索算法为 Q 的第一个元素尝试所有行动。如果有 k 个行动，则对应于上述模板中的 k 步迭代。

2.3 离散最优规划

本节将表述 2.1 扩展到能够定义最优规划问题。假设我们希望得到最优化诸如时间、距离或能量消耗等指标的解，而不是仅满足于得到产生目标集合的任何行动序列。可以进行三个方面的扩展：（1）用一个阶段索引来指示当前的规划步；（2）引入一个代价泛函，它的行为就像一个出租车的里程表，它指示在规划执行过程中累积的代价；（3）引入一个终止行动，它直观地指示什么时候终止规划和确定总代价。

这涉及三个步骤。首先，第 2.3.1 节介绍了关于寻找固定长度最优路径的问题。这种方法被称为值迭代，它在状态空间中迭代地计算最优尚需代价函数。尽管这并不十分有用，但它比可变长度规划更容易理解。一旦理解了该节的概念，那么很容易扩展到可变长度规划，这将在第 2.3.2 节中介绍。第 2.3.3 节说明值迭代与第 2.2.1 节介绍的 Dijkstra 算法之间的密切关系。

几乎所有的优化问题都可以用最小化或者最大化来定义性能指标。如果代价是一种能量或花费，那么最小化似乎才是有意义的，这种情况在机器人学和控制论中非常典型；如果代价是一种收益，就像在投资规划或人工智能中那样，那么人们就希望最大化。尽管这个问题贯串本书，但我们将选择最小化。如果希望最大化，那么将代价乘以–1，并将求最小值用求最大值来代替就足够了。

我们将在稍后介绍固定长度最优规划的表述，但首先需要引入一些新的符号，π_K 表示 K 步规划，它是 K 个行动的序列 $(u_1,u_1,\cdots,u_K)$，如果给定 π_K 和 x_I，那么用状态转移函数 f 可以得到状态序列 $(x_1,x_1,\cdots,x_{K+1})$。初始时 $x_1=x_I$，每个后续状态可用 $x_{k+1}=f(x_k,u_k)$ 求得。

现在给出模型，给表述 2.1 增加的最重要的一项是代价泛函 L。

1 在许多变化中，可以加入顶点，而不需要相应的边，这将在多树方法中创建另外一棵树。

表述 2.2（离散固定长度最优规划）

1. 保留表述 2.1 中所有组成部分：X、$U(x)$、f、x_I和 x_G，除了这里假设 X 是有限的以外（一些算法很容易扩展到 X 是可数无限的情况，但这里不予以考虑）。

2. 阶段数 K 是一个规划的准确长度（行动 $u_1, u_2, \cdots, u_K$ 的数量）。还可以得到阶段索引。例如 x_{k+1} 表示应用 u_k 以后所得到的状态。

3. L 表示 K 步规划 π_K 的代价（损失）泛函。这意味着在 L 的表达式中会有行动序列 $(u_1, u_2, \cdots, u_K)$ 和状态序列 $(x_1, x_2, \cdots, x_{K+1})$。为方便起见，用 F 表示最后一个阶段 $F=K+1$。代价泛函是：

$$L(\pi_K) = \sum_{k=1}^{K} l(x_k, u_k) + l_F(x_F) \tag{2.4}$$

对于每个 $x_k \in X$ 和 $u_k \in U(x_k)$，代价项 $l(x_k, u_k)$产生一个实值。最后一项 $l_F(x_F)$在求和之外，它定义为：如果 $x_F \in X_G$，那么 $l_F(x_F)=0$；否则，$l_F(x_F)=\infty$。

关于 l_F有一个重要的说明。如果 L 仅用于计算到达 X_G的规划，那么在式（2.4）中包含 l_F实际上是不必要的。那样对所有其他规划就不用定义 l_F。在这种假设下，稍后提出的算法依然运行良好；但是符号和阐述会变得更加不方便，因为行动空间必须予以限制以确保产生成功的规划。所以应将 L 的定义域扩展到包括所有规划，并且那些没有到达 X_G的规划被罚以无穷大代价以便这样的规划在任何优化步骤中都会被自动排除。在某些方面上，l_F 的作用可能会变得容易混淆，其实这只是将可行性约束转换为直截了当的优化的一个技巧，记住这一点是有用的（$L(\pi_K)=\infty$意味着不可行；$L(\pi_K)<\infty$意味着是可行的，其代价是 $L(\pi_K)$）。

现在的任务是寻找最小化 L 的规划。为了得到像表述 2.1 所描述的，但限于只有 K 步的规划问题，应令 $l(x, u)\equiv 0$。为了得到要求最小化阶段数的规划问题，应令 $l(x, u)\equiv 1$。如果令不同的 $x \in X_G$有不同的 $l_F(x)$，而不是 $l_F(x)=0$，那么有可能存在其他中间目标。这比可行性规划更加具有一般性，因为状态可以取任何值，而不只是 X_G的内部或外部。

2.3.1 最优定长规划

如果要计算表述 2.2 的最优规划，一种基本的方法是生成所有长度为 K 的序列，选择能够生成最好代价的序列，但是这种方法需要的时间为 $O(|U|^K)$，这是不允许的。幸运的是我们知道动态规划原理。我们首先简要介绍将在方程中使用的方法。因为最优规划本身就是最优的。其他部分不会产生比最优规划更小的总代价，否则有悖于规划的最优性。

最优性原则直接导致一种迭代算法——**值迭代**[1]（value iteration），它能够解决一大类最优规划问题，包括变长度规划、随机不确定性、不完美状态测量以及一些其他复杂情况。思路是采用迭代方法在状态空间上计算最优尚需代价（或已付代价）函数。在某些情况下该方法可以演变为 Dijkstra 算法；当然这只能在某些特殊条件下才能发生。值迭代算法将在下一节介绍，第 2.2.3 节将讨论它与 Dijkstra 算法的联系。

1 “值”是指最优尚需代价或者已付代价。因此它也可以被称为尚需代价迭代。

2.3.1.1 后向值迭代

就像搜索方法一样，值迭代也有前向与后向之分。我们首先介绍后向情况。即使更容易从 x_I 向前进行，但从 x_G 后向进行却更简单。然后在引入一些附加符号的基础上再介绍前向情况。

从短的最优规划中产生长的最优规划的关键是在 X 上构造最优尚需代价函数。对于从 1 到 F 中的第 k 步，令 G_k^* 表示在最优规划执行过程中从 k 到 F 的累积代价：

$$G_k^*(x_k)=\min_{u_k,\cdots,u_K}\left\{\sum_{i=k}^{K}l(x_i,u_i)+l_F(x_F)\right\} \tag{2.5}$$

在式（2.5）中，最小化的是代价泛函式（2.4）的最后 $F-k$ 项。对于边界条件 $k=F$，最优尚需代价简化为：

$$G_F^*(x_F)=l_F(x_F) \tag{2.6}$$

直观意义是：因为没有行动，所以立即就能够得到最后阶段（步）的代价。

现在考虑 X 上的一个 K 步算法，每次从 G_{k+1}^* 计算 G_k^*，k 的变化范围从 F 到 1。在第一次迭代时，G_F^* 直接从 l_F 而来。在第二次迭代中，对于每个 $x_K\in X$，G_K^* 的计算如下：

$$G_K^*(x_K)=\min_{u_K}\{l(x_K,u_K)+l_F(x_F)\} \tag{2.7}$$

因为 $l_F=G_F^*$ 以及 $x_F=f(x_K,u_K)$，代入式（2.7）得到：

$$G_K^*(x_K)=\min_{u_K}\{l(x_K,u_K)+G_F^*(f(x_K,u_K))\} \tag{2.8}$$

它对每个 $x_K\in X$ 直接进行计算。这就计算了从 K 到 F=K+1 的所有最优一步规划。

一旦给定 G_{K+1}^*，那么可以用相似的方法计算 G_K^*。仔细研究式（2.5），并通过将求和中的第一项提出去，将 u_k 的最小化与 u_{k+1} 到 u_K 的最小化分离，它可以表示为：

$$G_k^*(x_k)=\min_{u_k}\left\{\min_{u_{k+1},\cdots,u_K}\left\{l(x_k,u_k)+\sum_{i=k+1}^{K}l(x_i,u_i)+l_F(x_F)\right\}\right\} \tag{2.9}$$

第二个最小化不影响 $l(x_k,u_k)$项，所以 $l(x_k,u_k)$可以被提出来，得到：

$$G_k^*(x_k)=\min_{u_k}\left\{l(x_k,u_k)+\min_{u_{k+1},\cdots,u_K}\left\{\sum_{i=k+1}^{K}l(x_i,u_i)+l_F(x_F)\right\}\right\} \tag{2.10}$$

内部的最小化项就是最优尚需代价函数 G_{k+1}^* 的定义。代入上式就产生了递归形式：

$$G_k^*(x_k)=\min_{u_k}\{l(x_k,u_k)+G_{k+1}^*(x_{k+1})\} \tag{2.11}$$

其中 x_{k+1}=$f(x_k,u_k)$。既然式（2.11）的右端仅仅依赖于 x_k、u_k 和 G_{k+1}^*，那么 G_k^* 的计算时间为 $O(|X||U|)$。这种计算被称为值迭代。在每次值迭代中，某些状态得到一个无穷大的值，那是因为它们是不可达的；或者说是不存在从 x_k 到 X_G 的($K-k$)步规划。这意味着没有行动 $u_k\in U(x_k)$将状态 x_k 引导到 $x_{k+1}\in X$，而从 x_{k+1} 出发的($K-k-1$)步规划是存在的并且终止在 X_G。

综上所述，值迭代进行的过程如下：

$$G_F^*\to G_K^*\to G_{K-1}^*\cdots G_k^*\to G_{k-1}^*\cdots G_2^*\to G_1^* \tag{2.12}$$

直到经过 $O(K|X||U|)$时间后最后确定 G_1^*。所产生的 G_1^* 可以用来产生从 x_I 到目标的最优代价 $G_1^*(x_I)$。也能很方便地给出其他初始状态的最优尚需代价。对于在 K 阶段内不能到达 X_G 的状态来说这个代价是无穷的。

最优规划的代价似乎容易计算出来，但是实际规划是如何提取出来的呢？一种可能性

是保存每个阶段、每个状态下满足式（2.11）最小化的行动。非常遗憾，这需要 $O(K|X|)$ 的存储量，如果采用第 2.3.2 节介绍的更具有一般意义的变长度规划所采用的技巧，那么存储量能减少到 $O(|X|)$。

例 2.3（五状态最优规划问题） 图 2.8 显示了一个规划问题的图的表示，规划问题中 $X=\{a, b, c, d, e\}$。假设 $k=4$，$x_I=a$ 且 $X_G=\{d\}$。那么将有四次值迭代，一旦给定最后一个阶段的尚需代价 G_5^*，那么迭代得到 G_4^*、G_3^*、G_2^* 和 G_1^*。

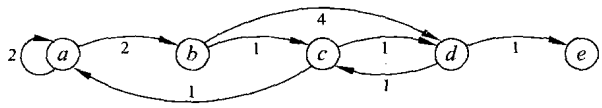


图 2.8 一个五状态例子；每个顶点表示一个状态，每条边表示一个可以应用于状态转移方程来改变状态的输入；边的权重表示 $l(x_k, u_k)$（x_k 是边的起始顶点）

尚需代价函数如图 2.9 所示。图 2.10 和图 2.11 说明了计算问题。对于计算 G_4^*，只有 b 和 c 产生有限值，因为只有它们能够一步到达 d。对于计算 G_3^*，只有值 $G_4^*(b)=4$ 和 $G_4^*(c)=1$ 很重要。只有到达 b 和 c 的路径才可能在 $k=5$ 时到达 d。注意到式（2.11）中的最小化所选择的行动使得在下一个阶段到达一个顶点所付出的总代价最小。 ■

	a	b	c	d	e
G_5^*	∞	∞	∞	0	∞
G_4^*	∞	4	1	∞	∞
G_3^*	6	2	∞	2	∞
G_2^*	4	6	3	∞	∞
G_1^*	6	4	5	4	∞

图 2.9 用后向值迭代计算的最优尚需代价函数

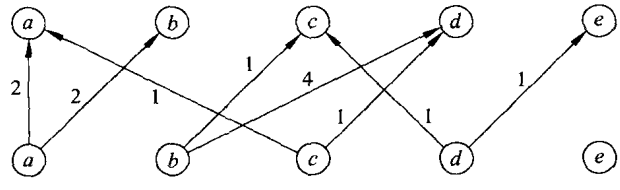


图 2.10 向前一步的可能性；这是通过生成图 2.8 的状态的两个副本来得到的，一个副本对应于目前状态，而另一个副本对应于潜在的下一个状态

2.3.1.2 前向值迭代

第 2.3.1.1 节的思路可以产生用于计算从初始阶段开始的最优已付代价函数的等价方法。鉴于后向值迭代能够寻找从所有初始状态开始的最优规划，那么前向值迭代能用来寻找到达 X 中所有状态的最优规划。在后向情况下，必须固定 X_G；在前向情况下，必须固定 x_I。

再次出现关于保持解的可行性的问题。在向前方向，l_F 的作用并不重要。它可以用在最后一次迭代中，或者在一个没有预先确定 X_G 的问题中删去它。尽管如此，必须迫使前向值迭代所考虑的所有规划从 x_I 开始。我们可以选择采用对行动空间进行约束或者简单地增加一项使不可行规划具有无穷大代价。这里采用后者。

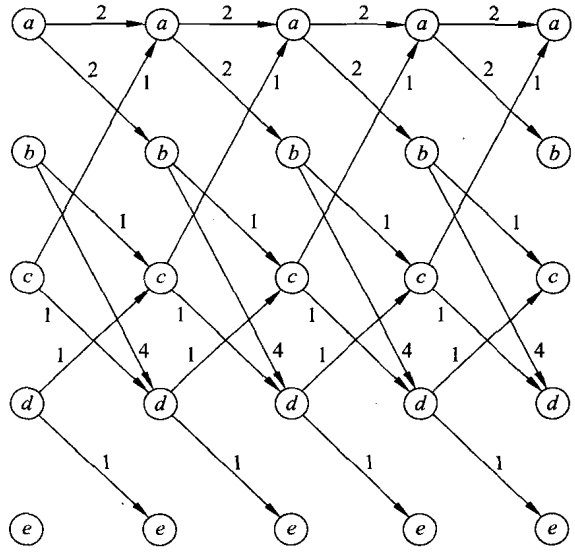


图 2.11 将图 2.10 顺时针旋转 90°并复制 K 次，可以绘制一幅图，它方便地显示了从一个初始状态到一个最终状态的所有路线；计算过程自动选择最优路线

令 C_k^* 表示从阶段 1 到 k 的最优已付代价（optimal cost-to-come），它们是在所有(k–1)步规划上进行优化而来的。为了排除掉不从 x_I 开始的规划，定义 C_1^*：

$$C_1^*(x_1) = l_I(x_1) \tag{2.13}$$

其中 l_I 是一个新函数，它产生 $l_I(x_I)$=0，而对所有 $x \neq x_I$，$l_I(x)=\infty$。因此任何试图从 x_I 以外的状态开始的规划立即产生无穷大代价。

对于中间阶段 $k \in \{2,\cdots,K\}$，下式表示最优已付代价：

$$C_k^*(x_k) = \min_{u_1,\cdots,u_{k-1}} \left\{ l_I(x_1) + \sum_{i=1}^{k-1} l(x_i,u_i) \right\} \tag{2.14}$$

求和表示由行动序列($u_1,\cdots, u_{k-2}$)产生的状态序列 $x_1,\cdots,x_{k-1}$，最后一个状态 x_k 并未被包括在内，因为它的代价项 $l(x_k, u_k)$需要行动 u_k，但此时还没得到 u_k。如果有可能将代价表示成 $l(x_k, u_k)=l_1(x_k)+l_2(u_k)$，那么如果需要，$l_1(x_k)$部分能够包含在已付代价的定义中。这种细节将不予进一步考虑。

像式（2.5）那样，假设式（2.14）中对于每个 $i \in \{1,\cdots,k-1\}$，有 $u_i \in U(x_i)$，应用行动 u_{k-1} 后所产生的状态 x_k 必须与式（2.14）左边的 x_k 相同。在 min 中出现 x_1 可能有点奇怪，但这不是问题。一旦给定 $u_1,\cdots, u_{k-1}$ 以及 x_k，那么完全能够确定状态 x_1。

最后一次前向值迭代是到达最后阶段 F。这种情况下的已付代价是：

$$C_F^*(x_F) = \min_{u_1,\cdots,u_K} \left\{ l_I(x_1) + \sum_{i=1}^{K} l(x_i,u_i) \right\} \tag{2.15}$$

式（2.15）看起来像将 k=1 代入后的式（2.5）；但是这里用 l_I 代替了 l_F。这有过滤掉仅包含从 x_I 开始的那些规划的效果。前向值迭代寻找从 x_I 到达可达最终状态的最优规划。这种行为是后向值迭代的互补形式。对后向情况，X_G 是固定的，目的是寻找从任何初始状态开始的最优规划；对前向值迭代，过程相反。

为了表达动态规划的递归，还有另外一个问题。假设通过推导已知 C_{k-1}^*，我们想要计

算特定 x_k 的 $C_k^*(x_k)$。这意味着必须应用某行动使某 x_{k-1} 到达 x_k。第 2.2.3 节的后向状态转移方程此处仍非常有用。使用阶段索引，这可以表示为 $x_{k-1}=f^{-1}(x_k,u_k^{-1})$。

这种递归是：

$$C_k^*(x_k)=\min_{u_k^{-1}\in U^{-1}(x_k)}\left\{C_{k-1}^*(x_{k-1})+l(x_{k-1},u_{k-1})\right\} \tag{2.16}$$

其中 $x_{k-1}=f^{-1}(x_k,u_k^{-1})$，$u_{k-1}\in U(x_{k-1})$ 是与 $u_k^{-1}\in U^{-1}(x_k)$ 相对应的输入。应用式（2.16），最终的已付代价在 $O(K|X||U|)$时间内通过迭代计算出来，就像后向值迭代方法中计算第一阶段尚需代价那种情况。

例 2.4（前向值迭代） 对于固定规划长度 K=4 的前向值迭代情况，再次回到例 2.3。如图 2.12 所示的已付代价函数可用式（2.16）得到。再次参考图 2.10 和图 2.11 是非常有帮助的。第 1 行对应于立即应用 l_I。在第 2 行，得到 a 和 b 的有限值，它们是从 $x_I=a$ 一步就能够到达的。迭代继续进行直到 k=5，在该阶段就确定了每个状态的最优已付代价。 ■

	a	b	c	d	e
G_1^*	0	∞	∞	∞	∞
G_2^*	2	2	∞	∞	∞
G_3^*	4	4	3	6	∞
G_4^*	4	6	5	4	7
G_5^*	6	6	5	6	5

图 2.12 用后向值迭代计算的最优尚需代价函数

2.3.2 不指定长度的最优规划

固定长度规划的值迭代方法能够推广到允许不同长度的规划。对规划的最大长度不予限制，因此这种情况是对表述 2.1 的推广，因为可以采取任意长的规划。一般情况的模型不需要指定 K，而是引入了一个特殊行动 u_T。

表述 2.3（离散最优规划）

1. 保留表述 2.1 中所有组成部分：X、$U(x)$、f、x_I和 x_G，并且继续使用表述 2.2 中关于阶段的概念。

2. L 表示逐阶段相加的代价泛函，它可以应用于任何 K 步规划 π_K，产生：

$$L(\pi_K)=\sum_{k=1}^{K}l(x_k,u_k)+l_F(x_F) \tag{2.17}$$

与表述 2.2 中的 L 相比较，该表达式并不预先固定 K 值。该值根据规划的长度可以改变，因此 L 的定义域非常大。

3. 每个 $U(x)$包含特殊终止行动 u_T。如果 u_T应用于 u_k，那么该行动将永远重复继续，状态仍然不会变化，并且代价不再累积。因此对所有 $i\geqslant k$，有 $u_i=u_T$，$x_i=x_k$，$l(x_i,x_T)=0$。

终止行动是允许不同长度的规划的关键。它贯串本书。假设值迭代进行到了 K=5，并且对于本问题，存在一个两步规划解(u_1, u_2)使状态从 x_I 到达 X_G。该规划等价于五步规划$(u_1, u_2, u_T, u_T, u_T)$，因为终止行动不改变状态，也不累积代价。所产生的五步规划到达 X_G，并且代价与(u_1, u_2)相同。基于这种扩展，第 2.3.1 节中的前向与后向值迭代方法可以用来优化所有长度为 K 或小于 K 的规划（而不是固定 K）。

接下来的是消除对 K 的依赖性。假设无限地进行后向值迭代。在某点上计算 G_i^*，但没

有原因能够解释为什么过程不能继续在 $G_0^*, G_{-1}^*, \cdots$ 方向上进行。请注意，在后向值迭代方法中并未用 x_I，因此并不关心规划的初始状态。假设 K=16 应用后向值迭代，并且执行到 G_{-8}^*。这考虑了所有长度小于等于 25 的规划。将所有阶段索引增加 9 来移位所有尚需代价函数，这样做是无害的。不是从 G_{-8}^* 到 G_{16}^*，它们可以从 G_1^* 到 G_{25}^*，这样并不影响它们的值。能够进行索引移位的原因是代价不依赖于特定的索引。对值迭代来说重要的是它们向后进行，连续地从一个阶段到另一个阶段。

最终需要进行足够的迭代以使对每个能够到达 X_G 的状态产生一个最优规划。从第 k 步往后，从一次值迭代到下次值迭代的尚需代价值将是平稳的，即对于所有 $i \leqslant k$，$x \in X$，有 $G_{i-1}^*(x) = G_i^*(x)$。一旦满足平稳条件，那么尚需代价函数不再依赖于特定阶段 k。在这种情况下，可以去掉阶段索引，递归变成：

$$G^*(x) = \min_u \left\{ l(x,u) + G^*(f(x,u)) \right\} \tag{2.18}$$

是否存在这样的条件，在该条件下后向值迭代永远执行下去，并且每次迭代产生的一个尚需代价函数的某些值与前面迭代的不同？对于所有 $x \in X$ 以及 $u \in U(x)$，如果 $l(x, u)$是非负的，那么这将不会发生。对于任何固定 K，还存在更长的规划，但这些规划并不能认为是最优规划。对每一个 $x \in X$，要么存在一个规划解以有限代价到达 X_G，要么无解。对于每个存在到达 X_G 的规划的状态，都要考虑最优规划中的阶段数。假设存在从所有能够到达 X_G 的状态开始的最大阶段数。这可以作为在尚需代价变得平稳之前值迭代次数的上界值。任何进一步的迭代所得到的解只会比已经考虑到的解（由于终止行动和阶段移位，一些解可能是等价的）更糟糕。如果 $l(x, u)$包含负值，则会有一些麻烦。如果状态转移图包含使代价为负的循环，那么更可取的是执行一个规划，它永远在循环的周围，因此将总代价减到$-\infty$。因此假设以一种有意义的方式定义代价泛函使负循环不存在。否则最优模型本身就是有缺陷的。只要没有负循环，那么就允许 $l(x, u)$的某些值是负的。在进行值迭代之前，直接检测和报告负循环是很简单的。

因为阶段的索引并不重要，那么令 $k=0$ 是最后阶段的索引，它是后向值迭代的开始。所以 G_0^* 是最后阶段的代价，它是直接从 l_F 得到的。令$-K$ 表示尚需代价值都变得平稳时的阶段索引。在这个阶段，最优尚需代价函数 $G^*: X \to \mathbb{R} \cup \{\infty\}$ 用赋值 $G^* = G_{-K}^*$ 来表达。也就是说，特定阶段索引不再重要。值 $G^*(x)$给出从状态 $x \in X$ 到特定目标状态 X_G 的最优尚需代价。

如果在值迭代中没有保存最优行动，那么用最优尚需代价 G^*可以有效地得到它们。考虑从某些 $x \in X$ 开始，最优的下一个行动是什么呢？这由下式给出：

$$u^* = \arg\min_{u \in U(x)} \left\{ l(x,u) + G^*(f(x,u)) \right\} \tag{2.19}$$

其中 arg min 表示产生最小值的自变量。行动使与式（2.11）非常相似的式（2.19）最小化。仅有的不同在于式（2.19）中去掉了阶段索引，因为尚需代价值不再依赖于它们，并使用 arg min 以得到 u^*。应用 u^*以后，用状态转移方程得到 $x' = f(x, u^*)$，式（2.19）可以再一次应用于 x'。这个过程将继续，直到到达 X_G 中的一个状态。该过程以动态规划递归为基础，因此将得到最优规划。函数 G^*用作一个导引，它将系统最优地从初始状态引导到目标集合。所以它可以被认为是导航函数（navigation function）的特殊情况，导航函数将在第 8.2.2 节介绍。

像固定长度规划一样，翻转值迭代的方向可以得到可变长度规划问题的前向值迭代方法。在这种情况下，后向状态转移方程 f^{-1} 再次得到应用。像式（2.14）一样，用初始代价项 l_I 来代替 l_F。前向值迭代方法从 $k=1$ 开始，然后迭代直到尚需代价变得平稳。终止行动 u_T 保存了在以前迭代中到达一个状态所采用的规划的代价。请注意，此处不要求指定 X_G。得到与 G^* 相对应的值，从该值就能够得到最优行动。当已付代价值变得平稳时，最优已付代价函数 $C^*: X \to \mathbb{R} \cup \{\infty\}$ 可以用 $C^* = C_F^*$ 来表达，其中 F 是算法终止时所到达的最后阶段。值 $C^*(x)$ 给出从 x_I 到 x 的最优规划的代价。对于任何指定的目标 $x_G \in X$，通过如下方程可以得到最优行动序列：

$$\arg\min_{u^{-1} \in U^{-1}} \left\{ C^*(f^{-1}(x,u^{-1})) + l(f^{-1}(x,u^{-1}),u') \right\} \tag{2.20}$$

它是式（2.19）的前向形式。u' 是当应用状态转移函数 f 时产生 x 的 $U(f^{-1}(x,u^{-1}))$ 中的行动。迭代过程从 x_G 向后进行，当到达 x_I 时终止。

例 2.5（可变长度规划的值迭代） 再次用到例 2.3，但是由于有终止行动，因此这次规划的长度不固定。它的效果在图 2.13 中通过重叠具有零代价的新边来刻画。开始时它可能没有选择非终止行动的动机，但是没有终止在状态 $x_G=d$ 的任何行动的代价都是无限的。

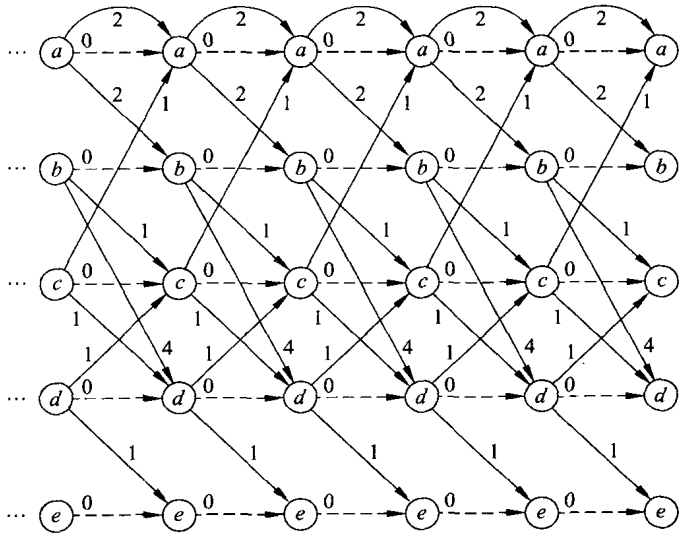


图 2.13 将本图与图 2.11 进行比较，图 2.11 中 K 固定为 4；终止行动的效果用点线来刻画，在遍历时它产生 0 代价；这考虑到了所有有限长度的规划；并且阶段向左无限延伸（后向值迭代情况）

如图 2.14 所示，多次后向值迭代以后，尚需代价值变得平稳了。从此往后，从所有可达状态都可应用终止行动，并且不会进一步累积代价。最后尚需代价函数定义为 G^*。因为从 e 不能到达 d，所以 $G^*(e)=\infty$。

作为一个用式（2.19）来产生最优行动的例子，现在假设从状态 a 开始。下面选择导致 b 的行动，因为总代价 $2+G^*(b)=4$ ∶ $2+G^*(a)=6$ 好（2 来自于行动所产生的代价）。从状态 b，最优行动导致 c，这所产生的总代价为 $1+G^*(c)=1$。相似地，下一个行动导致 $d \in X_G$，这会终止规划。

	a	b	c	d	e
G_0^*	∞	∞	∞	0	∞
G_{-1}^*	∞	4	1	0	∞
G_{-2}^*	6	2	1	0	∞
G_{-3}^*	4	2	1	0	∞
G_{-4}^*	4	2	1	0	∞
G^*	4	2	1	0	∞

图 2.14 用在可变长度规划中所用的后向值迭代来计算的最优尚需代价函数

用前向值迭代，假设 $x_I = b$。得到如图 2.15 所示的已付代价函数。对于从一次迭代到下一次迭代都保持固定的任意有限值，应用终止行动。注意到，最后的值迭代在本例中没有用处。一旦计算了 C_3^*，那么就确定了从 x_I 到每个可能状态的最优已付代价，并且将来的已付代价函数也是相同的。因此最终的已付代价被重新命名为 C^*。 ■

	a	b	c	d	e
G_1^*	∞	0	∞	∞	∞
G_2^*	∞	0	1	4	∞
G_3^*	2	0	1	2	5
G_4^*	2	0	1	2	3
G^*	2	0	1	2	3

图 2.15 用在可变长度规划中所用的前向值迭代来计算的最优已付代价函数

2.3.3 再论 Dijkstra 算法

前面已经介绍了两种不同类型的动态规划。第 2.3.2 节中的值迭代方法涉及在整个状态空间上的重复计算。第 2.2.2 节的 Dijkstra 算法在状态空间中只流动一次，但必须保持状态是活动的。

通过聚焦前向值迭代，如例 2.5 所示，并识别“有趣的”变化发生的地方，这样就可以得到 Dijkstra 算法。请回顾一下，对于 Dijkstra 算法，假设所有代价是非负的。对于任何不可达的状态，它们的值将保持无穷大。它们是未访问的状态。最优已付代价已经变得平稳的那些状态是不活动的。对于其余状态，将得到一个初始代价，但该代价可能被一次或者多次降低，直到得到最优代价。所有这样的状态都在队列 Q 中：它们的代价为有限，但可能不是最优的。

理解了值迭代后，就容易理解为什么动态规划的 Dijkstra 形式能够正确地计算最优解。两种算法中未访问的状态将保持无穷，因为没有规划能够到达它们。与例 2.5 中的前向值迭代进行比较是十分有价值的。从某种意义上来讲，Dijkstra 算法非常像值迭代，除了它可有效地维持尚需代价值能够发生变化的那些状态的集合以外。它可插入第一次到达的状态，将它们的已付代价从无穷大改为一个有限值。像值迭代中的情况一样，值以相同的方式发生改变。在这两种算法结束时，所产生的值对应于平稳的、最优已付代价 C^*。

既然 Dijkstra 算法看起来如此巧妙，那为什么我们还要花费时间研究值迭代方法呢？对某些问题来说，维持排序的队列是非常昂贵的，此时值迭代却提供了一种更有效的选择。一个重要的原因就是值迭代很容易扩展到更广泛的问题。这方面的例子包括连续状态空间上的最优规划（参见第 8.5.2 节和第 14.5 节）、随机最优规划（参见第 10.2 节）、计算动态

对策平衡点（参见第 10.5 节）。在某些情况下，通过将计算集中在“有趣”的区域，仍然可能得到 Dijkstra 算法；但是随着模型更加复杂，维持该区域也可能会变得不是太有效或不可能。因此对这两种算法的正确理解对于确定在一个给定问题中哪个更合适是非常重要的。

Dijkstra 算法是更广泛的 label-correcting 算法族中的一种，这些算法通过对图 2.4 所示的一般前向搜索算法稍微进行修改就可以产生最优规划。图 2.16 给出了所产生的算法。主要区别在于如果发现更好的已付代价，那么允许状态再次成为活动的。这能使其他已付代价值得到相应改进。这对 Dijkstra 算法和 A^*搜索算法并不重要，因为它们对每个状态只需要访问一次。因此图 2.4 和图 2.16 的算法在这种情况下本质上是相同的。但是对于 Q 的任何排序，包括 FIFO（广度优先）和 LIFO（深度优先），只要 X 是有限的，那么 label-correcting 算法将产生最优解。如果 X 不是有限的，那么需要进行系统的搜索，因为必须保证足够多次重复地访问状态，以确保最终得到最优解。

```
FORWARD_LABEL_CORRECTING(x_G)
1    Set C(x)=∞ for all x≠x_I, and set C(x_I)=0
2    Q.Insert(x_I)
3    while Q not empty do
4        x←Q.GetFirst()
5        forall u∈ U(x)
6            x'←f(x,u)
7            if C(x)+l(x,u)<min{C(x'),C(x_G)} then
8                C(x')←C(x)+l(x,u)
9                if x'≠x_G then
10               Q.Insert(x')
```

图 2.16 Dijkstra 算法的一般化，它在终止时会为 Q 的任意优先级产生一个最优规划（如果存在），只要 X 是有限的；可将它与图 2.4 进行比较

label-correcting 算法与标准前向搜索模型之间另外一个重要的区别在于 label-correcting 方法用目标状态的代价来删除一些候选路径，如第 7 行所示。因此它只适合单目标状态；它经过适应性修改后也能应用于多目标状态，但是性能会下降。在第 7 行中包括 $C(x_G)$的目的在于如果它的新已付代价比 $C(x_G)$大，那么不需要为改进某状态 x′的代价而发愁；没有一条能够改进到达 x_G 的代价的路径。相似地，x_G 没有插入第 10 行是因为没有必要考虑将 x_G 作为中间状态的路径。为了得到规划，可以在第 7 行中每次更新时保存从 x 到 x'的指针，也可以通过式（2.20）采用最终的、最优已付代价 C^*来寻找行动。

2.4 用逻辑来表示离散规划

对于一些我们希望用计算机来进行计算求解的离散规划问题，状态空间是巨大的（例如有 10^{100} 个状态）。因此大量的工作在于对问题进行隐式表示，希望算法在求解时不必探索整个状态空间。这将是本书中不断出现的主题，因此问题的表示是非常重要的。一旦一

切都明确给定，那么许多规划问题其实是很平常的。

对于构造离散规划的这种隐式表示，通常采用基于逻辑的表示。一个历史的原因是这种表示是 1950 年～1980 年期间绝大多数人工智能研究的基础。另一个原因是它们对紧凑地表示某些类型的规划问题非常有用。可以认为这些表示是对问题的一种压缩表示。对一个字符串 010101010101…而言，我们能够很好地对其进行压缩，但是我们不可能压缩一个随机比特串。对离散规划来说也有相似的情况。某些问题含有一种能使它们的表达更为紧凑的规律性，而其他的问题则不可能找到这样的表示。这就是为什么在过去几十年的规划研究中会出现各种各样表示逻辑的原因。

采用基于逻辑的表示的另一个原因是许多离散规划算法是在软件系统中实现的。当这些系统求解问题时，它们必须给用户提供完整的规划，用户并不关心规划的内部。基于逻辑的表示产生的输出阐述了到达某目标的步骤。当然也存在采用其他表示的可能性，但逻辑表示以其普遍性而成为首选。

虽然有这些优点，但是逻辑表示也有不足之处，那就是它很难进行一般化。在一些应用中使诸如连续空间、不可预测性、感测不确定性以及多人决策等概念融入规划是非常重要的。这是采用状态空间表示的主要原因：它很容易扩展和适应性修改到本书所论述的其他问题。因此研究基于逻辑的表示对于理解离散规划与本书考虑的其他问题（诸如运动规划和微分约束下的规划）之间的关系是非常重要的。贯串于这些不同类型的问题，存在许多不断出现的主题，即使它们在历史上是在不同领域独立发展起来的。理解这些联系为我们洞察这些领域间的规划问题提供了强有力的帮助。

2.4.1 类似 STRIPS 的表示

类似 STRIPS 的表示是离散规划问题的最常见的逻辑表示。STRIPS 被认为是最早的一种规划算法和表示[339]，它的名字来自于斯坦福问题求解研究中心（Stanford Research Institute Problem Solver）。最初的表示采用一阶逻辑，它有很强的表达能力，但存在一些技术困难。因此这种表示后来只用于命题逻辑[746]，这种逻辑类似于本节介绍的谓词逻辑。类 STRIPS 表示有许多类型。以下是其中之一。

表述 2.4 类似 STRIPS 的规划）

1. 一个有限、非空个体（instances）集合 I。

2. 一个有限、非空谓词（predicates）集合 P，这是一个个体的二值（部分）函数。谓词应用于个体的一个特定集合被称为正文字（positive literal）。正文字逻辑取反被称为负文字（negative literal）。

3. 一个有限、非空算子集合 O，每个算子有：（1）前提条件，它是应用算子时必须成立的正文字或负文字；（2）效应，它们是作为应用算子的结果的正文字或负文字。

4. 一个初始集合 S，表示为正文字的一个集合。负文字是隐含的。对于任何不在 S 中出现的正文字，假设初始时它的相应负文字成立。

5. 一个目标集合 G，表示为正文字和负文字的一个集合。

表述 2.4.1 提供了一个以类 STRIPS 形式表示的离散可行规划的定义。3 个最重要的组成部分是个体 I、谓词 P 和算子 O。非正式地，个体描述了世界中不同事物的完整集合。

例如它们可以是书、汽车、树，等等。谓词对应于个体的基本特征或描述。例如可以用一个叫 Under 的谓词来指示像 *Under*(*Book*, *Table*)（书在桌子下面）或 *Under*(*Dirt*, *Rug*)这样的事情。一个谓词可以阐述为一个产生 TRUE 或 FALSE 的函数；但是应该注意到，它只是部分函数，因为我们可能并不希望任何个体都作为谓词的变元。

如果在一个个体上对一个谓词进行估值，例如 *Under*(*Dirt*, *Rug*)，这个表达被称为正文字。将所有可能的个体应用于谓词的定义域上可以形成所有正文字集合。每个正文字有相应的负文字，这是对正文字取反得到的。例如 ¬ *Under*(*Dirt*, *Rug*)就是相应正文字 *Under*(*Dirt*, *Rug*)的负文字，¬ 表示取反。令互补对（complementary pair）表示正文字与其相应负文字。规划问题的各种组成用正文字和负文字来表达。

算子的作用是改变世界。为了应用算子，必须满足一个前提条件的集合。这个集合的每个元素是一个正文字或负文字，为了应用算子，文字必须为 TRUE。任何可以从谓词中产生，但前提条件中未涉及的互补对可以假设是任何不影响算子适用性的值。如果算子被应用，那么世界将以一种效应（effect）集合所指定的方式更新，该效应指示应用算子所产生的正文字或负文字。即假设所有未涉及的互补对的真值不受影响。

多个算子通常通过使用变量而被定义在一个描述中。例如 *Insert*(*i*)允许插入任何个体 $i \in I$。在某些情况下，这能够大量减小表示问题所用的空间。

规划问题用正文字的初始集合 *S*、正文字和负文字的目标集合 *G* 来表达。通过为每个可能互补对选择正或负文字，就可以定义一个状态。初始集合 *S* 通过只给出正文字指定这种状态。对于没有在 *S* 中出现的所有可能的正文字，假设它们的负文字在初始状态中成立。目标集合 *G* 实际是指状态的一个集合，因为对于任何没有提及的互补对，可以选择正文字或负文字，目标都能够实现。任务是寻找一个算子序列，使得当连续应用这些算子时，能够将世界从初始状态变换为 *G* 的所有文字都为 TRUE 的世界。对于每个算子，在应用这些算子之前，前提条件必须得到满足。下面的例子阐述了表述 2.4。

例 2.6（将电池放进一个手电筒） 假设将两个电池放入手电筒这样一个规划问题，如图 2.17 所示。个体的集合是：

$$I=\{Bettery1,Bettery2,Cap,Flashlight\} \tag{2.21}$$

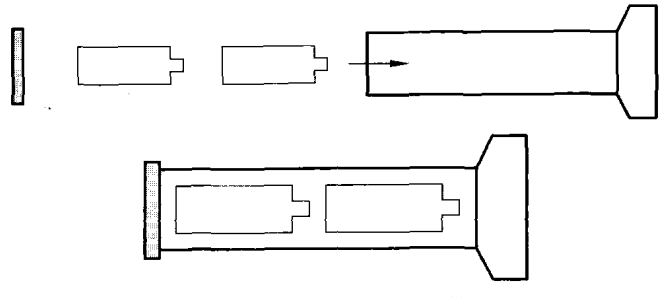


图 2.17 一个将电池放入手电筒的例子

定义两个不同的谓词 *On* 和 *In*，它们都是 *I* 上的一个部分函数。谓词 *On* 只用来评价 *Cap* 是否在 *Flashlight* 上，记为 *On*(*Cap*, *Flashlight*)。谓词 *In* 的应用有两种方式：*In*={*Bettery*1, *Flashlight*}和 *In*={*Bettery*2, *Flashlight*}，用来指示每个电池是否在手电筒中。回顾一下，谓词在一般情况下只是部分函数。对于谓词 *In*，并不希望将任意个体都用作变元。

例如定义 $In=\{Bettery1, Bettery2\}$和 $In=\{Flashlight, Bettery1\}$是没有意义的（它们可以包括在总保持一个负值的模型中，但是效率很低）。

初始集合为：

$$S=\{On(Cap, Flashlight)\} \tag{2.22}$$

基于 S，假设 $\neg In=\{Bettery1, Flashlight\}$和 $\neg In=\{Bettery2, Flashlight\}$成立。因此 S 指示手电筒的盖儿在手电筒上，但是电池在外面。

目标状态是：

$$G=\{On(Cap, Flashlight), In(Bettery1, Flashlight), In(Bettery2, Flashlight)\} \tag{2.23}$$

这意味着两个电池都在电筒中，并且电筒的盖儿也在电筒上。

集合 O 由 4 个算子组成，如图 2.18 所示。这里有一个规划能够用最少的步数到达目标状态：

$$(RemoveCap, Insert(Bettery1), Insert(Bettery2), PlaceCap) \tag{2.24}$$

简单地说，过程就是打开电筒盖儿，放入电池，盖上电筒盖儿。

Name	Preconditions	Effects
PlaceCap	$\{\neg On(Cap, Flashlight)\}$	$\{On(Cap, Flashlight)\}$
RemoveCap	$\{On(Cap, Flashlight)\}$	$\{\neg On(Cap, Flashlight)\}$
Insert(i)	$\{\neg On(Cap, Flashlight)\}, \neg In(i, Flashlight)\}$	$In(i, Flashlight)\}$

图 2.18 手电筒问题的 3 个算子。注意到 1 个算子可以用可变变元来表示，这样就可以表示不同的个体

这个例子十分简单，规划算法很容易发现问题的解。给 I 加入更多个体后，问题就复杂了，例如有更多的电池、更多电筒、一串与实现目标无关的物体，并且可以加入许多其他的谓词和算子以使算子的不同组合成为决定因素。 ■

对于采用逻辑表达的规划来说存在大量的复杂结果。图搜索问题能够在多项式时间内有效地解决；但是一个状态转移图并没有作为输入而给出。表述 2.4 所表达的输入可以用很少的个体、谓词、算子来描述大量的状态转移图。从某种意义上，当采用某些逻辑表示时，模型将得到大量压缩。这使它更接近于状态转移图的 Kolmogorov 复杂度[250,633]，这是采用图灵机可能压缩并能够完全恢复的最短比特长度。这使规划问题变得更加难以解决。简洁的输入能够对非常具有挑战性的规划问题进行编码。大多数已知的困难结果在文献[384]的第 3 章进行了概述。在大多数表述中，基于逻辑的规划是 NP 复杂度的。复杂（NP、PSPACE、EXPTIME 等）的等级取决于问题的条件。例如复杂度依赖于算子是否提前确定或者是否包含在输入中。后一种情况难度更大。基于在算子效应中以及在前提条件中是否允许负文字，可以进行复杂度分离。如果在这些情况中同时允许正文字和负文字，问题普遍会更加复杂。

2.4.2 转换到状态空间表示

刻画表述 2.4 与离散可行规划的原始形式（见表述 2.1）之间的关系是有用的。一个益处是它能够立即显示如何适应性修改第 2.2 节的搜索方法来进行逻辑表达。它对理解这两

种表示的算法复杂度之间的关系是很有帮助的。

到目前为止，只在类 STRIPS 表示中含糊地提到过“状态”的概念。现在将它具体化。假设每个谓词有 k 个变元，任何个体都能够出现在每个变元中。这意味着存在$|P||I|^k$个互补对，它们对应于将个体代入所有谓词的所有变元的方法。为了表达状态，必须从每个互补对中选择一个正文字或负文字。为方便起见，通过给个体和谓词安排一个线性顺序，这种选择可以被编码为二进制串。通过例 2.6，可以按照如下顺序指定状态：

$$(On(Cap,Flashligh),\ \neg In=\{Bettery1, Flashlight\}, In=\{Bettery2, Flashlight\} \quad (2.25)$$

采用一个二进制字符串，每个元素“0”表示一个负文字或者“1”表示一个正文字。式（2.25）编码的状态是 x=101。如果任何个体都能够出现在所有谓词的变元中，那么串的长度是$|P||I|^k$。能够区分开的可能状态的总数对应于所有可能的比特串的集合。该集合的长度为：

$$2^{|P||I|^k} \quad (2.26)$$

这意味着用很小数量的个体与谓词就可以生成一个巨大的状态空间。即使在第 2.2 节中，关于搜索图的大小来说搜索算法是有效率的，搜索算法关于 P 和 I 的大小来说效率也是非常低的。这促进了利用特定表示的结构来帮助引导搜索的技术的发展。这是第 2.5 节的内容。

转换到状态空间表示的下一步是将初始状态 x_I 编码为一个串。目标集合 X_G 是与正目标文字和负目标文字相一致的所有串的集合。通过将串字母扩展到包含“don't care”符号 δ，这就可以得到压缩。用“0”表示每个负文字、“1”表示每个正文字、δ 表示所有其他符号，用它们表示的串，足够用来表示任何用正文字和负文字表达的 X_G。

现在转换算子。对每个状态 $x \in X$，集合 $U(x)$表示 x 为满足前提条件的算子的集合。应用第 2.2 节的搜索算法，对于所有 $x \in X$，不必提前明确地确定 $U(x)$。而是在搜索中当第一次遇见 x 时，计算 $U(x)$。算子的效应采用状态转移方程来编码。对于给定的 $x \in X$，通过翻转算子的效应所指定的比特位，得到下一个状态 $f(x, u)$。

表述 2.1 的所有组成部分都已经从表述 2.4 中产生。很容易对第 2.2 节的搜索技术进行修正。也很容易将表述 2.4 扩展到表示最优规划。可以将每个算子和获取目前状态的文字集合与一个代价联系起来。这将第 2.3 节中的代价泛函 L 表达为 $l(x, u)$。因此有可能适应性修改值迭代方法使之应用于基于逻辑的表示中产生最优规划。

2.5 基于逻辑的规划方法

在过去的几十年里采用逻辑表示的规划得到了广泛的研究[384,842]。这些方法通常利用一些特殊表示的结构，并且在研究中为提高性能而开发了大量的启发式方法。本节描述一些最有影响的方法的主要思路，但是并未提出具体的启发式方法。

本节着重介绍利用逻辑表示的一些主要方法，而不是所有基于逻辑的规划方法。将会用到第 2.2 节的搜索方法。一旦用表述 2.4 给定问题的描述，那么状态转移图是在搜索过程中逐步显现出来的。实际上，相对于问题描述的大小来说，搜索图可能是巨大的。试图减少图的大小的一个方法是 STRIPS 规划算法[339,746]；它大量减少分支因素，但遗憾的是，

这又使之不够完整。本节提出的方法在保持问题的完整性的同时试图减小实际搜索的复杂性。对于每一种方法，在一些应用中非常有效，而在其他应用中它的性能就会下降。因此对特定方法的选择依赖于具体的应用。

2.5.1 部分规划空间中的搜索

除了直接在 X 中进行搜索以外，可选的另外一种方法是构造部分规划，它不涉及特殊状态。使用算子表达，递增地构造部分规划。思路是在部分规划中迭代完成所需要的子目标，同时保证没有影响求解的冲突发生。

一个部分规划（partial plan）σ 定义为：

1. 需要应用的算子的集合 O_σ。如果算子包含变量，那么这些变量可以被赋予特定的值或只作为变量。相同的算子在 O_σ 中可能出现多次，但是变量的值可能不同。

2. O_σ 上的一个偏序关系 $\prec_\sigma$，它指示对于某些有序对 $o_1,o_2 \in O_\sigma$ 来说，其中一个必须出现在另外一个的前面：$o_1 \prec_\sigma o_2$。

3. 一个绑定约束集合 B_σ，其中每个元素指示算子间的某些变量必须具有相同的值。

4. 一个因果关联的集合 C_σ，其中每个元素具有(o_1, l, o_2)的形式，表示为了满足 o_2 的一个前提条件，o_1 完成文字 l。

例 2.7（部分规划） 每个部分规划编码了可能规划的一个集合。回顾例 2.6 的模型。假设

$$O_\sigma = \{RemvoeCap,\ Insert(Bettery1)\} \tag{2.27}$$

一个有意义的排序约束是：

$$RemvoeCap \prec_\sigma Insert(Bettery1) \tag{2.28}$$

一个因果关联是：

$$(RemvoeCap,\ \neg On(Cap, Flashlight),\ Insert(Bettery1)) \tag{2.29}$$

表示 *RemvoeCap* 算子完成文字 $\neg On(Cap, Flashlight\}$，这是 *Insert*(*Bettery*1)的前提条件。这里没有绑定约束。部分规划隐含表示所有这样规划的集合：对于它们来说，在不违反因果关联的约束下，*RemvoeCap* 出现在 *Insert*(*Bettery*1)之前。 ■

几种算法用于搜索部分规划空间。为了得到对部分规划方法的直观认识，图 2.19 描述了一个规划算法。部分规划搜索图中的一个顶点是一个部分规划，通过将一个部分规划延

PLAN-SPACE 规划

1. 从任意初始部分规划 σ 开始。

2. 发现 σ 的一个缺陷，它们可能是（1）没有实现的一个算子前提条件；（2）O_σ 中的一个威胁到 C_σ 中某个因果约束的算子。

3. 如果没有缺陷，那么 σ 就是一个完整的解，并计算满足所有约束的 O_σ 的一个线性排序。

4. 如果缺陷是某算子 σ_2 的一个未实现的前提条件 l，那么发现一个算子 σ_1 来实现它，并记录一个新的因果约束(σ_1, l, σ_2)。

5. 如果缺陷是对因果关联的一个威胁，那么可以通过更新 $\prec_\sigma$ 来诱导产生一个合适的算子排序来删除该威胁，或者通过更新 B_σ 来绑定算子，解决该威胁。

6. 返回步骤 2。

图 2.19 PLAN-SPACE 中的规划是通过在规划中迭代地发现缺陷并解决它来进行的

伸得到另外一个离任务完成更近的部分规划来构造搜索图的边。尽管通用模板很简单，但是算法性能依赖于初始规划的选择以及每次迭代所解决的特定缺陷。一个直接的推广是发展多个部分规划，并在每次迭代中确定哪个得到了改进。

在以前工作中，基于部分规划的方法似乎提供了非常实在的好处；但目前与搜索状态空间的方法相比较而言，它并没有“足够的竞争力”[384]。它存在的一个问题是没有明确应用状态，更难发展出针对特定应用的启发式方法，并且与普通状态比较而言，保持和操作部分规划搜索图中的顶点的代价是很大的。

2.5.2 建立规划图

Blum 和 Furst 引入了“**规划图**”（planning graph）的概念，这是一个强有力的数据结构，它对关于哪些状态是可达的这样的信息进行了编码[119]。对于表述 2.4 描述的基于逻辑的问题，进行可达性分析。从初始状态采用广度优先搜索来扩展状态转移图。根据输入表示，所产生的搜索图的大小可能是阶段数的指数形式。这给出精确可达性信息，并保证找到目标状态。

Blum 和 Furst 的思路是构造一个比状态转移图小，仅包含部分可达性信息的图。所产生的规划图的大小是多项式的，并且对于某些有挑战性的问题，它很容易构造。有所折中的是规划图指示的是可能到达的状态。通过从所考虑的状态中删除一些不可能的状态来近似得到真正可达状态集合。这能够在搜索过程中快速删除一些不可能的选项。现在已经开发出从规划图中提取一个规划的规划算法。在最坏的情况下，该算法将花费指数级时间，这并不奇怪，因为表述 2.4 描述的问题具有 NP 复杂度。尽管如此，在一些著名的规划标准测试中，性能指标还是得到显著提高。使用规划图的另外一个方法是作为研究特定问题的启发式搜索方法的信息源。

定义规划图 一个**分层图**（layered graph）是这样的一个图，它的顶点可被分拆为层（layers）的序列，并且它的边只允许连接相邻层之间的顶点。规划图是一个分层图，其中顶点的层形成文字和算子的一个交替序列：

$$(L_1, O_1, L_2, O_2, L_3, O_3, \cdots, L_k, O_k, L_{k+1}O_{k+1}) \tag{2.30}$$

边的定义如下。对于每个算子 $o_i \in O_i$，都有一条来自于 $l_i \in L_i$ 的有向边，l_i 是 o_i 的前提条件。对每个 $l_i \in L_i$，都有一条来自于 $o_{i-1} \in O_{i-1}$ 的有向边，l_i 是 o_{i-1} 的效应。

一个重要的要求是在算子中不允许有变量。表述 2.4 中的任何包含变量的算子必须转换为包含算子的不同副本的一个集合，每个副本对应于变量的一个可能的替代值。

逐层构造 规划图从 L_1 开始逐层构造。在第一阶段，L_1 表示初始状态。S 中的每个正文字被放入 L_1，不在 S 中的每个正文字取反后也被放入 L_1。现在考虑第 i 阶段，O_i 是所有算子的集合，它们的前提条件是 L_i 的一个子集。集合 L_{i+1} 是 O_i 中所有算子的效应的并集。迭代继续，直到规划图稳定下来，即 $O_{i+1}=O_i$ 和 $L_{i+1}=L_i$。这种情况与第 2.3.2 节中值迭代的稳定非常相似。这里需要一个与终止行动 u_T 相似的技巧以使各种长度的规划都能够得到正确处理。在第 2.3.2 节终止行动的作用是防止发生状态转移。这里需要相同的思路。对于每个可能的文字 l，构造一个**平凡算子**（trivial operator），对于该算子来说，l 是仅有的前提条件和效应。平凡算子的引入，保证了一旦到达一个文字 l，那么对于文字的每个后继层，文

字都保持在规划图中。因此每个 O_i 中除了含有初始给定的集合 O 中的算子外，还包含许多平凡算子。需要这些算子来保证规划图的扩展能够到达稳态，在稳态下规划图与未来的扩展是相同的。

互斥条件 在构造规划图的过程中，保留了关于层中的算子与文字之间的冲突信息。冲突称为**互斥条件**（mutex condition），它意味着文字对[1]或算子对是相互排斥的。它们不可能同时选择而不发生某种冲突。冲突对被称为**互斥**（mutex）。对于每一层，定义一个**互斥关系**（mutex relation），它用来表示满足互斥条件的对，定义 $o, o' \in O_i$ 算子对是互斥的，如果它们满足如下任何一个条件：

（1）不一致的效应：o 的效应是 o' 的效应的负文字。

（2）抵触：o 的效应是 o' 前提条件的负文字。

（3）竞争需求：分别来自于 o 和 o' 的一对前提条件在 L_i 中是互斥的。

最后一个条件依赖于将在后面介绍的文字互斥的定义。任何文字对 $l, l' \in L_i$ 被定义为是互斥的，如果如下两个条件至少满足其中之一：

（1）负文字：l 和 l' 构成一个互补对。

（2）不一致支持：每一个实现 l 和 l' 的算子对 $o, o' \in O_{i-1}$ 是互斥的。在这种情况下，一个算子必须实现 l，另外一个必须实现 l'。如果存在一个算子实现 l 和 l'，那么这个条件是错误的，更不用说其他算子对了。

互斥的定义依赖于层，因此在规划图构造过程中互斥条件是逐层计算的。

例 2.8（手电筒的规划图） 图 2.20 显示了例 2.8 的规划图。在第一层，L_1 表示初始状态。此时只能用算子 *Re moveCap*。算子层 O_1 包含 *Re moveCap* 和 3 个用来保持从 L_1 来的文字的平凡算子。$\neg On(Cap, Flashlight)$ 的出现使得可以用插入电池的算子。因为在规划图的算子定义中不允许有变量，那么出现两个不同的算子（用 I_1 和 I_2 标记），一个算子负责

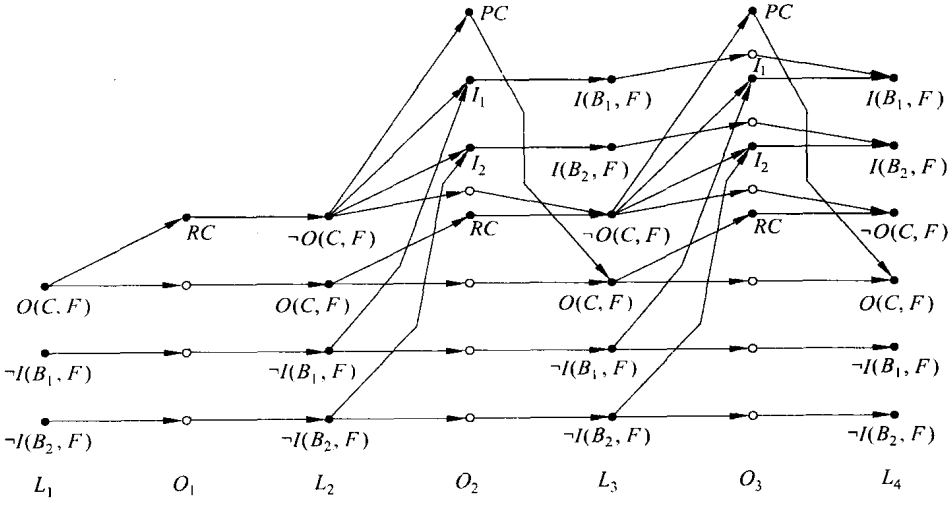


图 2.20 手电筒例子的规划图；没有标注的算子顶点对应于平凡算子；为清晰起见，算子和文字的名称被缩写

1 文字对不一定是一个互补对，如第 2.4.1 节中所定义的。

一个电池。注意从前提条件到 I_1 和 I_2 的边。手电筒的盖儿需要重新盖上，所以 *PlaceCap* 算子被包含在 O_2 中。在第三层得到所有可能的文字。在 O_3 中，包含所有可能的算子，其中包括平凡算子。最后 L_4=L_3，O_4 与 O_3 相同。这意味着规划图已经稳定。 ■

提取规划 假设规划图已经构造到 L_i。此时可以搜索规划图。如果没有发现结果而规划图已经稳定，那么一般情况下问题无解（如文献[119]所示，也可参见文献[384]）。如果规划图还没有稳定，那么进一步增加 O_i 和 L_{i+1} 进行图的扩展。继续为求解而搜索扩展后的规划图。从规划图产生的规划算法交替进行图的扩展和图的搜索。要么在某点上得到结果，要么规划图稳定后算法报告无解。所产生的算法是完备的。建立完备性的关键是文字和算子层随 i 单调递增，并且互斥对的集合单调递减，直到所有可能的冲突都得到解决。

并不是得到完全确定的规划，而是规划图产生分层规划，这是部分规划的一种特殊形式。包含所有必要的算子，并且将分层规划指定为：

$$(A_1, A_2, \cdots, A_k) \tag{2.31}$$

其中 A_i 表示算子集合。在任意 A_i 中，算子是不互斥的，并且可以以任何顺序进行而不会改变分层规划所得到的状态。仅有的约束是对于 $1 \sim k$ 的每个 i，A_i 中的每个算子必须用在 A_{i+1} 中的算子之前。对手电筒例子，从图 2.20 的规划图所构造的分层规划是：

$$(\{Re\,moveCap\},\{Insert(Battery1)\},\{Insert(Battery2)\},\{PlaceCap\}) \tag{2.32}$$

为了得到完全确定的规划，需要通过指定与层约束相一致的算子的线性排序来将分层规划线性化。对于式（2.32），这产生式（2.24）。实际规划的执行所涉及的阶段通常比规划图中的更多。对于复杂的规划问题，这种不同可能是巨大的。规划图因其较小阶段数，所以可以考虑很长的规划，因为它可以在单个层中应用多个非互斥的算子。

在每个级别上，为规划而进行搜索的代价非常大。思路是从 L_i 开始，进行后向与/或搜索。从搜索的开始，目标文字 G 必须是 L_i 的一个子集合，并且不允许有互斥对；否则立刻宣告搜索失败。从每个文字 $l \in G$，搜索的“或”部分尝试可能产生效应 l 的算子。搜索的“与”部分必须实现在以前“或”级别上选择的算子的前提条件中的所有文字。在搜索中导致另一个“或”级别的每个前提条件都必须被实现。递归地应用该思路直到得到文字的初始集合 L_1。在与/或搜索中，计算的互斥关系提供了立刻删除一些分支的信息。通常为互斥性而检查三阶或更高阶元组，即使它们不是两两互斥的。构造一个表格来有效地找回那些在搜索中多次考虑到的信息。尽管提取规划代价非常大，但它在几个重要的标准测试中显示了出众的性能见文献[119]，在最恶劣的情况下，搜索要求指数级时间（否则，对 NP 复杂度问题，将能够发现一个多项式时间算法）。

2.5.3 满足性规划

另外一个有趣的方法是将规划问题转换成一个巨大的布尔满足性问题。这意味着表述 2.4 规划问题的解决是通过确定对于产生 TRUE 值的表达式来说变量是否有可能赋某个值来实现的。确定满足性的一般方法都可以直接应用于编码规划问题的布尔表达式。Davis-Putnam 过程就是满足性算法中的一种。它进行深度优先搜索，即迭代地尝试为变量赋值，当赋值失败后按原路返回。在搜索过程中，由于目前的赋值，所以大部分表达可以被删除。算法是完备的、合理有效的。它在解决规划问题中的应用见文献[384]。除 Davis-

Putnam 过程以外，随机局部搜索方法是另外一个合理的选择见文献[462]。

假设规划问题已经由表述 2.4 给出。所有文字和算子用阶段索引加以标记。例如在两个不同阶段中出现的同一个文字是截然不同的。这种标记类似于情境演算（situation calculus）见文献[380]；但是在那种情况下，允许变量有标记。为了得到一个有限的、布尔表达式，必须声明阶段总数，令 K 表示阶段数。通常第一阶段是 $k=1$，最后阶段 $k=F=K+1$。给阶段设定一个界限是该方法的不足之处，因为在问题解决之前并不知道需要多少阶段。规划算法可以为 F 假设一个小的值，然后每次不满足所产生的布尔表达式时逐渐增加该值。如果问题没有解决，该方法将永远迭代下去。

令∨表示逻辑或，∧表示逻辑与。将布尔表达式写成许多项的合取式[1]，这些项来自于五个方面：

（1）初始状态：形成 S 中所有文字的合取，还包括对所有不在 S 中的正文字取反。它们都标记为初始阶段索引 1。

（2）目标状态：G 中所有文字的合取，用最后阶段索引 $F=K+1$ 进行标记。

（3）算子编码：每个阶段都必须复制每个算子。对于 $o\in O$，令 o_k 表示在第 k 阶段应用的算子。在所有阶段中的所有算子上形成合取。对于每个 o_k，表达式为：

$$\neg o_k \vee (p_1 \wedge p_2 \wedge \cdots \wedge p_m \wedge e_1 \wedge e_2 \wedge \cdots \wedge e_n) \tag{2.33}$$

其中 $p_1, \cdots, p_m$ 是 o_k 的前提条件，$e_1, \cdots, e_n$ 是 o_k 的效应。

（4）框架公理：这里隐含了一个假设：不是算子的效应的每个文字在下一阶段中保持不变。接下来对该假设进行编码。这也可以表述如下：如果一个文字 l 变成其取反 $\neg l$，那么包含 $\neg l$ 作为效应的算子必须被执行。如果 l 已经是一个负文字，那么 $\neg l$ 是正文字。对于每个阶段和每个文字，需要一个表示式。假设 l_k 和 l_{k+1} 是同一个文字但标记为不同阶段。表达式为：

$$(l_k \vee \neg k_{K+1}) \vee (o_{k,1} \vee o_{k,2} \vee \cdots \vee o_{k,j}) \tag{2.34}$$

其中 $o_{k,1}, \cdots, o_{k,j}$ 是用阶段 k 标记的算子，它的效应是 l_{k+1}。这确保如果 $\neg l_k$ 出现，并且后面紧跟 l_{k+1}，那么一定是某算子引起这个变化。

（5）完全不相容公理：这表示在每一阶段只应用一个算子。对第 k 阶段以及任何用阶段标记的算子对 o_k 和 o_k'，表达式是：

$$\neg o_k \vee \neg o_k' \tag{2.35}$$

该式逻辑等价于 $\neg(o_k \vee o_k')$（表示“在同一阶段中不能两个都出现”）。

文献[515]显示：规划解存在，当且仅当所产生的布尔表达式是可满足的。

下面的例子阐述了构造方法。

例 2.9（手电筒问题作为一个布尔表达式） 为例 2.6 构造一个布尔表达式。下面给出的每个表达式用∧连接成一个大的表达式。

初始状态的表达式是：

$$O(C,F,1) \wedge \neg I(B1,F,1) \wedge \neg I(B2,F,1) \tag{2.36}$$

这里使用缩写名，并且已经加入阶段标记作为谓词的变元。目标状态的表达式是：

$$O(C,F,5) \wedge I(B1,F,5) \wedge I(B2,F,5) \tag{2.37}$$

1 合取是指逻辑“与”。

表示目标必须在阶段 $k=5$ 完成。该值已经确定，因为从式（2.24）已经知道了规划解。对于非常大的 k 值，该方法仍然能够正确运行。算子的表达式为：

$$\begin{aligned}
&\neg PC_k \vee (\neg O(C,F,k) \wedge O(C,F,k+1))\\
&\neg RC_k \vee (O(C,F,k) \wedge \neg O(C,F,k+1))\\
&\neg I1_k \vee (\neg O(C,F,k) \wedge \neg I(B1,F,k) \wedge I(B1,F,k+1))\\
&\neg I2_k \vee (\neg O(C,F,k) \wedge \neg I(B2,F,k) \wedge I(B2,F,k+1))
\end{aligned} \tag{2.38}$$

每个 k 从 1 到 4。

框架公理产生表达式：

$$\begin{aligned}
&(O(C,F,k) \wedge \neg O(C,F,k+1)) \wedge (PC_k)\\
&(\neg O(C,F,k) \wedge O(C,F,k+1)) \wedge (RC_k)\\
&(I(B1,F,k) \wedge \neg I(B1,F,k+1)) \wedge (I1_k)\\
&(\neg I(B1,F,k) \wedge I(B1,F,k+1))\\
&(I(B2,F,k) \wedge \neg I(B2,F,k+1)) \wedge (I2_k)\\
&(\neg I(B2,F,k) \wedge I(B2,F,k+1))
\end{aligned} \tag{2.39}$$

每个 k 从 1 到 4。没有算子将电池从手电筒从拿出来，因此表达式中有两个没有算子。

最后，完全不相容公理产生表达式：

$$\begin{aligned}
&\neg RC_k \vee \neg PC_k \quad \neg RC_k \vee \neg O1_k \quad \neg RC_k \vee \neg O2_k\\
&\neg PC_k \vee \neg O1_k \quad \neg PC_k \vee \neg O2_k \quad \neg O1_k \vee \neg O2_k
\end{aligned} \tag{2.40}$$

每个 k 是从 1 到 4。整个问题的编码通过将所有给出的表达式组合成一个大的合取式来完成。通过给 RC_1、$IB1_2$、$IB2_3$ 和 PC_4 赋 TRUE 值，表达式就能够得到满足。另外一个可选的解是 RC_1、$IB2_2$、$IB1_3$ 和 PC_4。阶段索引标记指示在得到的规划中应用行动的顺序。 ■

进一步阅读

本章的大多数思路和方法数十年来已经为人们所熟知。第 2.2 节的多数搜索算法在关于算法的文献中作为图搜索 [245,406,695,859]和人工智能的文献中作为规划或者搜索方法[554,746,747,780,842,975]来介绍。人工智能中关于搜索的许多历史文献见文献[842]。在文献[800,801]中介绍了双向搜索方法，它与 mean-end 分析有密切关系[738]；更多关于双向搜索的讨论见文献[187,186,500,572,842]。好的启发式搜索对许多离散规划应用来说是至关重要的。关于这方面的大量材料见文献[384,553,780]。对于规划和任务调度之间的关系见文献[268,384,897]。

动态规划原理形成了最优控制理论的基础和计算机科学中的许多算法。这些思路遵循 Bellman 的最优性原理[86,87]。这些经典的研究工作直接产生了第 2.3 节的值迭代方法。关于这方面的最新资料见文献[97]，它包括 Dijkstra 算法及其推广的 label-correcting 算法。Dijkstra 算法的一个重要的特殊版本是 Dial 算法[274]（见文献[946]和第 8.2.3 节）。贯串本书始终，控制理论和规划方法之间有密切的联系。该方向的最早研究出现在文献[269]。

基于逻辑的规划的基础出现在 Nilsson 的早期工作中[339,746]，它包括第 2.4 节中的大多数概念。在最近的几十年间，出现了大量的文献。第 2.5 节对其中一些进行了简要介绍；

但是还需要更多章节来充分介绍它们的作用。一个最新的综合性的基于逻辑的规划方法见文献[384]；这里没有介绍的内容包括满足约束规划、任务调度和时间逻辑。其他关于基于逻辑的规划的资料包括文献[380,842,963,984]。文献[467]介绍了一种用来比较基于逻辑的规划算法的基准方法。

为了给本章的问题加入不确定性或者多决策者等因素，跳至第 10 章（这可能要求一些第 9 章的背景知识）。为了将离散搜索过渡到连续空间，请阅读第 5 章和第 6 章（也要求第 3 章和第 4 章的某些背景知识）。

习　　题

1. 考虑图 2.21 所考虑的规划问题。令 a 为初始状态，e 为目标状态。

（a）用后向值迭代方法来确定平稳尚需代价。

（b）用前向值迭代方法来确定平稳尚需代价。

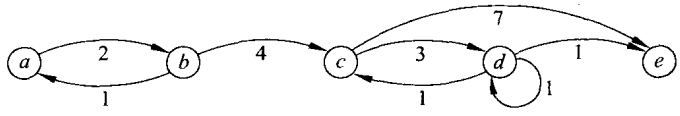


图 2.21　另外一个五状态离散规划问题

2. 尝试构造具有类似于图 2.5 所示特性的最好优先搜索的一个最坏例子。但是涉及在有障碍物的二维世界中运动，如例 2.1 所介绍的那样。

3. 值迭代可以推广到如下形式的一个代价泛函：

$$L(\pi_K)=\sum_{k=1}^{K} l(x_k,u_k,x_{k+1})+l_F(x_F) \tag{2.41}$$

其中式（2.4）中的 $l(x_k,u_k)$已经被 $l(x_k, u_k, x_{k+1})$代替。

（a）说明动态规划原理可以应用于这种更一般的情形来得到解决固定长度最优规划问题的前向和后向值迭代方法。

（b）对于使用终止条件的可变长度规划这种更一般的问题，重复进行（a）的工作。

4. 代价泛函可以推广到与阶段有关，这意味着代价除了与状态 x_k 和行动 u_k 有关以外，还依赖于特定阶段 k。将第 2.3.1 节的前向和后向值迭代方法扩展到这种情况，并说明它们能够给出最优解。更一般的代价泛函的每一项都应该表示为 $l(x_k, u_k, k)$。

5. 回顾第 2.3.2 节的定义，一个终止行动 u_T 来使得值迭代能够用于可变长度规划中，并不是要求保持在相同的状态，也可能通过产生一个被称为终端状态 x_T 的特殊状态来表述问题。当应用 u_T 时，状态变成 x_T。详细描述如何修改代价泛函、状态转移方程以及任意其他必要组成部分使得值迭代能够准确计算最短规划。

6. Dijkstra 算法在第 2.2.1 节作为一种前向搜索方法而提出。

（a）给出 Dijkstra 算法的一种后向版本，它从目标开始。试说明它总能够产生最优规划。

（b）描述（a）中的算法与第 2.3.2 节中的后向值迭代之间的关系。

（c）推导 A^*算法的一种后向版本并说明它能够产生最优规划。

7. 重新表述第 2.2.1 节的通用前向搜索算法使得它用类似 STRIPS 的表示来表达。仔细考虑规划算法需要明确构造什么以及什么仅需要隐含考虑。

8. 不用比特串，对基于逻辑的规划问题给出一种基于集合的表述。这种情况下的状态可以表示为一个正文字的集合。

9. 对表述 2.4 进行扩展以允许析取的目标集合（存在另外的必须满足的文字集合）。这是如何影响二进制串表示的？

10. 为例 2.7 设计一个 *Remove* 算子，它将电池从手电筒中取出。应用该算子，电池必须在手电筒中，并且不能被另外一个电池阻挡。对模型进行扩展使 *Remove* 算子有足够的信息来完成功能。

11. 用类似 STRIPS 表示对图 1.1(b)中的拼图游戏进行建模。可能会用到算子定义中的变量。

12. 寻找由例 2.7 隐含编码的规划的完备集合。

13. 解释表述 2.4 中的 G 为什么需要包括正文字和负文字，而 S 仅需要正文字。作为另外一种定义，S 能否只包含负文字？请给予解释。

14. 用表述 2.4 来对一个问题进行建模，其中机器人检测确定一个房间是否是黑暗的，它向灯的开关运动并打开灯。谓词应该指出机器人是否在开关处、灯是否是亮着的。需要算子来移动机器人和打开开关。

15. 为习题 14 中的模型构造一个规划图。

16. 将习题 14 中的模型表达为一个布尔满足性问题。

17. 在最坏情况下，如果将规划的布尔表达式表示为满足性，需要多少项？用$|I|$、$|P|$、$|O|$、$|S|$和$|G|$表达你的答案。

实现

18. 当有许多可选解，并且这些解都是最优的或者接近最优的，那么用 A^*搜索时性能会下降。实现 A^*搜索并基于例 2.1，在各种基于栅格的问题上对它进行评估。比较两种不同情况下的性能：

（a）用$|i'-i|+|j'-j|$作为启发，就像第 2.2.2 节所建议的。

（b）用$\sqrt{(i'-i)^2+(j'-j)^2}$作为启发。

哪种启发似乎更出众？解释你的答案。

19. 对基于栅格的问题实现 A^*、广度优先和最好优先搜索。为每种搜索算法设计和演示例子来说明其中一个明显好于其他两个。

20. 为基于栅格的规划试验双向搜索。尝试理解和解释在利用状态空间和连接多棵树的代价之间进行的折中。

21. 通过检测什么时候搜索可能陷入一个局部最小值并进行随机游走来逃离之，用这种方法来尝试改进解决习题 18 所用的方法。尝试使用最好优先搜索来代替 A^*。在可能的方法中有很大的选择灵活性。对任意特定例子，能否得到平均意义上更好的性能？

22. 通过重构例 2.5 中所得到的代价来实现后向值迭代和验证它的正确性。验证某些

复杂例子的实现过程。

23. 对于表述 2.3 下的一个规划问题，实现 Dijkstra 算法和前向值迭代。验证它们能够得到相同的规划。说明它们的性能。

24. 考虑基于栅格的问题，它通常有大的开放空间。试给出一个多分辨率搜索算法，它首先采用更大的步长，只在大步长失败的时候才采用较小的步长。实现你的思路，对例子进行试验，并改进你的方法。

第II部分 运动规划

连续空间中的规划

第II部分从离散状态空间转移到连续状态空间，这部分有两个可选的标题：（1）运动规划；（2）连续状态空间中的规划。第3章～第8章基于来自于运动规划领域的研究，这方面的研究开始于1970年，因此运动规划这个名字更多的是指即将介绍的模型和算法的族。另一方面，第II部分也可以被认为是连续状态空间中的规划，因为这是与大多数其他形式的规划之间最基本的区别。

除此之外，运动规划经常是指在包含障碍物的二维或三维空间中机器人的运动。机器人可以是真实的机器人或者诸如人类或柔性分子等任何其他运动物体。运动规划是指确定机器人采用什么合适的运动来到达目标状态，并且不与障碍物发生碰撞。可以回忆第1.2节的例子。

第2章出现的一些问题在运动规划中仍然会出现。如下两个主题对理解它们之间的联系是有帮助的。

1. 隐式表示

在第2章中我们所熟悉的主题是规划算法必须处理状态空间的隐式表示。在运动规划中，这将变得更为重要，因为状态空间是不可数无限的，并且在定义模型的世界与规划发生的空间之间存在复杂的转换。第3章介绍运动规划问题建模的方法，包括定义二维或三维几何模型以及转换这些模型。第4章介绍在这些问题中出现的状态空间。根据运动规划文献[660,591]中的概念，我们称状态空间为位形空间。位形空间的维数对应于机器人的自由度。采用位形空间，运动规划就是一种在包含隐式表示的障碍物的高维位形空间中进行的搜索过程。额外的复杂因素是位形空间具有不寻常的拓扑结构，这种结构必须正确加以刻画才能确保规划算法的正确运行。运动规划被定义为位形空间中的一条连续路径。

2. 从连续到离散

运动规划中的一个核心主题是将连续模型变换成离散模型。基于这种变换，第2章的许多算法都能够移植到运动规划中。有两种方法完成这种转换，第5章和第6章将对它们分别进行介绍。第6章介绍的组合运动规划是指算法从输入模型建立一个精确表示原始问题的离散表示形式。这就产生了完整的规划方法，它保证在有解情况下发现规划解，无解时报告规划失败。第5章介绍基于采样的运动规划，这种规划方法指的是采用碰撞检测方法对位形空间进行采样，并利用这些采样点来引导离散搜索过程的算法。在这种情况下牺牲的是完备性，但是它经常被一个更弱的概念所代替，如分辨率的完备性或随机完备性。研究第5章和第6章是非常重要的，因为每种方法都有其优缺点。而组合方法能够解决任

何运动规划问题，并且在某些限制情况下可以构造出性能非常好的结果。当然对于绝大多数“工业级”的运动规划问题来说，这些算法运行时间长、实现起来困难等不足之处使它们失去吸引力。通过解决诸如自动化装配、类人型机器人规划、分析药品设计中的组成等一些具有挑战性问题，基于采样的算法近年来已经部分地满足了这种要求。尽管不能保证完备性，但是这些方法的高效性以及容易实现的特点已经激发人们有兴趣将运动规划算法应用到更加广泛的领域中。

第II部分还有另外两章。第 7 章将对前几章的基本运动规划问题进行扩展。这些扩展包括避开运动障碍物、多机器人协调、操作规划、带闭运动链的规划。解决这些问题的算法建立在以前章节介绍的原理基础之上，但每种扩展都有新的挑战。

第 8 章是过渡性的，它虽然涉及运动规划的许多基本原理，但更关心的是在执行过程中从没有预料到的偏差中恢复过来的能力。在第III部分之前尽管对在预测中的不确定性没有显式进行建模，但是第 8 章将规划重新定义为状态空间上的函数，而不是状态空间中的路径。函数给出规划执行中所采取的行动，而不管目前到达了什么位形。这使得真实位形漂离命令位形。在第III部分对这种不确定性进行了显式建模，但这带来巨大的建模和计算代价。需要研究有效的方法来避免发生这种情况。

第 3 章　几何表示与变换

本章将介绍第 II 部分所需要的背景材料。表述和解决运动规划问题需要定义和处理空间中物体系统的复杂的几何建模。第 3.1 节引入了几何模型，这种模型主要关注于半代数建模，因为它是第 6 章的重要部分。如果读者的兴趣主要在第 5 章，那么就不必理解半代数模型。第 3.2 节和第 3.3 节分别描述如何进行单个物体和物体链的变换。这使机器人“运动”起来。这些内容是理解第 II 部分和其他部分的基础。作者希望读者已经具备一些或所有这些背景知识（特别是第 3.2 节，但是为完整起见，这里仍然包括这些内容）。第 3.4 节将物体链的变换扩展到变换物体树，这使得可以对诸如类人型机器人、柔性组织分子等一些更加复杂的系统进行建模。最后，第 3.5 节将简要介绍的变换并不假设物体都是刚体。

3.1　几 何 建 模

有各种各样几何建模的方法和技术，根据不同的应用以及问题的难易程度可以选择不同的方法。在多数情况下通常有两种选择：（1）边界表示；（2）实体表示。假设我们要定义一个行星模型。如果采用边界表示，可以写出与行星表面大体一致的球体的方程；如果采用实体表示，那么可以描述球体所包含的所有点的集合。这两种表示方法都会被介绍。

第一步是定义世界$\mathcal{W}$，$\mathcal{W}$有两种：（1）对于二维世界，$\mathcal{W}=\mathbb{R}^2$；（2）对于三维世界，$\mathcal{W}=\mathbb{R}^3$。对大多数问题而言，这就足够了；但是也许还希望表达更加复杂的世界，例如球面或更高维空间。本书避免这种一般性，因为它们目前的应用还十分有限。除非特别提出，世界均包含两部分：

（1）障碍物：世界中被“永远”占据着的部分，例如建筑物的墙壁。

（2）机器人：被几何建模、通过运动规划进行控制的物体。

上述术语的一个非常明显的应用是对一个在建筑物内移动的机器人进行建模；但是也存在其他可能性。例如：机器人可以是一个柔性分子，障碍物可以是一个折叠起来的蛋白质。再比如：机器人可以是有障碍物的图形模拟中的一个虚拟人（想象三维视频游戏中的家庭）。

本节提出一个系统地构造障碍物和机器人表示的方法，方法采用的是本原族。认为障碍物与机器人都是$\mathcal{W}$的（闭）子集。令障碍区域$\mathcal{O}$表示$\mathcal{W}$中一个或多个障碍物上的所有点的集合；因此$\mathcal{O}\subseteq\mathcal{W}$。下一步是定义一个表示$\mathcal{O}$的系统方法，方法在有效地进行计算的同时，也具有巨大的表达能力。机器人也采用相同的方法进行定义；但在第 3.2 节中进行，在那里定义了几何物体的变换。

3.1.1 多边形与多面体模型

本节和下一节用本原的组合来进行$\mathcal{O}$的实体表示。每个本原H_i代表$\mathcal{W}$的一个子集，它在计算机中很容易表示和操作。一个复杂障碍区域用有限个本原的布尔组合表示。根据集合论，这意味着$\mathcal{O}$用本原的有限数量的并集、交集、集差来定义。

凸多边形 首先考虑的情况是障碍区域是凸的、二维空间$\mathcal{W}=\mathbb{R}^2$的多边形子集。一个子集$X\subset\mathbb{R}^n$是凸的，当且仅当对于X中的任何点对，连接该点对的线段上的所有点都在X的内部。更准确地，这意味着对任何$x_1,x_2\in X$以及$\lambda\in[0,1]$，那么：

$$\lambda x_1+(1-\lambda)x_2\in X \tag{3.1}$$

因此在x_1和x_2之间插值产生的点总在X内。直观地，X没有凹缺。一个不是凸的集合称为非凸的（不同于“凹”的，“凹”的似乎更适宜于透镜）。

$\mathcal{O}$的边界表示是一个m边形，它有两个要素：顶点和边。每个顶点就是多边形的角；每条边是指两个顶点间的线段。多边形可以用$\mathbb{R}^2$中m点的序列$(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)$来表示，沿逆时针方向。

$\mathcal{O}$的实体表示是表示为m个半平面的交集。每个半平面就是多边形的边的一侧所有点的集合。图3.1给出一个八边形，它表示为8个半平面的交集。

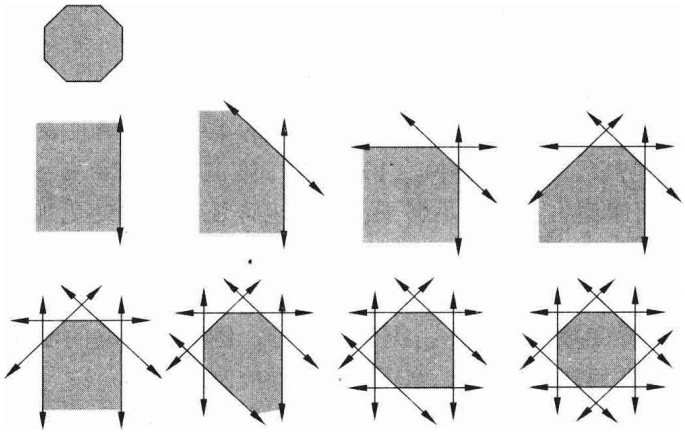


图3.1 凸多边形区域可以用半平面的交集来表示

多边形的边用两点表示，例如(x_1,y_1)和(x_2,y_2)。考虑经过(x_1,y_1)和(x_2,y_2)的直线的方程。方程可以采用$ax+by+c=0$的形式来确定，其中$a,b,c\in\mathbb{R}$是由x_1,y_1,x_2,y_2确定的常数。令$f:\mathbb{R}^2\to\mathbb{R}$表示由$f(x,y)=ax+by+c$给定的函数。$f(x,y)<0$表示线的一侧，$f(x,y)>0$表示线的另外一侧。$f$也可以用从$(x,y)$到直线的带符号的欧氏距离来表示。$f(x,y)$的符号表示一个以直线为界的半平面，如图3.2所示。不失一般性，假设$f(x,y)$的定义使得$f(x,y)<0$表示从(x_1,y_1)到(x_2,y_2)的边的左边所有的点。

令$f_i(x,y)$表示由从(x_i,y_i)到(x_{i+1},y_{i+1})的边所产生的f函数，其中$1\leqslant i<m$。令$f_m(x,y)$表示直线的方程，它对应于从(x_m,y_m)到(x_1,y_1)的边。对于$1\leqslant i\leqslant m$，半平面H_i定义为$\mathcal{W}$的子集：

$$H_i=\{(x,y)\in\mathcal{W}\,|\,f_i(x,y)\leqslant 0\} \tag{3.2}$$

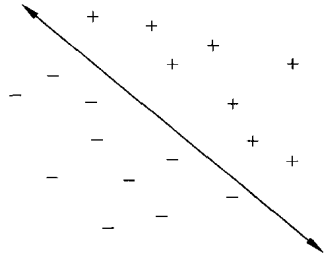


图 3.2　$f(x,y)$的符号将 $\mathbb{R}^2$ 分割成三个区域：由 $f(x,y)<0$ 和 $f(x,y)>0$ 所给出的两个半平面和直线 $f(x,y)=0$

其中 H_i 是描述线 $f_i(x,y)=0$ 的一侧所有点的集合的一个本原（包括线上的点）。一个凸的、有 m 个边的多边形障碍区域 $\mathcal{O}$ 表示为：

$$\mathcal{O}=H_1 \cap H_2 \cap \cdots \cap H_m \tag{3.3}$$

非凸多边形　对许多应用而言，$\mathcal{O}$ 是凸的这个假设过于严格。现在假设 $\mathcal{O}$ 是 $\mathcal{W}$ 的非凸的多边形子集。这种情况下，$\mathcal{O}$ 的表示为：

$$\mathcal{O}=\mathcal{O}_1 \cup \mathcal{O}_2 \cup \cdots \cup \mathcal{O}_n \tag{3.4}$$

其中 $\mathcal{O}_i$ 是用式（3.3）半平面表示的一个凸的多边形集合。当 $i \neq j$ 时的 $\mathcal{O}_i$ 和 $\mathcal{O}_j$ 不必是没有交集的。采用这种表示，可以定义 $\mathcal{W}$ 中非常复杂的障碍区域。尽管这些区域可能包含多个分支和孔，如果 $\mathcal{O}$ 是有界的（例如 $\mathcal{O}$ 能够装入一个足够大的矩形方盒中），那么它的边界由线段组成。

一般情况下，用本原的任何有限并集、交集、集差可以定义 $\mathcal{O}$ 的更加复杂的表示；但是总可能将复杂的表示简化为式（3.3）和式（3.4）的形式。重新定义本原可以避免集差的出现。假设模型需要删除一个由包含[1] $f_i(x,y)<0$ 的本原 H_i 所定义的集合。这等价于使所有点满足 $f_i(x,y)\geqslant 0$，即 $-f_i(x,y)\leqslant 0$。这可以用来定义一个新本原 H_i'，当 H_i' 与其他集合进行并集运算时就等价于去掉了 H_i。给定本原的一个复杂组合，一旦去掉集差，那么通过布尔代数法则，表达式就可以简化为有限交集的一个有限并集。

非凸多边形并不是唯一的。有许多方法将 $\mathcal{O}$ 剖分为多个凸分支。应该仔细选择剖分方法以保证算法的最优计算性能。在多数情况下，这些分支可以允许重叠。理想情况下，最好是用最小数量的本原表示 $\mathcal{O}$，自动进行这种剖分过程可能会导致一个 NP 复杂度问题（第 6.5.1 节将简要介绍 NP 复杂度）。一个高效、实用的剖分方法是用垂直胞腔剖分算法，见第 6.2.2 节。

定义一个逻辑谓词　以前表示的值是什么呢？作为一个简单的例子，这里定义一个逻辑谓词，它用作碰撞检测器。回忆第 2.4.1 节，谓词是一个布尔值函数。令 ϕ 表示谓词：$\mathcal{W}\rightarrow\{\text{TRUE,FALSE}\}$，如果一个 $\mathcal{W}$ 的点在 $\mathcal{O}$ 中，那么返回 TRUE；否则，返回 FALSE。对于一条由 $f(x,y)=0$ 给定的直线，令 $e(x,y)$ 表示一个逻辑谓词：如果 $f(x,y)<0$，那么返回 TRUE；否则，返回 FALSE。

与凸多边形区域相对应的谓词用逻辑合取式表示：

$$\alpha(x,y)=e_1(x,y)\wedge e_2(x,y)\wedge\cdots\wedge e_m(x,y) \tag{3.5}$$

1　在本节中，我们想使所产生的集合包括所有沿着边界的点。因此用<来建模删除部分的集合，而不是用≤。

如果点(x, y)在凸多边形区域内，谓词$\alpha(x, y)$返回 TRUE；否则，返回 FALSE。一个由n个凸多边形组成的障碍区域用逻辑析取式表示：

$$\phi(x,y)=\alpha_1(x,y)\vee\alpha_2(x,y)\vee\cdots\vee\alpha_n(x,y) \tag{3.6}$$

尽管有更有效的方法，ϕ能够用来检查在$o(n)$时刻点(x, y)是否在$\mathcal{O}$内，其中n是在$\mathcal{O}$的表示中本原的个数（在固定时间内计算每个本原），尽管还有更有效的方法。

注意逻辑谓词表示与集合论表示之间的联系。采用逻辑谓词，集合理论表示的并集、交集用逻辑“或”和“与”代替。从布尔代数知道任何复杂逻辑命题都能够简化成一个逻辑析取式（在计算机工程中叫做“积的和”）。这等价于前面的陈述：$\mathcal{O}$总是可以表示为本原的交集的一个并集。

多面体模型 对于三维世界$\mathcal{W}=\mathbb{R}^3$，用多面体代替多边形、半空间本原代替半平面本原，前面的概念就可以从二维扩展到三维。边界表示可由三个要素来定义：顶点、边和面。每个面是$\mathbb{R}^3$中的一个“平的”多边形。每条边是两个面之间的边界。每个顶点形成三条或三条以上边的边界。

几种数据结构可以使我们方便地“穿行”在多面体的各个要素之间。例如，二重连接边的列表[266]包含三种记录：面、半边以及顶点。直观地，半边是一条有向边。每个顶点记录点的坐标以及指向任意与该顶点有接触的半边的指针。每个面记录包含一个在它的边界上指向任意半边的指针。每个面由半边的循环列表构成。多面体的每条边都有一个有向半边记录的对。在图 3.3(b)中每个半边都用一个箭头表示。每个半边记录包含指向 5 个其他记录的指针：（1）半边开始的顶点；（2）限制相邻面的“双胞胎”半边，它们方向相反；（3）半边所限制的面；（4）用来限制面的边的循环列表中的下一个因素；（5）用来限制面的边的循环列表中的上一个因素。一旦定义了所有这些记录，那么对多面体的结构进行详细研究就非常方便。

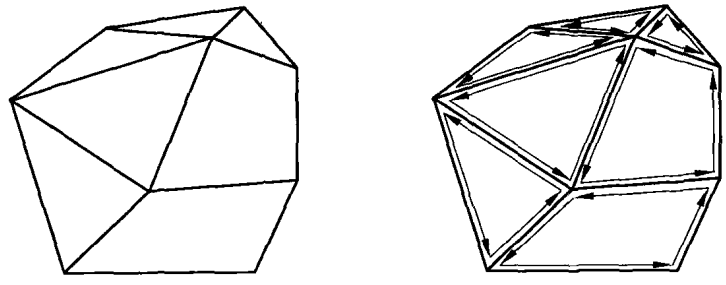


图 3.3 (a) 多面体可以用面、边和顶点来进行描述；(b) 每个面的边可以保存在一个循环列表中，它按照关于面的向外的法向量逆时针的顺序进行遍历

现在考虑多面体的实体表示。假设$\mathcal{O}$是一个凸多面体，如图 3.3 所示。可以用顶点来构造实体表示。$\mathcal{O}$的每个面沿着边界至少有 3 个顶点。假设这些顶点不共线，那么经过这些点的面的方程有如下形式：

$$ax+by+cz+d=0 \tag{3.7}$$

其中$a,b,c,d\in\mathbb{R}$是常数。

构造$f:\mathbb{R}^3\to\mathbb{R}$

$$f(x,y,z)=ax+by+cz+d \tag{3.8}$$

令 m 是面的数量。对$\mathcal{O}$的每个面，半空间 H_i 定义为$\mathcal{W}$的子集：

$$H_i=\{(x,y,z)\in\mathcal{W}|f_i(x,y,z)\leqslant 0\} \tag{3.9}$$

重要的是选择 f_i 以使在多面体的内部它取负值。在多边形模型中，沿着边界以逆时针顺序定义 f_i 是可能的。在多面体情况下，使用半边数据结构得到每个面的边的列表，这些边以逆时针的顺序构成面的边界。图 3.3(b)显示每个面的边的顺序。对每条边，箭头指向相反方向，这是半边数据结构的要求。每个面的方程这样确定。在面的边界上按逆时针的顺序选择三个连续的顶点 p_1,p_2,p_3（它们必须是不共线的）。令 v_{12} 表示从 p_1 到 p_2 的向量，令 v_{23} 表示从 p_2 到 p_3 的向量。叉积 $v=v_{12}\times v_{23}$ 产生一个指向多面体外且与面正交的向量。回忆到向量[a　b　c]与面的正交向量平行。如果它的元素 $a=v[1]$、$b=v[2]$、$c=v[3]$，那么对于包含多面体的半空间中的所有点来说都满足 $f(x,y,z)\leqslant 0$。

像多边形模型一样，凸多面体可以定义为有限数量的半空间的交集，每个半空间对应于一个面。一个非凸多面体可以定义为有限数量凸多面体的并集，谓词 $\phi(x,y,z)$ 也能用相似的方式来定义，在这种情况下，如果$(x,y,z)\in\mathcal{O}$，那么返回 TRUE；否则，返回 FALSE。

3.1.2　半代数模型

在多边形和多面体模型中，f 是线性函数。在二维空间的半代数模型中，f 可以是任何实值系数、实数变量 x 和 y 的多项式。对三维空间，f 是变量 x、y 和 z 的多项式。半代数模型的类型包括多边形模型和多面体模型，它们都采用一阶多项式。由单个多项式本原确定的点集称为“代数集合”；由代数集合的有限的并与交构成的点集称为半代数集。

在二维空间中，可以用如下形式的代数本原定义一个实体表示：

$$H=\{(x,y)\in\mathcal{W}|f(x,y)\leqslant 0\} \tag{3.10}$$

作为一个例子，令 $f=x^2+y^2-4$。在这种情况下，H 表示一个中心在原点、半径为 2 的圆盘。这对应于满足 $f(x,y)\leqslant 0$ 的点(x,y)的集合，如图 3.4(a)所示。

例 3.1（姜饼面） 构造如图 3.4(b)所示的一个阴影区模型。令外圆的半径为 r_1、中心在原点。假设“眼睛”的半径为 r_2 和 r_3，中心在(x_1,y_1)和(x_2,y_2)，令“嘴”是半长轴为 a、半短轴为 b、中心在$(0,y_4)$的椭圆，那么定义函数：

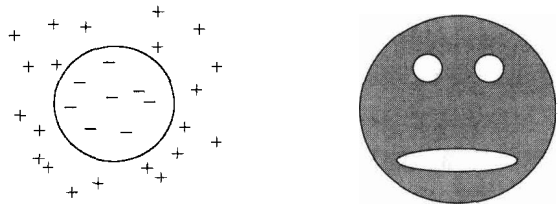
$$\begin{aligned}
f_1&=x^2+y^2-r_1^2\\
f_2&=-((x-x_2)^2+(y-y_2)^2-r_2^2)\\
f_3&=-((x-x_3)^2+(y-y_3)^2-r_3^2)\\
f_4&=-(x^2/a^2+(y-y_4)^2/b^2-1)
\end{aligned} \tag{3.11}$$

对于 f_2,f_3,f_4，熟悉的圆和椭圆方程乘以－1 就产生圆或椭圆外的所有点的代数本原。阴影区域$\mathcal{O}$表示如下：

$$\mathcal{O}=H_1\cap H_2\cap H_3\cap H_4 \tag{3.12}$$

■

在半代数模型中，本原的交集不一定产生$\mathcal{W}$的凸子集。一般情况下，可能需要用代数本原的并集和交集来形成$\mathcal{O}$。



(a) (b)

图 3.4 (a) f用来将$\mathbb{R}^2$分为两个区域，在这种情况下，代数本原表示一个碟形区域；(b) 只用四个代数本原就可以准确建模阴影的“脸”

可以再次构造逻辑谓词$\phi(x,y)$，在与本原的数量成线性关系的时间内完成碰撞检测。仍然可以很有效率地计算每个本原，f仅仅是在点(x,y)上进行计算的多项式。

半代数的表述很容易扩展到三维空间。这会导致如下形式的代数本原：

$$H=\{(x,y,z)\in\mathcal{W}|f(x,y,z)\leqslant 0\} \tag{3.13}$$

这些本原可以用来定义三维障碍$\mathcal{O}$的实体表示以及一个逻辑谓词ϕ。

方程（3.10）和方程（3.13）足以表达任何感兴趣的模型。还可以定义其他基于不同关系的本原。例如：$f(x,y,z)\geqslant 0$，$f(x,y,z)=0$，$f(x,y,z)<0$，$f(x,y,z)\neq 0$ 等；但是它们并不提高能够被表达的模型的集合。但在不同的情形中采用不同的关系会更加方便。为了说明某些本原不允许表达新的模型，考虑本原：

$$H=\{(x,y,z)\in\mathcal{W}|f(x,y,z)\geqslant 0\} \tag{3.14}$$

等式右边也可以表示成$-f(x,y,z)\leqslant 0$，可以认为$-f$是x, y和z的新的多项式函数。作为一个涉及“$=$”关系的例子，考虑其本原：

$$H=\{(x,y,z)\in\mathcal{W}|f(x,y,z)=0\} \tag{3.15}$$

它可以表示为$H=H_1\cap H_2$，其中：

$$H_1=\{(x,y,z)\in\mathcal{W}|f(x,y,z)\leqslant 0\} \tag{3.16}$$

$$H_2=\{(x,y,z)\in\mathcal{W}|-f(x,y,z)\leqslant 0\} \tag{3.17}$$

“$<$”关系确实能够增加表达能力，如果用它来构造本原[1]。构造不包括外部边界的模型是必要的（例如：球的内部所有点的集合，不包括球上的点）。这些通常被称为“开集”，将在第 4 章中定义。

3.1.3 其他模型

模型的选择依赖于规划算法所进行的运算的类型。对于组合运动规划方法（见第 6 章），特殊的表示是至关重要的。另一方面，对于基于采样的规划方法（见第 5 章），特殊表示仅对碰撞检测算法而言很重要，当考虑规划时，算法被当作“黑箱”。因此本节其余部分给出的模型更可能在基于采样的方法中出现，对于规划算法的设计者来说可能是不可见的（尽管完全忘记这些表示是不明智的）。

非凸多边形和多面体 第 3.1.1 节中的方法要求非凸多边形表示成凸多边形的并集。而

1 另外一个有相同表达能力的方法还是使用$\leqslant$，但是除了允许并集和交集以外，还允许集合的补集。

这里是通过用特定顺序对顶点进行列表来对非凸多边形边界表示直接进行编码；假设指定逆时针顺序。有 m 个顶点的多边形可以以一个列表的形式$(x_1, y_1),(x_2, y_2),\cdots,(x_m, y_m)$进行编码。假设对于 i 从 1 到 $m-1$ 的每个点(x_i, y_i)与(x_{i+1}, y_{i+1})之间都有一个边，在(x_m, y_m)和(x_i, y_i)之间也有一条边。通常地，选择的顶点应该使多边形更加简单，也就是说没有边相交。在这种情况下，如果顶点以逆时针的顺序进行列表，那么多边形的内部就是每条边的左边。

如果多边形有一个孔，会怎样呢？在这种情况下，孔的边界可以表示成一个多边形，但是它的顶点以顺时针的顺序出现。每个边的左边是外部多边形的内部，右边是孔，如图 3.5 所示。

尽管三维情况下的数据结构有些复杂，非凸多面体的边界表示仍然可以用相似的方法进行表达。在这种情况下不用边的列表，而是必须指定面、边以及顶点，还有表示它们的关联关系的指针。仍然必须选择一致的方向，然后选择相反方向对孔进行建模。

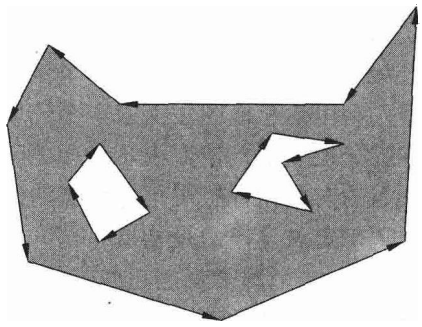


图 3.5　一个有孔的多边形可以用不同的方向来表达：外边界用逆时针，孔的边界用顺时针；注意到，当沿着箭头前进时，阴影部分总在左边

三维三角形　假设 $\mathcal{W}=\mathbb{R}^3$，最便于表达的几何模型是三角形集合，每个三角形由 3 个点(x_1, y_1, z_1)、(x_2, y_2, z_2)、(x_3, y_3, z_3)确定。在计算机图形学中这种模型非常普遍，因为图形加速的硬件使用三角形本原。假设三角形的内部是模型的一部分。如果一个三角形插入另一个三角形的内部，则认为两个三角形产生“碰撞”。这种模型提供了巨大的灵活性，因为对三角形的表达方式没有约束；但这也是它的众多不足之一。没有可以用来断定一个点是在三维障碍物的内部或外部的相干性。如果有某些相干性，那么有时我们更愿意减少三角形坐标的性能冗余（一些三角形分享相同的角）。去除冗余的表示被称为三角形条或者三角形扇形。三角形条是这样一个三角形序列：每个邻近三角形的对都享有一条共同边；三角形扇形是一个三角形条，其中所有三角形有一个共同顶点，如图 3.6 所示。



图 3.6　三角形形成的条形和扇形能够减小冗余点的数量

不一致有理数 B 样条（NURBS）　这种 B 样条在工程设计系统中得以应用，在诸如飞机和可移动物体的设计应用中，使用 NURBS 可以方便地对曲面进行设计和调整。与半代数模型的隐式方程不同，NURBS 和其他样条是参数方程。这使得透视图的计算更加容易；但也使碰撞检测变得更加困难。这些模型可以是任意维的。这里给出一个简单的二维表述。

一条曲线可以表示为：

$$C(u)=\frac{\sum_{i=0}^{n} w_i P_i N_{i,k}(u)}{\sum_{i=0}^{n} w_i N_{i,k}(u)} \tag{3.18}$$

其中 $w_i \in \mathbb{R}$ 是权重，P_i 是控制点。$N_{i,k}$ 是归一化的 k 阶基函数，它可以表示为递归形式：

$$N_{i,k}(u)=\left(\frac{u-t_i}{t_{i+k}-t_i}\right)N_{i,k-1}(u)+\left(\frac{t_{i+k+1}-u}{t_{i+k+1}-t_{i+1}}\right)N_{i+1,k-1}(u) \tag{3.19}$$

递归的基是 $N_{i,0}(u)=1$，如果 $t_i< u < t_{i+1}$；在其他情况下，$N_{i,0}(u)=0$。节向量（knot vector）是真值的一个非递减系列$\{t_0, t_1,\cdots,t_m\}$，它控制某些基函数生效的区间。

位图 对于 $\mathcal{W}=\mathbb{R}^2$ 或 $\mathcal{W}=\mathbb{R}^3$，将一个有界空间离散化为矩形胞腔（单元），这些胞腔可以被占据，也可以不被占据。所产生的模型与例 2.1 很相似。这个离散化的分辨率决定每个轴上胞腔的数量以及近似后的效果。表示可以看作一个二值图像，该图像中的“1”表示包含至少一个$\mathcal{O}$的点的矩形区域，“0”表示不包含$\mathcal{O}$的任何点的矩形区域。尽管位图不像其他模型那么精确，但是它经常出现在具体应用中。一个例子就是可移动机器人所构造的数字地图，带着传感器的机器人利用数字地图去探索一个环境。位图的扩展就是灰度图。在这种情况下，可以给每个胞腔分配一个数值，表示诸如“障碍物存在的可能性”、“穿越胞腔的预期难度”等数量。下面的内容在行星飞行器导航的地形图中经常被使用。

超二次曲面 不采用多项式来定义 f_i，而是构造一些一般性。一个普通的本原是一个超二次曲面，它是二次曲面的扩展。一个例子就是超椭圆体，对于 $\mathcal{W}=\mathbb{R}^3$，它由下式给出：

$$\left(\left|x/a\right|^{n_1}+\left|y/b\right|^{n_2}\right)^{n_1/n_2}+\left|z/c\right|^{n_1}-1\leqslant 0 \tag{3.20}$$

其中 $n_1\geqslant 2$，$n_2\geqslant 2$。如果 $n_1=n_2=2$，那么就产生一个椭圆。当 n_1 和 n_2 增加，超椭圆体的形状就变得像一个带圆角的盒子。

广义圆柱 广义圆柱是普通圆柱的推广。中心轴并不局限于直线，而可以是连续中心曲线$(x(s), y(s), z(s))$，参数 $s\in[0,1]$。半径也不是固定的，半径函数 $r(s)$沿着中心轴定义。广义圆柱在点$(x(s), y(s), z(s))$处的截面是圆，圆柱半径函数的值 $r(s)$就是这个圆的半径。截面的法线与中心曲线在 s 处相切。

3.2 刚体变换

第 3.1 节中的任何技术都可以用来定义障碍物和机器人。令$\mathcal{O}$表示障碍区，它是$\mathcal{W}$的子集。令$\mathcal{A}$表示机器人，它是 $\mathbb{R}^2$ 或 $\mathbb{R}^3$ 的子集，与$\mathcal{W}$的维数相同。尽管$\mathcal{O}$在$\mathcal{W}$中保持固定，但是运动规划问题要求机器人$\mathcal{A}$“运动”。

3.2.1 一般概念

在给出特定变换之前，当直观概念失效时为了避免与后续内容产生混淆，需要定义这些变换的一般形式。假设刚体机器人$\mathcal{A}$是 $\mathbb{R}^2$ 或 $\mathbb{R}^3$ 的子集。那么刚体变换是这样的函数 $h:\mathcal{A}\rightarrow\mathcal{W}$，它将$\mathcal{A}$中的每个点映射到$\mathcal{W}$时有两个要求：（1）$\mathcal{A}$的任何点对之间的距离必须保持；（2）必须保持$\mathcal{A}$的方向（无“镜像”）。

采用标准函数符号，对于某 $a \in \mathcal{A}$，$h(a)$表示$\mathcal{W}$中被 a 占据的点。令：

$$h(\mathcal{A}) = \{h(a) \in \mathcal{W} | a \in \mathcal{A}\} \tag{3.21}$$

它是 h 的像（image），表示$\mathcal{W}$中被变换后的机器人占据的所有点。

变换机器人模型　考虑变换机器人模型。如果通过命名 $\mathbb{R}^2$ 中特定点来表达$\mathcal{A}$，就像多边形的边界表示一样，那么每个点从 a 变换到 $h(a) \in \mathcal{W}$。在这种情况下，用 h 对整个模型进行变换就非常简单。但是如果机器人模型用如下本原表达，变换会稍微复杂些：

$$H_i = \{a \in \mathbb{R}^2 \mid f_i(a) \leqslant 0\} \tag{3.22}$$

这与式（3.2）稍有不同，因为机器人定义在 $\mathbb{R}^2$ 中（不必在$\mathcal{W}$），a 被用来表示点 $(x, y) \in \mathcal{A}$。通过 h 变换，本原变换为：

$$h(H_i) = \{h(a) \in \mathcal{W} | f_i(a) \leqslant 0\} \tag{3.23}$$

为了完全地变换本原，最好直接命名 $w \in \mathcal{W}$ 中的点，而不是 $h(a) \in \mathcal{W}$。基于 $a = h^{-1}(w)$ 的事实，这可以变成：

$$h(H_i) = \{w \in \mathcal{W} | f_i(h^{-1}(w)) \leqslant 0\} \tag{3.24}$$

其中 h 的反函数出现在右边，那是因为需要原始点 $a \in A$ 来计算 f_i。因此需要非常仔细，因为需要 h 或者 h^{-1} 来对模型进行变换。这可以从一些后续的例子中观察到。

参数化的变换族　研究变换的族非常重要，因为可以通过不同的参数来选择特定的变换。因此一般化 h 来接收两个参数是非常有意义的：一个是参数向量 $q \in \mathbb{R}^n$，另一个是 $a \in \mathcal{A}$。所产生的变换后的点 a 用 $h(q, a)$表示，整个机器人变换到 $h(q, \mathcal{A}) \subset \mathcal{W}$。

将用到下面的速记符号，它要求特别的 h 从上下文中加以推断。令 $h(q, a)$简化为 $a(q)$，$h(q, \mathcal{A})$简化为$\mathcal{A}(q)$，这种符号使得这种情况能够出现：当通过调整参数 q，从参数化的变换族中选择不同的变换时，机器人$\mathcal{A}$在$\mathcal{W}$周围移动。它使用起来非常方便。表达$\mathcal{A}(q)$可以看作是一个集合的值函数，该函数用 q 进行变换，从而产生$\mathcal{W}$中被$\mathcal{A}$占据的点的集合。大多数时间内，该符号不会产生问题，但当问题产生时，记住本节中的定义是非常有用的，特别是当试图确定是否需要 h 或者 h^{-1}。

定义坐标系　假设$\mathcal{A}$定义在 $\mathbb{R}^2$ 或者 $\mathbb{R}^3$ 中，但是在转换前，并不认为它是$\mathcal{W}$的子集。h 变换将机器人放置在$\mathcal{W}$ 中。在下面的内容中，用坐标系来指示这种区别是非常方便的。$\mathcal{W}$ 的原点和坐标基向量称为“世界坐标系”[1]。因此任何点 $w \in \mathcal{W}$ 都用世界坐标系来表达。

用来定义$\mathcal{A}$的坐标最初是在体坐标系中表达的，该坐标表示 $\mathbb{R}^2$ 或者 $\mathbb{R}^3$ 的原点和坐标基向量。在 $\mathcal{A} \subset \mathbb{R}^2$ 情况下，可以想象或将体坐标系描绘在机器人上。变换机器人等价于将它的模型从体坐标系转换到世界坐标系。在某位置和定向下，这有将$\mathcal{A}$放置[2]在$\mathcal{W}$中的效果。当第 3.3 节介绍多体问题时，每个物体都有它自己的体坐标系，变换要求关于世界坐标系对所有物体进行表达。

1　世界坐标系就是牛顿力学中的惯性坐标系；直觉地，它是一个静止不动的坐标系，可以在其上进行所有测量，参见第 13.3.1 节。

2　从技术上讲，这种“放置”是一个函数，被称为保向等距嵌入。

3.2.2 二维变换

平移 刚体 $\mathcal{A} \subset \mathbb{R}^2$ 用两个参数 $x_t, y_t \in \mathbb{R}$ 进行平移。采用第 3.2.1 中的定义，$q = (x_t, y_t)$ 和 h 定义为：

$$h(x,y) = (x + x_t, y + y_t) \tag{3.25}$$

采用式（3.25）对多边形顶点序列中的每个顶点进行变换就可以实现对 $\mathcal{A}$ 的边界的平移。序列中的每个点(x_t, y_t)用(x_i+x_t, y_i+y_t)代替。

考虑用本原定义的$\mathcal{A}$的实体表示。将每个如下形式的本原：

$$H_i = \{(x,y) \in \mathbb{R}^2 \mid f(x,y) \leqslant 0\} \tag{3.26}$$

变换为：

$$h(H_i) = \{(x,y) \in \mathcal{W} \mid f(x - x_t, y - y_t) \leqslant 0\} \tag{3.27}$$

例 3.2（平移一个圆盘） 假设机器人是一个中心在原点的单位半径圆盘。它用单个本原进行建模：

$$H_i = \{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 - 1 \leqslant 0\} \tag{3.28}$$

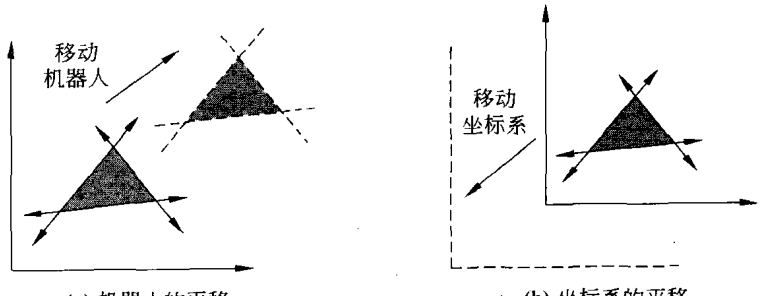
假设将$\mathcal{A}=H_i$沿 x 轴方向平移 x_t 个单元，沿 y 轴方向平移 y_t 个单元。变换后的本原为：

$$h(H_i) = \{(x,y) \in \mathcal{W} \mid (x - x_t)^2 + (y - y_t)^2 - 1 \leqslant 0\} \tag{3.29}$$

这就是我们所熟悉的原点在(x_t, y_t)的圆盘方程。在这个例子中用到第 3.2.1 节所描述的反函数 h^{-1}。■

平移后的机器人用$\mathcal{A}(x_t, y_t)$表示。平移(0,0)实际上是恒等变换，即$\mathcal{A}(0,0)=\mathcal{A}$，如果假设 $\mathcal{A} \subset \mathcal{W}$（$\mathcal{A}$开始时并不一定在$\mathcal{W}$中）。采用自由度的概念来表示完全刻画变换的特征所需要的独立参数最大数量。如果 x_t 和 y_t 的允许值的集合构成一个 $\mathbb{R}^2$ 上的二维子集，那么自由度是 2。

假设$\mathcal{A}$直接在$\mathcal{W}$中进行定义。那么如图 3.7 所示，应用于$\mathcal{A}$的刚体变换有两种阐述：（1）世界坐标系保持固定，而机器人被变换；（2）机器人保持固定，而世界坐标系被平移。第一种阐述刻画了从固定的世界坐标系进行变换的效果，第二种阐述从机器人的视角指出变换是如何出现的。第一种阐述将应用于运动规划问题，因为它经常用来对在物理空间中运动的机器人进行建模。大量的书籍介绍的坐标变换则是在第二种阐述下进行的。这已经引起混淆，因为变换有时从运动规划中所期望的状态出现“后退”现象。



(a) 机器人的平移　　(b) 坐标系的平移

图 3.7　每个变换都有两种解释

旋转　机器人$\mathcal{A}$能够被逆时针旋转一定角度$\theta\in[0,2\pi)$，通过将每个$(x,y)\in\mathcal{A}$映射为

$$(x,y)\mapsto(x\cos\theta-y\sin\theta,x\sin\theta+y\cos\theta) \tag{3.30}$$

采用一个 2×2 旋转矩阵：

$$R(\theta)=\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix} \tag{3.31}$$

那么变换可以写成：

$$\begin{pmatrix}x\cos\theta-y\sin\theta\\ x\sin\theta+y\cos\theta\end{pmatrix}=R(\theta)\begin{pmatrix}x\\ y\end{pmatrix} \tag{3.32}$$

采用第 3.2.1 节的符号，$R(\theta)$变成 $h(q)$，其中 $q=\theta$，对于诸如式（3.32）所定义的线性变换，列向量表示新坐标系的基向量，$R(\theta)$的列向量是单位向量，它们的内积（或者点积）是 0，表示它们是正交的。假设表示体坐标系的 x 和 y 轴“画”在$\mathcal{A}$上。完成逆时针旋转 θ 后，考虑 x 和 y 变换后相应的方向就可以确定 $R(\theta)$的列，这种阐述可以扩展到更高维旋转矩阵。

旋转围绕原点进行。因此当定义$\mathcal{A}$的模型时，原点应该放置在预期的旋转轴上。采用半代数模型，通过将每个本原进行变换就可以将整个机器人模型进行旋转$\mathcal{A}(\theta)$。也可以对每个本原进行反向旋转 $R(-\theta)$。

平移与旋转的组合　假设进行了 θ 角旋转，然后平移 x_t, y_t。这可以将机器人放置到任何所期望的位置和方向上。需要注意的是平移和旋转是不能交换顺序的。如果运算成功进行，那么每个$(x,y)\in\mathcal{A}$变换到：

$$\begin{pmatrix}x\cos\theta-y\sin\theta+x_t\\ x\sin\theta+y\cos\theta+y_t\end{pmatrix} \tag{3.33}$$

下面的矩阵乘法对前两个向量分量产生相同的结果：

$$\begin{pmatrix}\cos\theta & -\sin\theta & x_t\\ \sin\theta & \cos\theta & y_t\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}x\\ y\\ 1\end{pmatrix}=\begin{pmatrix}x\cos\theta-y\sin\theta+x_t\\ x\sin\theta+y\cos\theta+y_t\\ 1\end{pmatrix} \tag{3.34}$$

这意味 3×3 矩阵：

$$T=\begin{pmatrix}\cos\theta & -\sin\theta & x_t\\ \sin\theta & \cos\theta & y_t\\ 0 & 0 & 1\end{pmatrix} \tag{3.35}$$

表示一个紧接着进行平移的旋转。矩阵 T 称为相似变换矩阵。记住 T 表示一个紧接着进行平移的旋转。每个本原都可以用 T 的逆进行变换，产生一个变换后的机器人实体模型。变换后的机器人用$\mathcal{A}(x_t, y_t, \theta)$表示，在这种情况下有 3 个自由度。相似变换矩阵是组合变换的简便表示；因此它经常在机器人学、力学、计算机图形学等领域得到应用，之所以称为相似的是因为在 $\mathbb{R}^3$ 中它只是没有任何平移的线性变换，在射影几何中很常见的是通过增加一维来包含平移部分[807]。

3.2.3　三维变换

三维空间的刚体变换在概念上与二维情况相似；但是三维情况更加复杂，因为旋转更

加复杂。

三维平移 机器人$\mathcal{A}$平移$x_t, y_t, z_t \in \mathbb{R}$，采用：

$$(x, y, z) = (x + x_t, y + y_t, z + z_t) \tag{3.36}$$

本原：

$$H_i = \{(x, y, z) \in \mathcal{W} | f_i(x, y, z) \leqslant 0\} \tag{3.37}$$

被变换为：

$$\{(x, y, z) \in \mathcal{W} | f_i(x - x_t, y - y_t, z - z_t) \leqslant 0\} \tag{3.38}$$

平移后的机器人用$\mathcal{A}(x_t, y_t, z_t)$表示。

偏航、俯仰、滚动旋转 三维物体可以绕三个正交轴进行旋转，如图 3.8 所示。借用航空术语，三个旋转称为偏航、俯仰和滚动旋转：

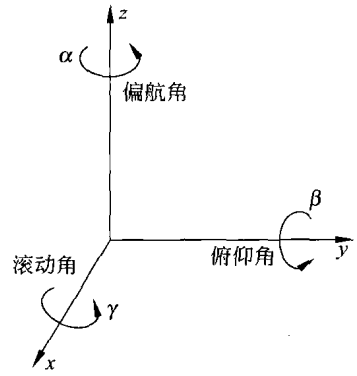


图 3.8 任意三维旋转都可以描述为一个偏航、俯仰、滚动旋转的序列

（1）偏航角是绕 z 轴沿逆时针方向旋转的角度 α。旋转矩阵为：

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.39}$$

$R_z(\alpha)$ 的左上部分形成 x 轴和 y 轴的二维旋转，z 轴保持不变。

（2）俯仰角是绕 y 轴沿逆时针方向旋转的角度 β。旋转矩阵为：

$$R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \tag{3.40}$$

（3）滚动角是绕 x 轴沿逆时针方向旋转的角度 γ。旋转矩阵为：

$$R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix} \tag{3.41}$$

每个旋转矩阵都是二维旋转矩阵式（3.31）的扩展。例如偏航矩阵 $R_z(\alpha)$本质上是完成了一个关于 x 轴和 y 轴的二维旋转，而保持 z 轴保持不变。因此第三行和第三列看起来像单位矩阵的一部分，而 $R_z(\alpha)$的左上部分看起来像一个二维旋转矩阵。

偏航、俯仰和滚动旋转可以将三维物体旋转到任何方向。偏航矩阵、俯仰矩阵和滚动矩阵相乘得到一个矩阵：

$$R(\alpha,\beta,\gamma)=R_z(\alpha)R_y(\beta)R_x(\gamma)$$

$$=\begin{pmatrix}\cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma-\sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma+\sin\alpha\sin\gamma\\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma+\cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma-\cos\alpha\sin\gamma\\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma\end{pmatrix} \tag{3.42}$$

$R(\alpha, \beta, \gamma)$首先进行滚动旋转，然后进行俯仰旋转，最后进行偏航旋转。如果运算顺序发生改变，将产生不同的旋转。当阐述旋转的时候必须非常仔细。考虑最后进行的偏航旋转角 α。将$\mathcal{A}$想象成一架飞机。如果 $\beta=\gamma=0$，那么偏航改变飞机的方向就像将汽车向左转。但是对于任意 β 和 γ 值，z 旋转轴并不垂直于飞机，因为在进行 α 旋转之前飞机并不在通常的方向上。偏航旋转绕世界坐标系的 z 轴旋转，而不是$\mathcal{A}$的体坐标系。每次左乘一个新的旋转矩阵时，并不关心$\mathcal{A}$的原来的体坐标系。只是根据世界坐标系来旋转 R^3 中的每个点。三维旋转依赖于 3 个参数 α，β，γ，而二维旋转仅仅依赖于一个参数 θ，可以用 $R(\alpha, \beta, \gamma)$ 将模型的本原变换到$\mathcal{A}(\alpha, \beta, \gamma)$。

确定偏航角、俯仰角和滚动角　可以直接从给定的旋转矩阵中确定 α, β, γ。假设给定任意旋转矩阵：

$$\begin{pmatrix}r_{11} & r_{12} & r_{13}\\ r_{21} & r_{22} & r_{23}\\ r_{31} & r_{32} & r_{33}\end{pmatrix} \tag{3.43}$$

将其与式（3.42）相应元素对应，就可以求解 α，β，γ 的方程。因为 $r_{21}/r_{11}=\tan\alpha$，$r_{32}/r_{33}=\tan\gamma$。$r_{31}=-\sin\beta$ 和 $\sqrt{r_{32}^2+b_{33}^2}=\cos\beta$。所以：

$$\alpha=\tan^{-1}(r_{11}/r_{21}) \tag{3.44}$$

$$\beta=\tan^{-1}\left(-\sqrt{r_{32}^2+b_{33}^2}/r_{31}\right) \tag{3.45}$$

$$\gamma=\tan^{-1}(r_{32}/r_{33}) \tag{3.46}$$

反正切函数存在选择象限的问题。如何确定正确象限呢？应该通过分子和分母的符号选择象限。分子的符号决定方向是在 y 轴的左边还是右边，分母的符号决定方向是在 x 轴的上边还是下边。这与 C 语言中的 atan2 函数相同，该函数将反正切的范围扩展到[0,2π)。它同样可以应用于式（3.44）、式（3.45）、式（3.46）。

$$\alpha=\text{atan2}(r_{11},r_{21}) \tag{3.47}$$

$$\beta=\text{atan2}\left(\sqrt{r_{32}^2+b_{33}^2},-r_{31}\right) \tag{3.48}$$

$$\gamma=\text{atan2}(r_{32},r_{33}) \tag{3.49}$$

这里假设 $r_{21}\neq 0$ 和 $r_{33}\neq 0$。

三维物体的相似变换矩阵　二维情况下可以定义相似变换矩阵。对于三维情况，一个 4×4 的矩阵可以首先完成 $R(\alpha, \beta, \gamma)$旋转，然后平移 x_t, y_t, z_t，它是：

$$T=\begin{pmatrix}\cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma-\sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma+\sin\alpha\sin\gamma & x_t\\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma+\cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma-\cos\alpha\sin\gamma & y_t\\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & z_t\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.50}$$

再次强调运算的顺序非常重要。矩阵式（3.50）的 T 表示变换的顺序：

1. 滚动 γ　　3. 偏航 α

2. 俯仰 β　　4. 平移 x_t, y_t, z_t

机器人本原的变换产生$\mathcal{A}(x_t, y_t, z_t, \alpha, \beta, \gamma)$。能够进行平移和旋转的三维刚体有六个自由度。

3.3 物体运动链的变换

对由相互连接的刚体链进行变换会变得更加复杂。为方便起见，将每个刚体都作为一个连杆。令$\mathcal{A}_1$, $\mathcal{A}_2$, …, $\mathcal{A}_m$表示 m 个连杆的集合。对于 $1 \leqslant i \leqslant m$，$\mathcal{A}_i$“连接”到$\mathcal{A}_{i+1}$，连接的方式允许$\mathcal{A}_{i+1}$关于$\mathcal{A}_i$做一些受约束的运动。运动约束必须显式给出，这将在稍后讨论。举例说明，假设有一个用钩连接在汽车后的拖车，拖车相对汽车可以旋转。一般情况下，一个相互连接起来的物体的集合称为一个连杆组。本节讨论的物体采用单链的方式进行连接。这产生了一种特殊的连杆组，称为运动链。

3.3.1 二维运动链

在考虑运动链之前，假设$\mathcal{A}_1$和$\mathcal{A}_2$是相互没有连接的刚体，它们每个都能够在$\mathcal{W} = \mathbb{R}^2$中进行平移和旋转。因为它们各自都有三个自由度，所以总共有六个自由度；独立的参数是$x_1, y_1, \theta_1, x_2, y_2, \theta_2$。

物体的连接　在运动链中物体被连接后，自由度将会降低。图 3.9 显示了采用两种不同的方法进行二维连杆的连接。连接部位称为关节。对于旋转关节，一个连杆关于另一个只能进行旋转；对于棱柱形的关节，一个连杆只能沿另一个进行滑动。每种关节从被连接的物体对中减少两个自由度。例如考虑一个连接$\mathcal{A}_1$和$\mathcal{A}_2$的旋转关节。假设$\mathcal{A}_2$体坐标系中的点（0,0）连接到$\mathcal{A}_1$体坐标系中的点(x_a, y_a)。这意味着一旦给定x_a和y_a，那么完全可以确定$\mathcal{A}_2$的平移。注意到x_a和y_a依赖于x_1, y_1, θ_1。这意味着$\mathcal{A}_1$和$\mathcal{A}_2$连接后总共有四个自由度。独立的参数是$x_1, y_1, \theta_1, \theta_2$。本节其余部分的任务是确定当$\mathcal{A}_1, \mathcal{A}_1, \cdots \mathcal{A}_m$，连接成链后如何变换它们的模型，并给出用独立参数表示的表达式。

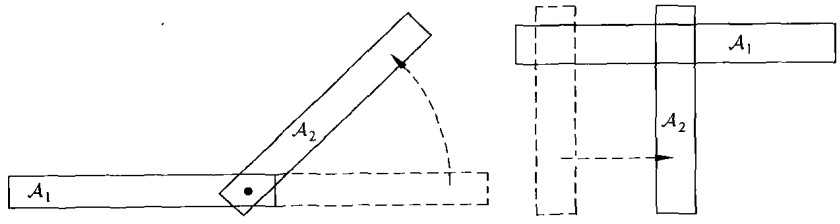


图 3.9　两种类型的二维关节：一个旋转关节允许一个连杆关于另外一个连杆进行旋转，棱柱形关节允许一个连杆关于另外一个连杆进行平移

考虑运动链的情况，其中每个连杆对都是用旋转关节连接起来的。首要任务是指定每个连杆$\mathcal{A}_i$的几何模型。考虑单个刚体，体坐标系的原点用于确定旋转轴。当定义运动链中连杆的模型时，仔细确定体坐标系可以避免额外的复杂性。因为绕旋转关节旋转时，对于

每个 $i>1$，原点将选择在$\mathcal{A}_i$和$\mathcal{A}_{i-1}$之间的关节上。为方便起见，$\mathcal{A}_i$的体坐标系的 x_i 轴被定义为通过两个关节且在$\mathcal{A}_i$上的直线，如图 3.10 所示。对于最后一个连杆$\mathcal{A}_m$，它的原点放置在连接$\mathcal{A}_m$和$\mathcal{A}_{m-1}$的关节上，它的 x_m 轴可以任意放置。第一个连杆$\mathcal{A}_i$的体坐标系可以采用与单个刚体相同的方法进行放置。

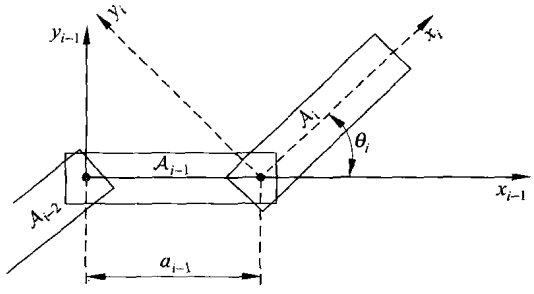


图 3.10　每个$\mathcal{A}_i$的体坐标系是基于将$\mathcal{A}_i$连接到$\mathcal{A}_{i-1}$和$\mathcal{A}_{i+1}$的关节，$1<i<m$

二维链的相似变换矩阵　我们准备确定每个连杆的位置。一个点$(x,y)\in\mathcal{A}_1$在$\mathcal{W}$中的位置用二维相似变换矩阵式（3.35）进行确定：

$$T_1=\begin{pmatrix}\cos\theta_1 & -\sin\theta_1 & x_t\\ \sin\theta_1 & \cos\theta_1 & y_t\\ 0 & 0 & 1\end{pmatrix} \tag{3.51}$$

如图 3.10 所示，令α_{i-1}表示$\mathcal{A}_{i-1}$中关节之间的距离。$\mathcal{A}_i$和$\mathcal{A}_{i-1}$方向差用 θ_i 表示。对于$1<i\leqslant m$，令 T_i 表示连杆$\mathcal{A}_i$的一个 3×3 相似变换矩阵（3.35）：

$$T_i=\begin{pmatrix}\cos\theta_i & -\sin\theta_i & \alpha_{i-1}\\ \sin\theta_i & \cos\theta_i & 0\\ 0 & 0 & 1\end{pmatrix} \tag{3.52}$$

它的变换顺序为：

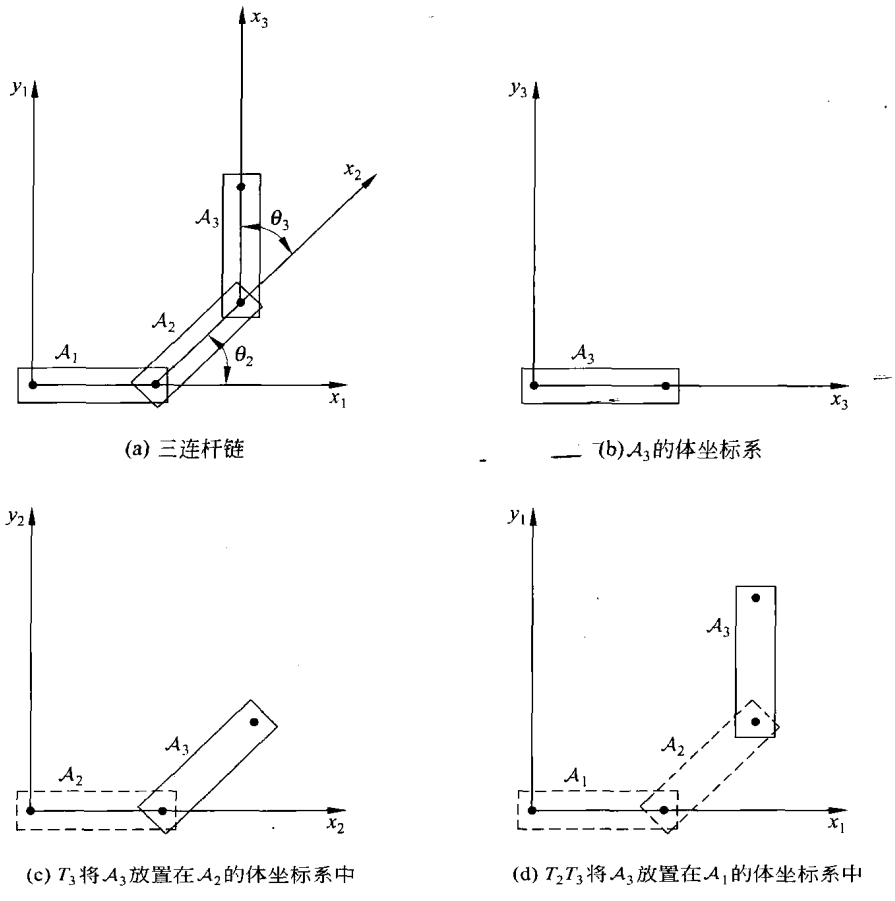
（1）逆时针旋转 θ_i。

（2）沿 x 轴平移 α_{i-1}。

T_i 表达了$\mathcal{A}_i$的体坐标系与$\mathcal{A}_{i-1}$的体坐标系之间的差。T_i 将$\mathcal{A}_i$的体坐标系移动到$\mathcal{A}_{i-1}$的体坐标系。$T_{i-1}T_i$ 将$\mathcal{A}_i$和$\mathcal{A}_{i-1}$移动到$\mathcal{A}_{i-2}$体坐标系。按照这种顺序，任何点$(x,y)\in\mathcal{A}_m$在$\mathcal{W}$中的位置可以用矩阵相乘来得到：

$$T_1T_2\cdots T_m\begin{pmatrix}x\\ y\\ 1\end{pmatrix} \tag{3.53}$$

例 3.3（三连杆的二维链）　为了得到这些变换的直观理解，考虑确定连杆$\mathcal{A}_3$的位形，如图 3.11 所示。图 3.11(a)显示了三连杆的链，其中$\mathcal{A}_1$处在初始位形，其他则各偏移 $\pi/4$。图 3.11(b)显示定义$\mathcal{A}_3$模型的坐标系。T_3 产生旋转 θ_3 和平移 a_2。如图 3.11(c)所示，$\mathcal{A}_3$处在适当的位形中。$\mathcal{A}_2$可以处在其初始位形中，并将连接到$\mathcal{A}_3$。T_2 将$\mathcal{A}_2$和$\mathcal{A}_3$放置到正确位形，$\mathcal{A}_1$处在初始位形。■



(a) 三连杆链　　(b) $\mathcal{A}_3$的体坐标系

(c) T_3将$\mathcal{A}_3$放置在$\mathcal{A}_2$的体坐标系中　　(d) T_2T_3将$\mathcal{A}_3$放置在$\mathcal{A}_1$的体坐标系中

图 3.11　对$\mathcal{A}_3$的模型进行 T_2T_3 变换；如果 T_1 是单位矩阵，那么这就产生$\mathcal{A}_3$中的点在$\mathcal{W}$中的位置

对于旋转关节，参数 a_i 是常数，参数 θ_i 则是可变的。变换后的第 m 个连杆用$\mathcal{A}_m(x_t, y_t, \theta_1, \cdots, \theta_m)$表示。在某些情况下，第一个连杆在参考系中有固定位置。在这种情况下，旋转关节决定所有自由度，产生$\mathcal{A}_m(\theta_1, \cdots, \theta_m)$。对于棱柱形的关节，参数 a_i 是可变的，参数 θ_i 则是固定的。在一个运动链中通常都有这两种关节。

3.3.2　三维运动链

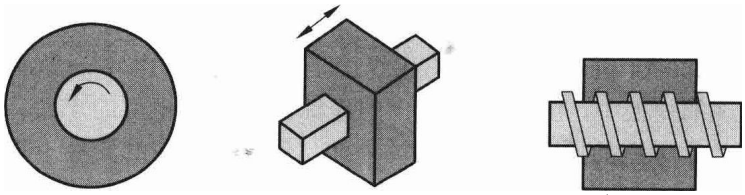
对于单个刚体，三维情况的旋转比二维情况更加复杂，并且可能还有更多种的关节类型，如图 3.12 所示。尽管如此，二维运动链变换的主要思路仍然可以扩展到三维情况。这里仍然利用第 3.3.1 节中介绍的步骤：

（1）仔细选择$\mathcal{A}_i$的体坐标系。

（2）基于关节的关系，测量几个参数。

（3）参数定义一个相似变换矩阵 T_i。

（4）$\mathcal{A}_m$ 中任何点在 $\mathcal{W}$ 中的位置由矩阵 $T_1T_2\cdots T_m$ 给出。



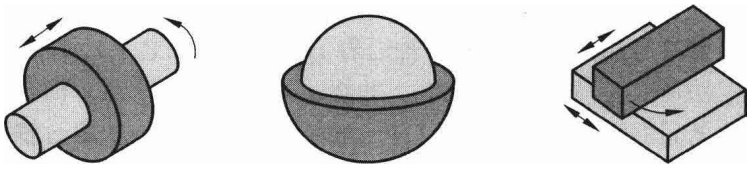


图 3.12　产生于两个物体之间二维表面接触的三维关节类型

考虑 $\mathcal{W}=\mathbb{R}^3$ 中由 m 个连杆组成的运动链，其中 $\mathcal{A}_i$ 通过旋转关节连接到 $\mathcal{A}_{i+1}$ 上。每个连杆可能都是如图 3.13 所示的一个复杂的刚体。对于二维问题，坐标系的选择取决于连接点。对于三维问题，用每个旋转关节的旋转轴是很方便的（这等价于二维情况的连接点）。旋转轴在 $\mathbb{R}^3$ 中通常是一条斜线，如图 3.14 所示。令 z_i 表示连接 $\mathcal{A}_i$ 和 $\mathcal{A}_{i+1}$ 的旋转关节的旋转轴。在每对连续轴之间，令 x_i 轴连接 z_i 和 z_{i+1} 轴间最近的点对，原点在 z_i 上，方向指向

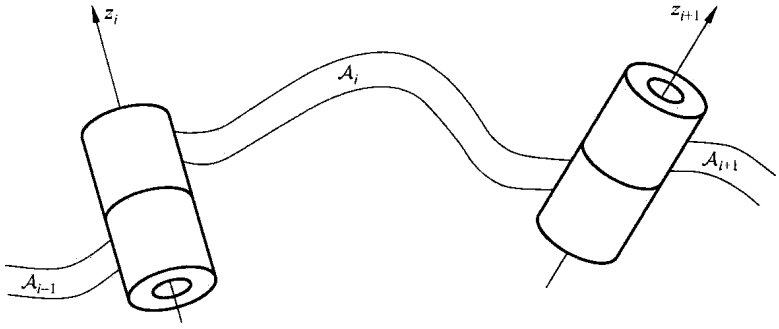


图 3.13　一般连杆的旋转轴都是用旋转关节连接的

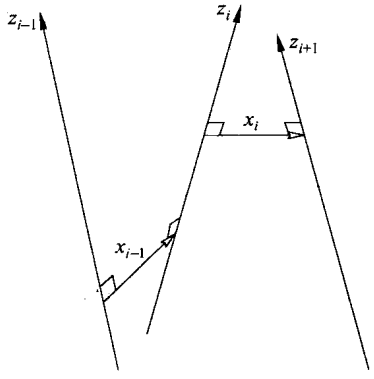


图 3.14　一般连杆的旋转轴都是 $\mathbb{R}^3$ 中的斜线

z_{i+1} 轴的最近点。那么这个轴是唯一的，如果 z_i 和 z_{i+1} 轴不平行。每个 $\mathcal{A}_i$ 的体坐标系相对于 z_i 和 x_i 轴给出，如图 3.14 所示。假设是右手螺旋坐标系，图 3.14 中 y_i 轴的指向远离我们的方向。在即将出现的变换中，由 x_i, y_i, z_i 给出的坐标系便于 $\mathcal{A}_i$ 模型的定义。但是这种方便并不总是出现，因为坐标系的原点可能在 $\mathcal{A}_i$ 的外面，但是所产生的变换矩阵很容易理解。

在第 3.3.1 节中，每个 T_i 由两个参数定义：a_{i-1} 和 θ_i。对三维情况，将定义四个参数：d_i、θ_i、a_{i-1}、α_{i-1}。这些参数称为 Denavit-Hartenberg（DH）参数[437]。图 3.15 指出了这些参数的定义。图 3.15(a)显示了 d_i 的定义。x_{i-1} 和 x_i 轴在两个不同位置接触到 z_i。令 d_i 表示这些接触点之间有符号的距离。如果 x_i 轴沿着 z_i 轴在 x_{i-1} 轴之上，那么 d_i 为正；否则，d_i 为负。参数 θ_i 是 x_i 轴和 x_{i-1} 轴之间的夹角，即绕 z_i 轴旋转 x_{i-1} 轴到 x_i 轴方向的角度。参数 a_i 是 z_i 轴和 x_{i-1} 轴之间的距离；在 $\mathbb{R}^3$ 中这些一般是斜线。参数 α_{i-1} 是 z_i 轴和 z_{i-1} 轴之间的角度。

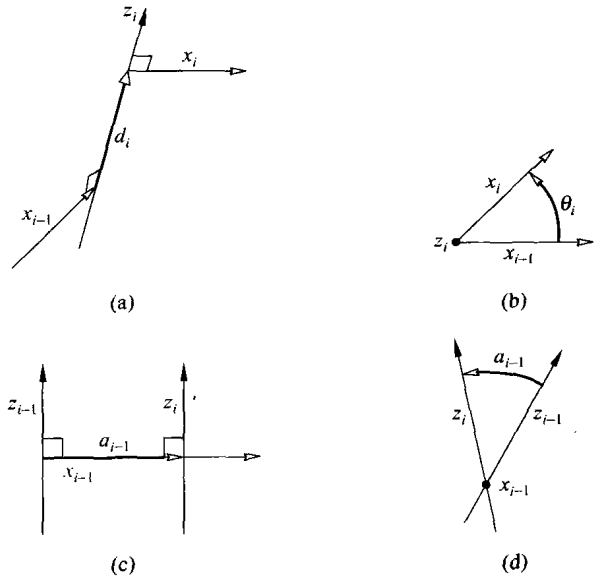


图 3.15　4 个 DH 参数的定义：d_i、θ_i、a_{i-1}、α_{i-1}；分别在（b）和（d）中的 z_{i-1} 轴和 x_{i-1} 轴指向外；任何参数都可能是正的、0 或者负的

双螺旋体　组合两个简单的变换可以构造相似变换矩阵 T_i。变换：

$$R_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.54}$$

产生绕 z_i 轴旋转 θ_i、沿 z_i 轴平移 d_i。旋转和平移的顺序是可以交换的，因为它们都是关于 z_i 轴的运算。关于同一个轴的平移和旋转的组合运算称为螺旋（就像螺钉上螺帽的运动）。R_i 的结果被认为是关于 z_i 轴的螺旋。第二个变换是：

$$Q_{i-1}=\begin{pmatrix}1 & 0 & 0 & a_{i-1}\\ 0 & \cos\alpha_{i-1} & -\sin\alpha_{i-1} & 0\\ 0 & \sin\alpha_{i-1} & \cos\alpha_{i-1} & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.55}$$

它可以被认为是沿 x_{i-1} 轴的螺旋。它绕 x_{i-1} 旋转 α_{i-1} 和沿 x_{i-1} 平移 a_{i-1}。

相似变换矩阵　对于每个 i，$1<i\leqslant i$，变换 T_i 是：

$$T_i=Q_{i-1}R_i=\begin{pmatrix}\cos\theta_i & -\sin\theta_i & 0 & a_{i-1}\\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i\\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.56}$$

它可以被认为是二维变换矩阵（3.52）在三维情况下的对应矩阵。连续完成四步运算：

（1）沿 z_i 轴平移 d_i。

（2）绕 z_i 轴逆时针旋转 θ_i。

（3）沿 x_{i-1} 平移 a_{i-1}。

（4）绕 x_{i-1} 轴逆时针旋转 α_{i-1}。

像二维情况一样，第一个矩阵 T_1 是特殊的。为了表达 $\mathcal{A}_1$ 的任意位置和任意方向，它可以被定义为一般刚体相似变换矩阵（3.50）。如果第一个物体只能绕一个旋转关节进行旋转，那么通常有一个简单的惯例。令 T_1 的参数 a_0, α_0 为 $a_0=\alpha_0=0$。这意味着矩阵（3.55）的 Q_0 是单位阵，那么 $T_1=R_1$。

对于 $i>1$，T_i 给出 $\mathcal{A}_i$ 体坐标系与 $\mathcal{A}_{i-1}$ 体坐标系之间的关系。$\mathcal{A}_m$ 上的点 (x, y, z) 的位置由下式给出：

$$T_1T_2\cdots T_m\begin{pmatrix}x\\ y\\ z\\ 1\end{pmatrix} \tag{3.57}$$

对于每个旋转关节，θ_i 被认为是 T_i 中仅有的变量。棱柱形关节的建模允许 a_i 发生变化。更复杂的关节可以建模为一系列退化关节。例如一个球形关节可以认为是 3 个零长度旋转关节的序列；它们分别完成滚动、俯仰、偏航旋转。对更复杂关节，也可以不采用 DH 表示，而直接形成相似变换矩阵。在保护拓扑特性时这可能是需要的，拓扑特性在第 4 章非常重要。

例 3.4（Puma 560）　这个例子展示了经典机器人操作臂 PUMA 560 的三维链运动学，如图 3.16 所示。目前的参数化是基于文献[37,558]的。步骤是确定相应体坐标系来表示每个连杆。前 3 个连杆允许手（称为末端执行器）在 $\mathcal{W}$ 中进行较大的运动，后 3 个使手执行到期望的方向。这 6 个自由度都产生于旋转关节。体坐标系如图 3.16 所示，相应的 DH 参数由图 3.17 给出。每个变换矩阵 T_i 都是 θ_i 的函数，因此记为 $T_i(\theta_i)$。对本例来说其他参数固定。只允许 $\theta_1, \theta_2, \cdots, \theta_6$ 发生变化。

将图 3.17 给出的参数代入相似变换矩阵后得到：

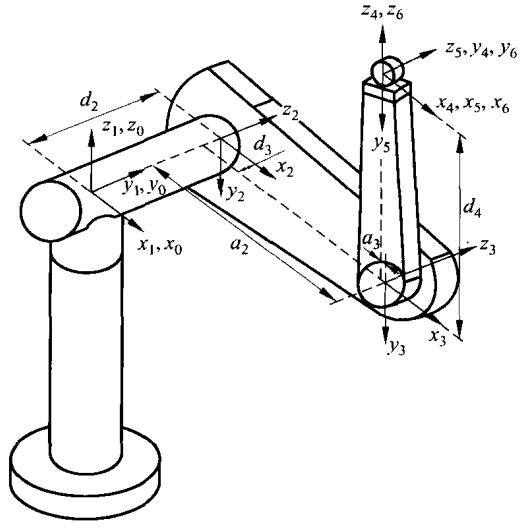


图 3.16 带 DH 参数和链中每个连杆的体坐标系的 Puma560；本图是在作者允许的情况下从文献[558]中借鉴而来的

Matrix	α_{i-1}	a_{i-1}	θ_i	d_i
$T_1(\theta_1)$	0	0	θ_1	0
$T_2(\theta_2)$	$-\pi/2$	0	θ_2	d_2
$T_3(\theta_3)$	0	a_2	θ_3	d_3
$T_4(\theta_4)$	$\pi/2$	a_3	θ_4	d_4
$T_5(\theta_5)$	$-\pi/2$	0	θ_5	0
$T_6(\theta_6)$	$\pi/2$	0	θ_6	0

图 3.17 代入每个相似变换矩阵（3.56）的 DH 参数；注意到，本例中 a_3 和 d_3 是负的（它们是有符号的位移，而不是距离）

$$T_1(\theta_1)=\begin{pmatrix}\cos\theta_1 & -\sin\theta_1 & 0 & 0\\ \sin\theta_1 & \cos\theta_1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.58}$$

$$T_2(\theta_2)=\begin{pmatrix}\cos\theta_2 & -\sin\theta_2 & 0 & 0\\ 0 & 0 & 1 & d_2\\ -\sin\theta_2 & -\cos\theta_2 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.59}$$

$$T_3(\theta_3)=\begin{pmatrix}\cos\theta_3 & -\sin\theta_3 & 0 & a_2\\ \sin\theta_3 & \cos\theta_3 & 0 & 0\\ 0 & 0 & 1 & d_3\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.60}$$

$$T_4(\theta_4)=\begin{pmatrix}\cos\theta_4 & -\sin\theta_4 & 0 & a_3\\ 0 & 0 & -1 & d_4\\ \sin\theta_4 & \cos\theta_4 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.61}$$

$$T_5(\theta_5)=\begin{pmatrix}\cos\theta_5 & -\sin\theta_5 & 0 & 0\\ 0 & 0 & 1 & 0\\ -\sin\theta_5 & -\cos\theta_5 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.62}$$

$$T_6(\theta_6)=\begin{pmatrix}\cos\theta_6 & -\sin\theta_6 & 0 & 0\\ 0 & 0 & -1 & 0\\ \sin\theta_6 & \cos\theta_6 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.63}$$

最后一个连杆$\mathcal{A}_6$的体坐标系中的点(x,y,z)在$\mathcal{W}$中为：

$$T_1(\theta_1)T_2(\theta_2)T_3(\theta_3)T_4(\theta_4)T_5(\theta_5)T_6(\theta_6)\begin{pmatrix}x\\ y\\ z\\ 1\end{pmatrix} \tag{3.64}$$

■

例 3.5（变换辛烷） 图 3.18 显示了一个辛烷分子的球杆模型。每个“球”是一个原子，“杆”代表原子对之间的化学键。有一个含八个碳原子的线性链，化学键存在于链中每个相邻碳原子对之间。还存在大量的氢原子，但这里将忽略它们。每个碳原子对之间的化学键能够扭曲，如图 3.19 所示。研究分子的位形（称为构象）是计算生物学的重要组成部分。假定有七个自由度，每个自由度产生于化学键的扭曲。那么可以应用本节的技术来表示这些变换。

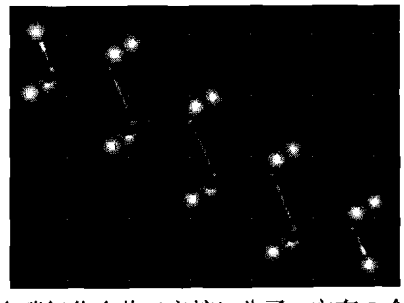


图 3.18　一个碳氢化合物（辛烷）分子，它有 8 个碳原子和 18 个氢原子（本图由纽约大学 MathMol 实验室提供）

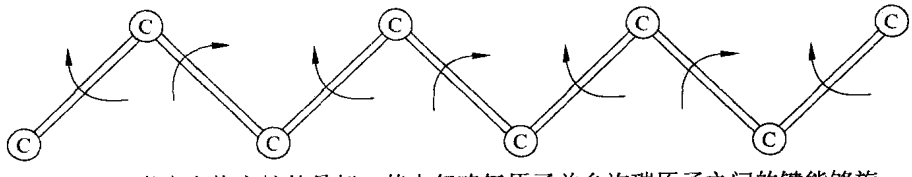


图 3.19　考虑变换辛烷的骨架，其中忽略氢原子并允许碳原子之间的键能够旋转；这很容易用球杆模型来构建（例如 Tinkertoys）；如果第一个连杆固定，那么就有六个自由度。忽略最后一个连杆的旋转

化学键就对应于旋转轴。这就提示 z_i 轴应该与化学键相一致。因为相邻的化学键相会于同一个原子，它们之间没有距离。从图 3.15(c)可以观察到对于所有 i，这使得 $a_i=0$。从

图 3.15(a)可以看到 d_i 对应于化学键的长度，即相邻碳原子间的距离。从图 3.20 看到，有两个角度参数 θ_i 和 α_i。因为连杆仅能绕 z_i 轴旋转，所以相邻轴之间的角度必须保持固定，如图 3.15(d)所示。在化学中这被称为键角，在 DH 参数中用 α_i 表示。剩下的参数 θ_i 是表示自由度的变量。但是从图 3.15(b)可以看出该例子是退化的，因为每个 x_i 轴因 a_i=0 而没有世界坐标系。但这并不会出现问题。为形象化，用 z_{i-1} 和 z_{i+1} 分别代替 x_{i-1} 和 x_i 可能是有益的。容易看见当 z_i 轴的化学键被扭曲时，观察到的角度也会相应发生变化。每个化学键阐述为一个连杆 $\mathcal{A}_i$。$\mathcal{A}_i$ 的原点必须是 z_i 和 z_{i+1} 轴的交点。因此 $\mathcal{A}_i$ 的大多数点将在 $-z_i$ 方向上，如图 3.20 所示。

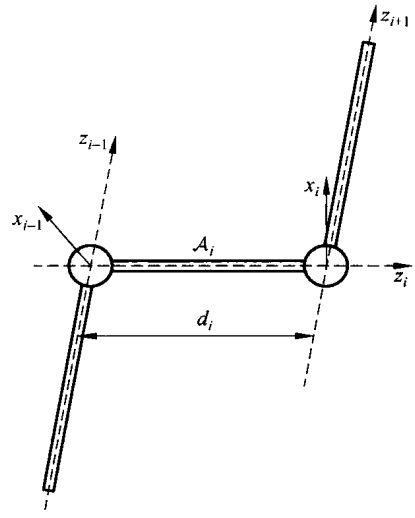


图 3.20 每个键都可以阐述为一个长度为 d_i 的“连杆”，它对准 z_i 轴方向。注意到，大多数 $\mathcal{A}_i$ 在 $-z_i$ 方向上

接下来的任务是写出矩阵。给第一个化学键选择一个世界坐标系，原点在第二个原子上，化学键与 z 轴平行，但方向相反，如图 3.20 所示。为了定义 T_1，回顾矩阵（3.54）的 $T_1=R_1$，因为 Q_0 被删去。参数 d_1 表示沿着 z_1 轴、x_0 轴和 x_1 轴之间的距离。因为没有 x_0 轴，所以可以自由选择 d_1；因此令 d_1=0，可得：

$$T_1(\theta_1)=R_1(\theta_1)=\begin{pmatrix}\cos\theta_1 & -\sin\theta_1 & 0 & 0\\ \sin\theta_1 & \cos\theta_1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.65}$$

T_1 使 $\mathcal{A}_1$ 中的点绕 z_1 轴旋转，这是正确的。

其余六个化学键的矩阵是：

$$T_i(\theta_i)=\begin{pmatrix}\cos\theta_i & -\sin\theta_i & 0 & 0\\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i\\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.66}$$

对于 $i\in\{2,\cdots,7\}$。任意点 $(d,y,z)\in\mathcal{A}_7$ 的位置由下式给出：

$$T_1(\theta_1)T_2(\theta_2)T_3(\theta_3)T_4(\theta_4)T_5(\theta_5)T_6(\theta_6)T_7(\theta_7)\begin{pmatrix}x\\y\\z\\1\end{pmatrix} \tag{3.67}$$

■

3.4　运动树的变换

目的　对一些有趣的问题，连杆组被连接成“树”，如图 3.21(a)所示。假设连杆没有连接成环（见图 3.21(b)）；尽管在本节的结尾对此进行了一些说明，但是环状情况还是将被放在第 4.4 节介绍。关节和四肢连接在躯体上而形成的人体就是一个可以建模为刚体连杆树的例子。膝关节和肘关节可以认为是旋转连接。肩关节是球关节，尽管它不能做到任意定向。在如第 1.4 节中，我们对虚拟环境中人的动画以及开发类人型机器人等方面存在广泛的兴趣。这都依赖于对模拟人体的运动的描述。

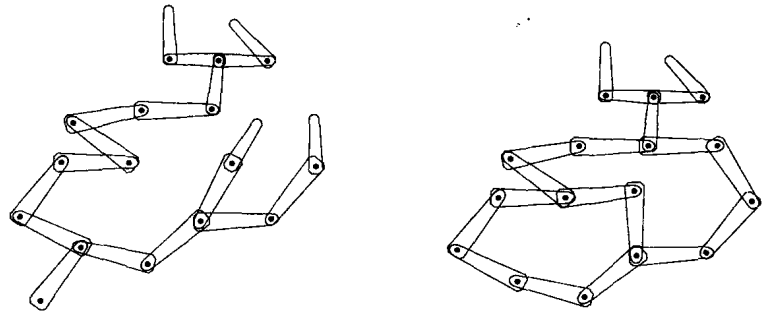


图 3.21　一般连杆组

(a) 考虑的是一个刚体“树”，而不是刚体链；(b) 如果有环，那么参数必须仔细选择以确保环是封闭的

另一个涉及运动树的问题是分子的构象分析。例 3.5 涉及一个单链；但是大多数的组织分子却更加复杂，就像图 1.14(a)所示的药物。化学键的扭曲形成分子的自由度。构造空间中的运动要求对运动树进行描述。研究这些构象非常重要，因为科学家需要确定一些候选药物分子是否能够通过扭曲而很好地嵌入蛋白质腔中（例如需要很低的能量）；这将产生非常希望得到的药理学效果。另一个重要问题是确定复杂的蛋白质分子如何折叠到特定位形。这些分子是比典型药物分子的数量多得多的分子的序列（根据原子的数量和自由度）。更多信息参见第 7.5 节。

$\mathcal{W}=\mathbb{R}^2$ 的公共关节　首先考虑最简单的情形，其中有连杆的一个二维树，每个连杆仅有两点可以进行旋转关节的连接。这与图 3.21(a)相对应。连杆$\mathcal{A}_1$被指定为树的“根”。为了进行树中$\mathcal{A}_i$的变换，将第 3.3.1 节的方法直接应用于$\mathcal{A}_i$到$\mathcal{A}_1$的链，而忽略其他物体。每个连杆给树的总自由度贡献一个θ_i。这种情况似乎很简单；然而遗憾的是在一般情况下，这并不容易。

多于两个旋转轴的联结　现在建模更复杂的连杆。主要的不同在于一个连杆可以有多

个关节，如图 3.22 所示的$\mathcal{A}_7$，一个有多于两个关节的连杆被称为一个联结。

如果只有一个联结，那么选择联结作为根就可以避免联结所带来的大多数复杂性。例如一个简单的类人型模型，躯体就是一个联结。将联结而不是将右脚作为树的根是很有意义的。腿、臂、头都可以建模为独立的链。对每个链我们仅关心的是它的第一个连杆不连接到躯体的相同点。这通过插入连接躯体的原点和四肢的连接点的这样一个固定的、假想的连杆就可以解决。

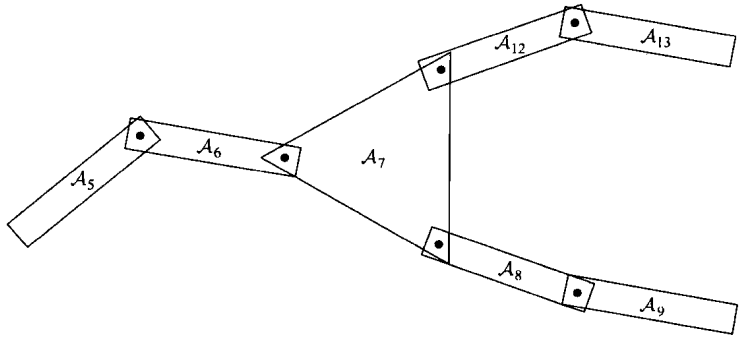


图 3.22　一个连杆可能有多于两个的关节，如$\mathcal{A}_7$

如果有更多的联结，情况会更加有趣。假设图 3.22 表示部分二维连杆系统，该系统的根$\mathcal{A}_1$通过连杆链连接到$\mathcal{A}_5$。为了变换连杆$\mathcal{A}_9$，对点$(x,y)\in\mathcal{A}_9$直接应用第 3.3.1 节的方法进行一系列变换：

$$T_1\cdots T_5T_6T_7T_8T_9\begin{pmatrix}x\\y\\1\end{pmatrix} \tag{3.68}$$

同样地，为了变换T_{13}，应用序列：

$$T_1\cdots T_5T_6T_7T_{12}T_{13}\begin{pmatrix}x\\y\\1\end{pmatrix} \tag{3.69}$$

而不考虑由$\mathcal{A}_8$和$\mathcal{A}_9$形成的链。仔细观察$\mathcal{A}_7$。如图 3.23 所示，它的体坐标系的定义采用了两种不同的方法，一个链采用一种方法。如果采用相同的方法，那么至少其中一个必须放弃第 3.3.1 节中选择坐标系的惯例。这种情况对三维树来说会更糟，因为这将建议放弃 DH 参数。Khalil-Kleinfinger 参数是 DH 参数的扩展，它能够解决坐标系的选择问题[527]。

约束参数　幸运地，对不同的链可以采用不同的坐标系；当然这需要额外的信息。想象对整个树的变换。变量θ_7将出现两次，一次来自于上边的链，一次来自于下边的链。令θ_{7u}和θ_{7l}表示这些θ。真的能够用不同的方法选择θ吗？这意味着树像图 3.24 那样，有两个独立运动的连杆$\mathcal{A}_{7u}$和$\mathcal{A}_{7l}$。对此，必须引入约束条件。假设将θ_{7u}当作一个独立变量。必须选择参数θ_{7u}为$\theta_{7l}+\phi$，其中ϕ如图 3.23 所示。

例 3.6（二维物体树）　图 3.25 显示一个涉及六个连杆的二维例子。为了对$(x,y)\in\mathcal{A}_6$进行变换，相关的连杆仅有$\mathcal{A}_6$、$\mathcal{A}_2$和$\mathcal{A}_1$。变换是：

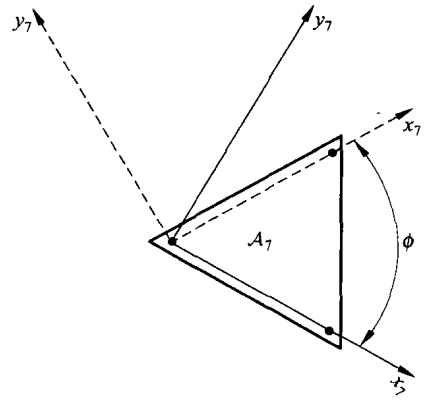


图 3.23　连接点有两个不同的坐标系，这取决于跟踪的是哪个链；变换$\mathcal{A}_9$可以得到实线轴，变换$\mathcal{A}_{13}$得到虚线轴

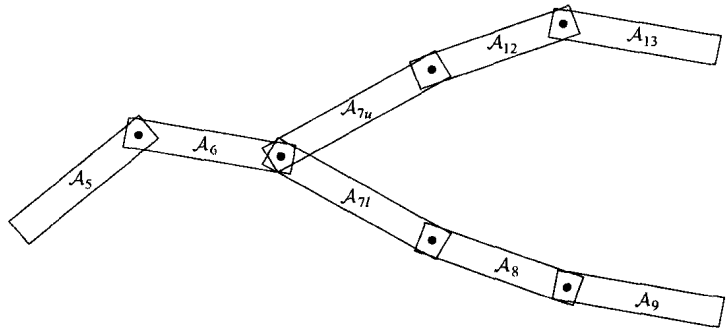


图 3.24　独立地选择每个 θ_7 将会导致一棵树，树忽略了$\mathcal{A}_7$是刚体这个事实

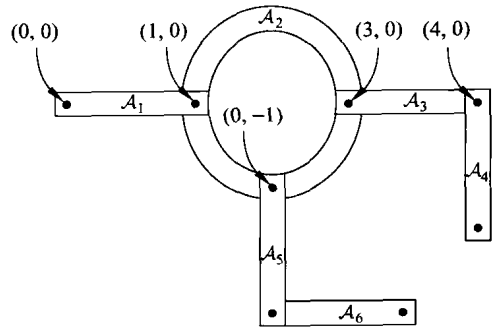


图 3.25　一棵物体树，其中关节连在不同的位置

$$T_1 T_{2l} T_5 T_6 \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.70}$$

其中：

$$T_1 = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & x_t \\ \sin\theta_1 & \cos\theta_1 & y_t \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.71}$$

$$T_{2l}=\begin{pmatrix}\cos\theta_{2l} & -\sin\theta_{2l} & a_1\\ \sin\theta_{2l} & \cos\theta_{2l} & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}\cos\theta_2 & -\sin\theta_2 & 1\\ \sin\theta_2 & \cos\theta_2 & 0\\ 0 & 0 & 1\end{pmatrix} \tag{3.72}$$

$$T_5=\begin{pmatrix}\cos\theta_5 & -\sin\theta_5 & a_2\\ \sin\theta_5 & \cos\theta_5 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}\cos\theta_5 & -\sin\theta_5 & \sqrt{2}\\ \sin\theta_5 & \cos\theta_5 & 0\\ 0 & 0 & 1\end{pmatrix} \tag{3.73}$$

$$T_6=\begin{pmatrix}\cos\theta_6 & -\sin\theta_6 & a_5\\ \sin\theta_6 & \cos\theta_6 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}\cos\theta_6 & -\sin\theta_6 & 1\\ \sin\theta_6 & \cos\theta_6 & 0\\ 0 & 0 & 1\end{pmatrix} \tag{3.74}$$

式（3.72）中的 T_{2l} 表示跟随下边的链。变换 $\mathcal{A}_4$ 中的点：

$$T_1T_{2u}T_4T_5\begin{pmatrix}x\\ y\\ 1\end{pmatrix} \tag{3.75}$$

其中 T_1 与式（3.71）相同，并且：

$$T_3=\begin{pmatrix}\cos\theta_3 & -\sin\theta_3 & a_2\\ \sin\theta_3 & \cos\theta_3 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}\cos\theta_3 & -\sin\theta_3 & \sqrt{2}\\ \sin\theta_3 & \cos\theta_3 & 0\\ 0 & 0 & 1\end{pmatrix} \tag{3.76}$$

$$T_4=\begin{pmatrix}\cos\theta_4 & -\sin\theta_4 & a_4\\ \sin\theta_4 & \cos\theta_4 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}\cos\theta_4 & -\sin\theta_4 & 0\\ \sin\theta_4 & \cos\theta_4 & 0\\ 0 & 0 & 1\end{pmatrix} \tag{3.77}$$

有趣的是：

$$T_{2u}=\begin{pmatrix}\cos\theta_{2u} & -\sin\theta_{2u} & a_1\\ \sin\theta_{2u} & \cos\theta_{2u} & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}\cos(\theta_{2l}+\pi/4) & -\sin(\theta_{2l}+\pi/4) & a_1\\ \sin(\theta_{2l}+\pi/4) & \cos(\theta_{2l}+\pi/4) & 0\\ 0 & 0 & 1\end{pmatrix} \tag{3.78}$$

其中约束条件 $\theta_{2u}=\theta_{2l}+\pi/4$ 使 $\mathcal{A}_2$ 为一个联结。 ■

对于三维物体树，也有相同的一般原则。在某些情况下，并没有涉及对联结和约束的特别考虑。一个例子是柔性分子的变换，因为所有相邻旋转轴相交，并且联结就在这些交点上。一般情况下，DH 参数技术可以应用于每个链，那么必须确定相应约束并用这些约束表示树的真实自由度，Khalil-Kleinfinger 参数能够方便地得到求解结果[527]。

环状情况 最一般的情况是包括连接成环的连杆，如图 3.26 所示。这被称为闭运动链。这在许多应用中出现，例如类人型机器人或者数字化演员，当双足接触地面时就形成了环。作为另外一个例子，假设两个诸如例 3.4 中的 Puma 560 的机器人操作臂协作完成物体的搬运。如果每个机器人用它的手抓住相同的物体，那么就形成了环。一个复杂的例子如图 1.5 所示，其中可移动机器人将移动钢琴。机器人领域之外，大部分组织分子都有柔性环。探索它们的构象空间要求仔细考虑这些环所带来的复杂性。

闭运动链的主要困难是难以确定哪个参数值在其可以接受的范围内来确保封闭。如果给定这些值，那么可以用与树相同的方式处理这些变换。例如：图 3.26 中的连杆可以通过

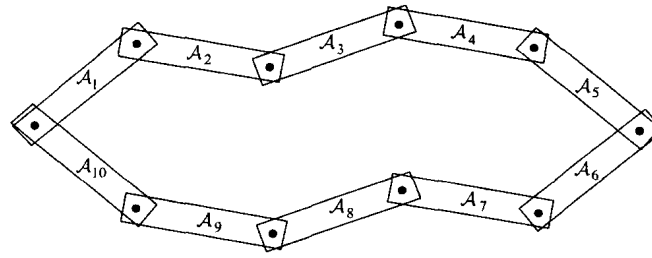


图 3.26　由十个连杆和十个旋转关节连接成一个环；这是闭运动链的一个例子

将环分成两个不同的链来进行变换。首先忽略$\mathcal{A}_5$和$\mathcal{A}_6$之间关节的存在，如图 3.27 所示。考虑两个不同的运动链，它们都开始于最左端的关节。上链从$\mathcal{A}_1$到$\mathcal{A}_5$，下链从$\mathcal{A}_{10}$到$\mathcal{A}_6$。直接应用第 3.3.1 节中的技术可以得到对这些物体的变换。因此物体的变换很容易，但是如何选择参数值来保证$\mathcal{A}_5$和$\mathcal{A}_6$在它们公共关节处连接呢？对于上链，该关节的位置可以表示为：

$$T_1(\theta_1)T_2(\theta_2)T_3(\theta_3)T_4(\theta_4)T_5(\theta_5)\begin{pmatrix} a_5 \\ 0 \\ 1 \end{pmatrix} \tag{3.79}$$

其中$(a_5,0)\in\mathcal{A}_5$是$\mathcal{A}_5$中与$\mathcal{A}_6$连接的关节的位置。该关节的位置也可以用下链表示：

$$T_{10}(\theta_{10})T_9(\theta_9)T_8(\theta_8)T_7(\theta_7)T_6(\theta_6)\begin{pmatrix} a_6 \\ 0 \\ 1 \end{pmatrix} \tag{3.80}$$

其中$(a_6,0)$表示在$\mathcal{A}_6$的体坐标系中该关节的位置。如果不必保持环路，那么$\theta_1,\cdots,\theta_{10}$值可以任意选择，产生 10 个自由度。但是如果必须保持环路，那么式（3.79）和式（3.80）必须相等，即：

$$T_1(\theta_1)T_2(\theta_2)T_3(\theta_3)T_4(\theta_4)T_5(\theta_5)\begin{pmatrix} a_5 \\ 0 \\ 1 \end{pmatrix} = T_{10}(\theta_{10})T_9(\theta_9)T_8(\theta_8)T_7(\theta_7)T_6(\theta_6)\begin{pmatrix} a_6 \\ 0 \\ 1 \end{pmatrix} \tag{3.81}$$

式（3.81）是一个非线性三角方程，该方程的解的集合非常复杂。对于这个例子，自由度是 8，因为公共关节去掉了两个自由度。因为公共关节允许连杆旋转，那么失去两个自由度。如果$\mathcal{A}_5$和$\mathcal{A}_6$之间是刚性连接，那么总自由度仅为 7。对多数涉及环路的问题，不可能得到解集的合适参数表示。这是著名的逆运动问题的一种形式[254,696,778,994]。

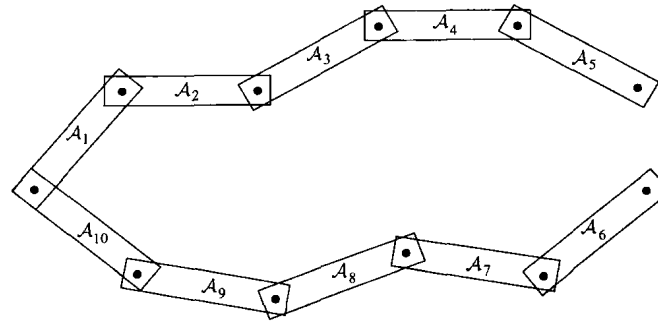


图 3.27　环可能是开的，使得可以应用树形变换；但是必须满足一个封闭约束

一般情况下，连杆的复杂排列可以想象为存在多个环路。每次“忽略”一个环路的关节，那么就少了一个环路，这个过程不断重复直到图中没有环路。所产生的连杆排列就是一棵树，那么就可以应用以前的技术进行处理。但是对于每个被“忽略”的关节，必须引入一个与式（3.81）相似的方程。所有这些方程必须同时满足才不违反原始的环路约束。假设给定值参数集合。例如当用运动捕获技术来测量接触地面的人体每一部分的位置和方向时就会发生这种情况。这样就可以计算解参数，并且所有变换都很容易表示。但是只要模型运动，就很难保证新的变换不违反封闭约束条件。例如数字演员的脚就可能踏入地板。关于这个问题的更多信息可参见第 4.4 节。

3.5 非刚体的变换

我们很容易想象到非刚体运动规划问题。这是本章目前所研究的变换族之外的问题。这里简单介绍几种非刚体变换。

线性变换 旋转是线性变换的特殊形式，在假设变换是在 $\mathbb{R}^n$ 中进行的情况下，它通常用一个 $n\times n$ 矩阵 M 表示。考虑对二维机器人 $\mathcal{A}$ 中的点 (x, y) 进行变换：

$$\begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \tag{3.82}$$

如果 M 是旋转矩阵，那么 $\mathcal{A}$ 的大小和形状将保持不变。但在某些应用中，可能希望扭曲它。通过下式可以实现机器人沿 x 轴放大 m_{11} 倍，而沿 y 轴放大 m_{22} 倍：

$$\begin{pmatrix} m_{11} & 0 \\ 0 & m_{22} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \tag{3.83}$$

m_{11}，m_{22} 均是正实数。如果其中一个为负，那么可以得到 $\mathcal{A}$ 的镜像。除进行比例缩放以外，下式可以使 $\mathcal{A}$ 产生切变：

$$\begin{pmatrix} 1 & m_{12} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \tag{3.84}$$

其中 $m_{12}\neq 0$。$m_{12}=1$ 的情形如图 3.28 所示。

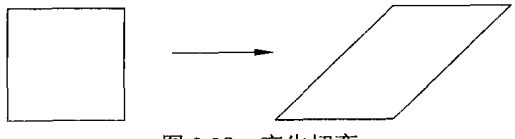


图 3.28 产生切变

比例缩放、切变、旋转等矩阵乘在一起产生一般意义的变换矩阵，该矩阵明确地将每个效果参数化。也可能将矩阵 M 从 $n\times n$ 扩展到 $(n+1)\times(n+1)$ 得到包含平移的相似变换矩阵，并且概念扩展到 $\mathbb{R}^3$ 及以上维数。这使得比例缩放、切变的效果直接融入第 3.2 节～第 3.4 节的概念中。

柔性材料 在一些应用中存在超出线性变换范围的运动。假设弯曲一个柔性材料，诸如使床垫通过一个门。床垫虽然可以近似成二维连杆阵列；但是复杂性和自由度过于庞大。也假设将 100 个旋转关节连接在一起形成一个蛇形机器人。采用第 3.3 节的方法用 100 个旋转参数 $\theta_1,\cdots,\theta_{100}$ 来变换它，但是在规划算法中这会变得难以处理。另外一个思路是将蛇形结构用一条可变形的曲线来近似。

对于类似这些问题，往往可以采用参数化的曲线或曲面族。样条模型是最合适的，因

为它们通过调整较小数量的参数来实现曲线形状的控制。其他模型包括在第 3.1.3 节提到过的广义圆柱和超二次曲面模型。

一个复杂性来源于对可允许参数空间的复杂约束。例如每个蛇形机器人的关节可以有一个小范围的旋转。采用运动链很容易对此进行建模；但是很难确定样条族中哪些样条曲线满足这些额外的约束条件。同样地，操纵柔性材料通常有复杂的材料弹性方面的约束。甚至确定在某些外力作用下材料的形状都需要对材料的弹性能量方程进行积分[580]。

进一步阅读

第 3.1 节只涉及很简单的几何模型。大多数文献的注意力集中在参数曲线和曲面[378,721,791]。这些模型对运动规划来说并不很常见，因为实际应用中最重要的是得到有效的碰撞检测，而在理论上最重要的是拥有隐式代数曲面。文献[457]中全面地介绍了实体和边界表示，其中还包括半代数模型。文献[707,708]介绍了关于半代数模型的理论算法问题。作为二重连通边的列表与其变形的一个比较，可参见文献[525]。

第 3.2 节的内容将出现在几乎任何关于机器人、计算机视觉或者计算机图形学的书籍中，查阅线性代数可能有助于加深对旋转的认识。有许多方法来参数化所有三维旋转矩阵的集合。选择航向-俯仰-滚动这种表示是因为它最容易理解。基于不同的旋转顺序和轴的不同选择，这种表示一般有 12 个不同的变量（也称为欧拉角）。但是这种表示并不适于运动规划算法。它很容易用来制作运动规划输出的快速三维动画，但是它不能为规划算法而准确获取状态空间的结构。第 4.2 节介绍的四元数能够准确获取状态空间的结构；但是它难以用例子进行阐述。因此理解它们这两种情况是非常有益的。除了欧拉角和四元数以外，也还有一些其他的旋转参数化表示，如球坐标、Cayley-Rodrigues 参数化和球极投影等。文献[212]的第 5 章介绍了各种三维旋转和不同参数化的方法。

第 3.3 节中介绍的物体链的变换深受两本经典机器人图书的影响[254,778]。文献[437]介绍了 DH 参数，后来扩展到树和环[527]。除了 DH 参数以外，还可以用指数坐标[728]，它简化了一些计算；但是在建模阶段确定参数可能并不直观。文献[438]介绍了机械学的历史。其他关于运动学的书籍包括文献[29,312,534,692]。在许多机器人学的书[368,858,908,994]的前两章都介绍运动链表示和 DH 参数，然后转到对于控制机器人操作臂来说至关重要的内容上，包括动力学建模、奇异性、操纵性以及控制等。因为本书所关心的是规划算法，所以我们并不像以前那样先介绍动力学，而是仔细研究第 4 章中用到的位形空间。

习　　题

1. 定义一个半代数模型，它从图 3.4 所示的区域中除去一个三角形“鼻子”。

2. 对于完全不同的航向角、俯仰角和滚动角，它们可能产生相同的旋转。也就是说在某些情况下，其中至少 $\alpha \neq \alpha'$、$\beta \neq \beta'$ 或者 $\gamma = \gamma'$ 的情况下，有 $R(\alpha,\beta,\gamma) = R(\alpha',\beta',\gamma')$。刻画发生这种情况的角的集合。

3. 用旋转矩阵，证明二维旋转是可交换的，但三维旋转是不可变换的。

4. 这里考虑第 3.2.3 节航向-俯仰-滚动表示以外的另一种方法。考虑如下旋转的欧拉角表示（有许多其他变形）。第一次旋转是 $R_z(\gamma)$，它只是用矩阵（3.39）中的 γ 代替 α。接下来的两个旋转与航向-俯仰-滚动表示的相同：应用 $R_y(\beta)$，然后应用 $R_z(\alpha)$。这就产生了 $R_{euler}(\alpha,\beta,\gamma)=R_z(\alpha)R_y(\beta)R_z(\gamma)$。

（a）确定矩阵 R_{euler}。

（b）说明 $R_{euler}(\alpha,\beta,\gamma)=R_{euler}(\alpha-\pi,\beta-\pi,\gamma-\pi)$。

（c）假设给定矩阵（3.43）所示的一个旋转矩阵。试解释欧拉角为：

$$\alpha=a\tan 2(r_{23},r_{13}) \tag{3.85}$$

$$\beta=a\tan 2(\sqrt{1-r_{33}^2},r_{33}) \tag{3.86}$$

$$\gamma=a\tan 2(r_{32},-r_{31}) \tag{3.87}$$

5. 根据所使用的轴和这些轴的使用顺序，航向-俯仰-滚动表示有 12 个不同的变量（或者欧拉角）。试确定这些变量，每个变量仅用诸如 $R_z(\alpha)R_y(\beta)R_z(\gamma)$ 这样的符号。简要论述 12 个变量的列表中为什么包括或者不包括旋转的特定组合。

6. 令 $\mathcal{A}$ 为一个单位圆盘，中心在原点，并且 $\mathcal{W}=\mathbb{R}^2$。假设 $\mathcal{A}$ 用单独一个代数本原 $H=\{(x,y)|x^2+y^2\leqslant 1\}$ 来表示。试说明应用任何旋转以后，变换后的本原没有发生改变。

7. 考虑如图 3.29 所示的有关节的物体链。在平面中有 3 个相同的矩形杆 $\mathcal{A}_1$、$\mathcal{A}_2$ 和 $\mathcal{A}_3$。每个杆的宽度为 2，长度为 12。连接点之间的距离为 10。第一个杆 $\mathcal{A}_1$ 连接到原点，第二个杆 $\mathcal{A}_2$ 连接到 $\mathcal{A}_1$，第三个杆 $\mathcal{A}_3$ 连接到 $\mathcal{A}_2$。每个杆可以围绕其连接点旋转。链的位形可以用 3 个角 $(\theta_1,\theta_2,\theta_3)$ 来表示。第一个角 θ_1 表示 $\mathcal{A}_1$ 的两个连接点之间的线段和 x-轴之间的夹角。第二个角 θ_2 表示 $\mathcal{A}_2$ 和 $\mathcal{A}_1$ 之间的角度（当它们平行时，θ_2=0）。第三个角 θ_3 表示 $\mathcal{A}_3$ 和 $\mathcal{A}_2$ 之间的角度。假设位形为 $(\pi/4,\pi/2,-\pi/4)$。

（a）用相似变换矩阵来确定点 a、b 和 c 的位置。

（b）刻画所有位形的集合，对该集合来说最后一个连接点（$\mathcal{A}_3$ 的末端附近）是(0,0)。

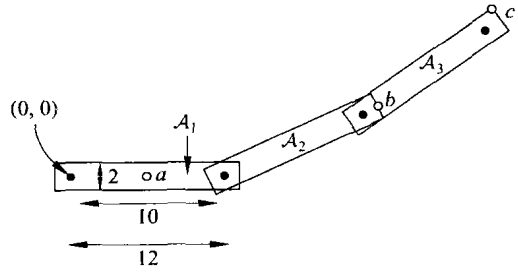


图 3.29　3 个物体的链

8. 如图 3.30 所示的在一个二维世界中运动的一个三连杆链。第一个连杆 $\mathcal{A}_1$ 连接在(0,0)并能够旋转。其余每个连杆都用一个旋转关节连接到下一个连杆上，第二个连杆 $\mathcal{A}_2$ 是一个刚性环，其他两个连杆是矩形杆。

假设该结构处在零位形。假设连杆组运动到位形 $(\theta_1,\theta_2,\theta_3)=\left(\frac{\pi}{4},\frac{\pi}{2},\frac{\pi}{4}\right)$，其中 θ_1 是 $\mathcal{A}_1$

的角度，θ_2 是 $\mathcal{A}_2$ 关于 $\mathcal{A}_1$ 的角度，θ_3 是 $\mathcal{A}_3$ 关于 $\mathcal{A}_2$ 之间的角度。当连杆组处在位形 $\left(\frac{\pi}{4},\frac{\pi}{2},\frac{\pi}{4}\right)$ 时，用相似变换矩阵，计算图 3.30 中(4,0)上点的位置，（该点在 $\mathcal{A}_3$ 上）。

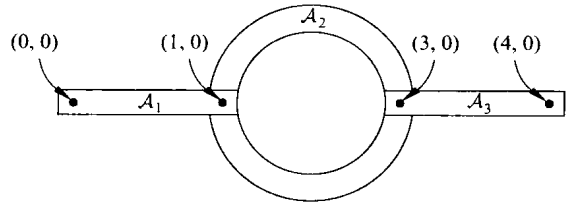


图 3.30　另外一个物体链

9. 将一个球形关节近似为一个有 3 个短的、垂直的连杆的链，它们用旋转关节连接起来并给定变换矩阵的序列。说明当连杆长度接近于零时，所产生的变换矩阵的序列收敛于精确表示一个球形关节的自由度。将这种方法与直接用旋转矩阵（3.42）来表示的关节进行比较。

10. 图 3.12 显示了六个不同的方法，用这些方法，二维曲面相互之间可以进行滑动来产生一个关节。

（a）假设两个物体沿着一条一维曲线相互接触。刻画多种不同类型的“关节”，并指出每种类型的自由度。

（b）假设两个物体在一点上发生接触。指出可能的滚动和滑动的类型以及它们相应的自由度。

11. 假设两个物体形成了一个螺旋关节，其中螺旋中心轴对准第一个物体的 x 轴。确定一个合适的相似变换矩阵代替 DH 矩阵。试用螺旋半径 r、每旋转一圈所前进的位移 d 作为参数来定义矩阵。

12. 回顾例 3.6。应该如何修改变换以使连杆组处在如图 3.25 所示的零位形位置上（每个旋转关节的 $\theta_i=0$，它的角度可以独立进行选择）？

13. 将式（3.84）的切变变换推广到裁剪三维模型。

实现

14. 设计和实现一个二维连杆组的运动模型。使用户能够在平面中展示连杆的排列。

15. 实现没有环的分子运动，并用图将它们显示为一个“球杆”模型。用户应该能够输入原子半径、键连接、键长度和每个键的旋转范围。

16. 设计和实现一个软件系统，其中用户能够交互地连接不同的连杆来制造与用 Tinkertoys 所产生的相似的连杆组（或者另外一种非常常见的构造集合，它允许其中一部分发生运动）。有几个不同长度的杆，它们插入几个硬币状部件的中心孔中和部件边缘的孔中。假设所有关节都是旋转的。应该允许用户改变参数，并看到所有连杆的位置。

17. 将一个人体模型构建为一个三维世界中的连杆树。为简单起见，几何模型可能限定为球形和圆柱形。设计和实现一个系统，它显示虚拟人并允许用户敲打关节使之旋转。

18. 用三维图形为图 3.4 所示的 Puma 560 模型设计一个模拟器。

第4章 位形空间

第3章介绍了物体的建模与变换，为了进行运动规划，还需要定义状态空间。运动规划的状态空间是可能应用于机器人的变换的集合。这称为位形空间（configuration space），它基于拉格朗日力学以及 Lozano-Pérez 所做的开创性的工作[659,663,660]。Lozano-Perezd 在规划中广泛地应用状态空间的概念（Udupa 在碰撞避免工作中也使用了状态空间的思路[947]）。围绕这个概念，Latombe 的书中[591]对运动规划的文献进一步进行了统一。一旦清楚地理解了位形空间，许多以诸如几何学、运动学等各种不同形式出现的运动规划问题都可以采用相同的规划算法加以解决。这种层次的抽象是非常重要的。

本章提出重要的理论基础，它们在第5章～第8章以及在连续状态空间中进行规划的其他地方都非常有用。本章引入的许多概念直接来自于数学，特别是拓扑学。因此第 4.1 节概要介绍拓扑的概念。第 4.2 节使用第3章的概念来定义位形空间。阅读这些内容以后，读者应该能够精确刻画机器人位形空间的特点，并且理解它的结构。在第 4.3 节中，真实世界中的障碍物被变换成位形空间中的障碍物，但是应理解到这种变换是不可能显式构造的。状态空间的隐式表示在规划问题中不断出现。第 4.4 节介绍在第 3.3 节中提及的存在环路的运动链。这种情况如此复杂以至于变换的空间往往也不能明确刻画它的特征（例如参数化）。

4.1 拓扑的基本概念

本节介绍拓扑的基本概念，这些概念对于理解位形空间是有帮助的。拓扑是一个深奥的学科。这里只简要介绍它，以起到抛砖引玉的作用（见本章结尾的“进一步阅读”）。本节的许多内容是提前给出的，所以没有必要理解本节的所有内容；但是如果读者理解得越多，那么对运动规划的理解也就越深刻。

4.1.1 拓扑空间

回忆在实数 $\mathbb{R}$ 集合中的开区间和闭区间的概念。开区间(0,1)包括 0 和 1 之间所有的实数，除 0 和 1 之外。对于每个端点来说，可以定义收敛于它的一个无限序列，例如当 i 趋近无穷时，序列 $\frac{1}{2},\frac{1}{4},\cdots,\frac{1}{2^i}$ 趋近于 0。这意味着可以在(0,1)中选择与 0 或者 1 的距离任意小的点，但是我们不能选择区间边界上的点。对于闭区间，例如[0,1]，则包括边界点。

开集合的概念是拓扑的核心。开集合的定义是开区间概念的推广。该概念应用于某更大空间的子集族。它如此一般以至于很容易涵盖规划中可能遇到的任何种类的位形空间。

集合 X 称为拓扑空间，如果存在 X 的一个称为开集合的子集族满足如下公理：

（1）可数的多个开集合的并集是一个开集合。

（2）有限个的开集合的交集是一个开集合。

（3）X和ϕ是开集合。

在第一个公理中，可以有无限多个开集合的并集，结果必须还是一个开集合。无限多个开集合的交集不一定是开集合。

对于特殊情况$X=\mathbb{R}$，开集合包括开区间。许多不是区间的集合是开集合，因为开区间的并集和交集产生其他开集合。例如集合：

$$\bigcup_{i=1}^{\infty}\left(\frac{1}{3^i},\frac{2}{3^i}\right) \tag{4.1}$$

它是两两不相交的区间的无穷并集，它是开集合。

闭集合 开集合直接出现在拓扑空间的定义中。接下来似乎还需要闭集合。假设X是拓扑空间。一个子集$C\subset X$定义为闭集合，当且仅当$X\backslash C$是一个开集合。因此任何开集合的补集是闭的，任何闭集合的补集是开的。任何诸如[0,1]的闭区间是闭集合，因为它的拓扑$(-\infty,0)\cup(1,\infty)$是开集合。另外，(0,1)也是开集合，因此$\mathbb{R}\setminus(0,1)=(-\infty,0]\cup[1,\infty)$是一个闭集合。在上述表达式中“(”的使用似乎是错误的，但是不能使用“[”，因为$-\infty$和∞不属于$\mathbb{R}$。因此“(”的使用仅仅是符号上的一种权宜之计。

X的所有子集都是开的或者闭的吗？尽管开集合和闭集合从某种意义上是相反的，但是回答是“不”。对于$X=\mathbb{R}$，区间[0.2π)既不是开的，也不是闭的（考虑它的补集[2π, ∞)是闭的，而(−∞,0)是开的）。对于任何拓扑空间，X和ϕ既是开的，又是闭的。

特殊点 从定义和例子中可以看出集合中“边界”或者“边缘”上的点是非常重要的。有几点关于相对于边界的点的内容。令X是一个拓扑空间，U是X的子集，并且令x是X中的任意点。如下几点可获取与U有关的点的位置（见图4.1）：

- 如果存在开集合O_1使得$x\in O_1$和$O_1\subseteq U$，那么称x为U的内点。U中所有内点的集合称为U的内部，记为$\text{int}(U)$。
- 如果存在开集合O_2使$x\in O_2$和$O_2\subseteq X\backslash U$，那么称$x$为$U$的外点。
- 如果x既不是内点，也不是外点，那么它就是U的边界点。X中所有边界点的集合称为U的边界，记为∂U。
- 所有点$x\in X$必须是以上三种情况之一；但是经常还会用到另外一种情况，即使这样是冗余的。如果x是一个内点，或者是一个边界点，那么称它为U的极限点（或者聚点）。U的所有极限点的集合是闭集合，称为U的闭包，记为$cl(U)$。有$cl(U)=\text{int}(U)\cup\partial U$。

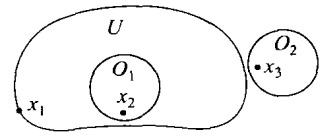


图4.1 阐述边界的定义：假设$X=\mathbb{R}^2$，并且U是一个子集；那么有三种类型的点：（1）x_1是一个边界点，（2）x_2是一个内点，（3）x_3是一个外点；x_1和x_2都是U的极限点

对于$X=\mathbb{R}$的情况，边界点是区间的端点。例如0和1是(0,1)、[0,1]、[0,1)、(0,1]的

边界点。因此 U 可能包括它的边界点，也可能不包括。(0,1)中的所有点是内点，[0,1]中所有点是极限点。命名“极限点”的目的是源于这样的点可能是 U 中点的一个无穷序列的极限。例如，0 是 $1/2^i$ 产生的点列的极限点，其中 $i \in \mathbb{N}$ 是自然数。

以上定义产生几个结果。一个闭集合 C 包含其任何子集序列的极限点。这意味着它包含所有边界点。闭包 cl 总产生一个闭集合，因为它将所有边界点添加到集合里。另一方面，一个开集合不包含任何边界点。在运动规划的位形空间中考虑障碍物时，这些阐述是有用的。

一些例子 用拓扑空间的定义可以构造各种各样的拓扑空间。

例 4.1（$\mathbb{R}^n$ 的拓扑） 我们预料对于任何整数 n，$X = \mathbb{R}^n$ 是一个拓扑空间。这要求描述开集合。一个开球（open ball）$B(x, \rho)$是半径为 ρ、中心在 x 的球的内部点的集合。因此：

$$B(x,\rho) = \left\{ x' \in \mathbb{R}^n \mid \|x' - x\| < \rho \right\} \tag{4.2}$$

其中 $\|\cdot\|$ 表示变量的欧氏范数。开球是 $\mathbb{R}^n$ 中的开集合，并且所有其他开集合都可以表示为一个可数个开球的并集[1]。对于 $\mathbb{R}$，这就将任何开集合表示为区间的并集。

即使可能将 $\mathbb{R}^n$ 的开集合表示为球的并集，但我们还是愿意采用其他的表示方法，因为认识到在需要的时候我们可以返回到开球。所以可以采用第 3.1 节中的本原来产生一些有趣的开集合和闭集合。例如任何以 $H = \left\{ x \in \mathbb{R}^n \mid f(x) \leqslant 0 \right\}$ 的形式表示的代数本原都可以产生一个闭集合。这些本原的有限的并集和交集将产生更多的闭集合。因此第 3.1.1 节和第 3.1.2 节中所有的模型都可以产生一个是闭集合的障碍区域 $\mathcal{O}$。在第 3.1.2 节中曾经提到，仅从用“<”关系的本原所构造的集合是开的。 ■

例 4.2（子空间拓扑） 从拓扑空间的子集可以构造一个新的拓扑空间。令 X 是拓扑空间，$Y \subset X$ 是一个子集。对于某些开集合 $U \subseteq X$，能够表示为 $U \cap Y$ 的 Y 的每个子集所构成的开集合就是 Y 上的子空间拓扑。因此 Y 的开集合与 X 的开集合几乎相同，除了剔除了那些不在 Y 中的点以外。$\mathbb{R}^n$ 的开集合与半代数模型定义的复杂区域的交集可以产生新的子空间。这产生许多有趣的拓扑空间，其中一些将在本章的后面出现。 ■

例 4.3（平凡拓扑） 对于任何集合 X，从它总可以构造一个拓扑空间的平凡例子。X 和 ϕ 是仅有的开集合。所有关于拓扑的公理都是满足的。 ■

例 4.4（奇异拓扑） 拓扑的定义所允许的一般性几乎达到荒谬的程度。可以为任何集合定义一个拓扑空间，只要所声明的开集合服从拓扑公理。假设一个有四个元素的集合为：

$$X = \{CAT, DOG, TREE, HOUSE\} \tag{4.3}$$

除了 ϕ 和 X，假设{CAT}和{DOG}是开集合。根据公理，{CAT, DOG}必须也是开集合。一旦定义了开集合，闭集合和边界点就可以从这个拓扑中产生。 ■

从最后一个例子中，我们可以看出拓扑空间非常具有一般性，几乎不需要做任何进一步的介绍。在拓扑分析中所考虑的大多数空间都满足更多公理。对于在本书中出现的 $\mathbb{R}^n$ 和任何位形空间，满足：

Hausdorff 公理 对于任何不同的 $x_1, x_2 \in X$，存在开集合 O_1 和 O_2 使 $x_1 \in O_1$，$x_2 \in O_2$，以及 $O_1 \cap O_1 = \phi$。

那就是说，有可能将 x_1 和 x_2 分割为相互不重叠的开集合。考虑如何通过选择足够小的

1 球的这种族通常被称为一个基。

开球来实现 $\mathbb{R}^n$ 的分割。任何满足 Hausdorff 公理的拓扑空间 X 被称为 Hausdorff 空间。第 4.1.2 节介绍的流形恰好是 Hausdorff 空间，并且它足够一般化以至于能够反映大多数位形空间。我们不需要考虑不是 Hausdorff 空间的拓扑空间。

连续函数 关于拓扑空间，有连续性的一个简单定义。它是标准微积分定义的推广。令 $f: X \to Y$ 表示拓扑空间 X 和 Y 之间的一个函数，对于任何集合 $B \subseteq Y$，令 B 的原像表示和定义为：

$$f^{-1}(B) = \{x \in X \mid f(x) \in B\} \tag{4.4}$$

这个定义不要求 f 有反函数。

函数 f 是连续的，如果对于每个开集合 $O \subseteq Y$，$f^{-1}(O)$ 是一个开集合。连续性的定义将大大简化我们的分析。例如任何连续函数的代数运算是连续的，为了显示这个结论，仅要求一个论据：任何开集合的原像的原像总是产生一个开集合。将它与那些要求多个 δ 和 ε 的烦琐的经典证明相比较。这个概念如此一般，所以连续函数可以定义在例 4.4 这个荒谬的拓扑空间上。

同胚 你可能听见过这种表达：对一个拓扑学家来说，油炸圈饼和咖啡杯是一样的。在许多数学分支中，都需要定义两个基本对象在什么时候是等价的。在图论（群论）中，这种等价关系被称为同构。在拓扑中，最基本的等价关系是同胚，在大多数其他学科中非常不同的空间，在拓扑中却是等价的。一个油炸圈饼和一个咖啡杯的表面可以认为是等价的，因为两者都有一个唯一的孔。这个概念还需要进行精确定义。

假设 $f: X \to Y$ 是拓扑空间 X 和 Y 之间的一个双射函数。因为 f 是双射函数，那么其反函数 f^{-1} 存在。如果 f 和 f^{-1} 都是连续的，那么 f 称为一个同胚。如果在两个拓扑空间 X 和 Y 存在一个同胚，那么它们是同胚的，记为 $X \cong Y$。这表示拓扑空间的集合上的等价关系（同胚含有自反的、对称的、可传递等特征）。

例 4.5（区间同胚） $\mathbb{R}$ 的任何开区间与任何其他开区间是同胚的。例如(0,1)通过连续映射 $x \mapsto 5x$ 可以映射到(0,5)。(0,1)和(0,5)在这里都是 $\mathbb{R}$ 的拓扑子空间。这种同胚可以用线性代数加以推广。如果子集合 $X \subset \mathbb{R}^n$ 能够通过非奇异线性变换映射到另一个子集合 $Y \subset \mathbb{R}^n$，那么 X 和 Y 是同胚的。例如第 3 章的刚体变换就是同胚应用于机器人的例子。因此机器人平移或者旋转时，它的拓扑并不发生变化。这个例子中机器人本身就是拓扑空间，本章其余部分并不是这种情况。

当混合闭集合和开集合时需要小心。空间[0,1]与(0,1)并不同胚，它们与[0,1)也不同胚。当试图构造一个连续双射函数时，端点会产生问题。但奇怪的是，一个有界集合和一个无界集合可能是同胚的。$\mathbb{R}^n$ 的子集 X 是有界的，如果存在一个球 $B \subset \mathbb{R}^n$ 使 $X \subset B$。映射 $x \mapsto 1/x$ 使(0,1)和$(1, \infty)$同胚。映射 $x \mapsto \tan^{-1}(\pi x/2)$ 使$(-1,1)$和整个 $\mathbb{R}$ 同胚。 ■

例 4.6（拓扑图） 令 X 表示拓扑空间。以前的例子可以扩展使同胚看起来像图同构。令拓扑图[1]是一个图，它的每个凸顶点与 X 中的点相对应，每条边对应于一个连续双射单射函数：τ: [0,1]→X。τ 的像连接 X 中的对应于边的端点（顶点）的点。不同边函数的像不允许相交，除了在顶点处相交以外。从图论中我们知道：两个图 $G_1(V_1, E_1)$和 $G_2(V_2, E_2)$称为同构的，如果存在双射 $f: V_1 \to V_2$ 使得 G_1 的 v_1 和 v_1' 之间存在一条边，当且仅当 G_2 的 $f(v_1)$

1 在拓扑学中，这是一个 1-复形[441]。

和 $f(v_1')$ 之间存在一条边。

在图的同构中用到的双射可以产生一个同胚。E_1 中的每条边被映射到 E_2 中相对应的边。映射在顶点处完全一致。现在读者应该发现两个拓扑图是同胚的，如果在图论标准定义下它们是同构的[1]。如果图不是同构的，那会怎样？仍然有拓扑图是同胚的可能性，如图 4.2 所示。问题是图中出现了“无用”的顶点。删除不影响图的连通性的自由度为 2 的点，问题就解决了。在这种情况下，不同构的图产生不同胚的拓扑图。这样可以构造许多独特的、有趣的拓扑空间。有一些如图 4.3 所示。 ■

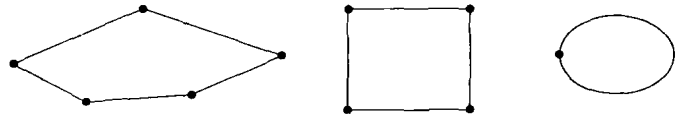


图 4.2 即使图不是同构的，但是由于有无用顶点的缘故，相应的拓扑空间也可能是同胚的；这个例子的图映射到 $\mathbb{R}^2$，它们都与圆是同胚的

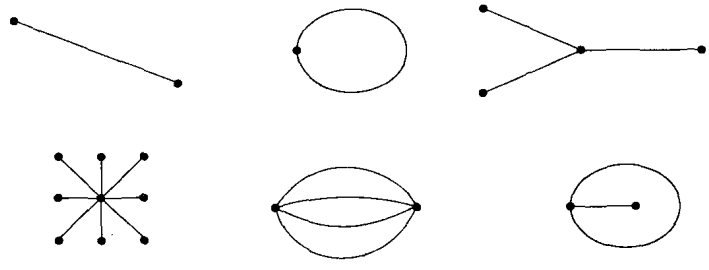


图 4.3 这些拓扑图映射到 $\mathbb{R}^2$ 的相互不同胚的子集

4.1.2 流形

运动规划中着力于保证所产生的位形空间具有良好的反映变换空间真实结构的特性。一个重要的拓扑空间称为流形，该拓扑空间非常普通，它包括第 II 部分所考虑的大多数位形空间。直觉地，流形可以认为是一个“好”的拓扑空间，它在每个点处的行为就像一个曲面。

流形的定义 一个拓扑空间 $M \subset \mathbb{R}^m$ 是一个流形[2]，如果对于每个 $x \in X$，存在一个开集合 $O \subset M$ 使：（1）$x \in O$；（2）O 与 $\mathbb{R}^n$ 同胚；（3）对于所有 $x \in M$，n 是固定的。那么称 n 为流形 M 的维数。第二个条件是最重要的。它指出，在任何点 $x \in M$ 的附近，空间的行为就像它在任何点 $y \in \mathbb{R}^n$ 的附近一样；直觉地，能够运动的方向的集合看起来和其中一个方向相同。图 4.4 显示几个简单的例子，它们可能是流形，也可能不是。

定义的自然结果是 $m \geqslant n$。按照 Whitney 的嵌入定理[452]，$m \leqslant 2n+1$。也就是说，$\mathbb{R}^{2n+1}$

1 从技术上讲，拓扑图是 X 的子集，它们的像是同胚的，而不是它们本身。

2 也可以定义不是 $\mathbb{R}^m$ 的子集的流形。这要求 M 是 Housdorff 空间且是第二可数的，这意味着有可数个开集，它们通过并集可以构造任意其他开集。当 $M \subseteq \mathbb{R}^m$，这些条件是自动满足的；因此，它避免了这些所产生的复杂性，并且满足我们的目的。有些作者用“流形”来表示“平滑流形”。这要求对平滑结构进行定义，并且同胚用微分同胚代替。这里不需求这种额外的结构，但在第 8.3 节中需要它们的时候我们将进行介绍。

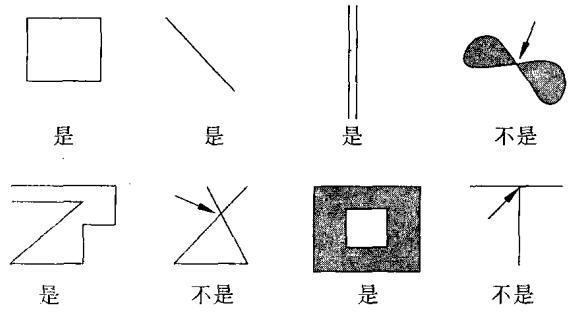


图 4.4 $\mathbb{R}^2$ 的一些开子集，它们可能是流形，也可能不是；对于不是流形的三个来说，妨碍它们成为流动的点被指示出来

“足够大”，它包含任何 n 维流形[1]。即 n 维流形 M 嵌入 $\mathbb{R}^m$ 中，这意味着存在一个从 M 到 $\mathbb{R}^m$ 的单射（如果不是单射，那么 M 拓扑可能发生改变）。

对一个流形来说，不可能包含它的边界点，因为这些点不包含在开集合中。一个有边界的流形的定义要求 M 的每个边界点的邻域与维数为 n 的半空间同胚（在第 3.1 节对 $n=2$ 和 $n=3$ 这两种情况进行了定义），并且要求内点必须与 $\mathbb{R}^n$ 同胚。

下面提出在运动规划中经常出现的构造流形的方法。两个流形是等价的，如果它们是同胚的，记住这一点是非常重要的（回顾油炸圈饼和咖啡杯）。

笛卡儿乘法 从一个拓扑空间很容易构造一个新的拓扑空间。假设 X 和 Y 是拓扑空间，笛卡儿乘法 $X\times Y$ 定义了一个新的拓扑空间。每个 $x\in X$ 和 $y\in Y$ 在 $X\times Y$ 中产生一个点 (x,y)。$X\times Y$ 中的每个开集合都是由 X 中的一个开集合与 Y 中的一个开集合的笛卡儿积产生的。对于每个来自于 X 和 Y 的开集合对来说，都对应于 $X\times Y$ 中的一个开集合。$X\times Y$ 中不会出现其他的开集合；因此它的开集合将被自动确定。

一个熟知的笛卡儿积的例子是 $\mathbb{R}\times\mathbb{R}$，它等价于 $\mathbb{R}^2$。一般情况下，$\mathbb{R}^n$ 等价于 $\mathbb{R}\times\mathbb{R}^{n-1}$。笛卡儿积可以用来代替许多空间。例如 $\mathbb{R}\times\mathbb{R}\times\cdots\times\mathbb{R}=\mathbb{R}^n$。后面将用笛卡儿积来构造一些重要的流形。

一维流形 实数集合 $\mathbb{R}$ 很明显是一维流形，因为 $\mathbb{R}$（通过同胚）看起来当然像每个点附近的 $\mathbb{R}$。将范围限制在单位区间就产生流形 $(0,1)$，因为它们是同胚的（回顾例 4.5）。

另外一个一维流形是一个圆 $\mathbb{S}^1$，它不与 $(0,1)$ 同胚。在这种情况下，$\mathbb{R}^m=\mathbb{R}^2$，令：

$$\mathbb{S}^1=\left\{(x,y)\in\mathbb{R}^2 \mid x^2+y^2=1\right\} \tag{4.5}$$

对于拓扑学家来说，这个特殊的圆并不重要，因为有许许多多的方法定义与 $\mathbb{S}^1$ 同胚的流形。对于与 $\mathbb{S}^1$ 同胚的任何流形，我们有时说流形是 $\mathbb{S}^1$，只是用不同的方式表达而已。$\mathbb{S}^1$ 将被称为一个圆，但是这仅具有拓扑意义；它只需要与我们在高中几何中所学的圆同胚。当考虑 $\mathbb{R}$ 时，我们可以用 $(0,1)$ 来代替，这没有任何问题。流形的另外一个表示可以认为是改变参数化，将在第 8.3.2 节介绍。

等同 表示 $\mathbb{S}^1$ 的一个简便的方法是由等同（identification）得到的，它声明空间中的一

1 定理的一种变形是对于平滑流形，$\mathbb{R}^{2n}$ 就足够了。这个界限是紧的，因为 $\mathbb{RP}^n$（n 维射影空间，本节将会介绍它）不能嵌入 $\mathbb{R}^{2n-1}$。

些点是等同的，即使它们开始是截然不同的[1]。对拓扑空间 X，令 $X/\sim$表示通过某种形式等同后重新定义的 X。X 的开集合被重新定义。等同后 $\mathbb{S}^1$ 可以定义为[0,1]/～，其中声明 0 和 1 是等价的，记为 0～1。这具有将区间的端点“黏合”在一起的效果，从而形成一个闭环。为了理解这种同胚，利用极坐标得到 $\theta \mapsto (\cos 2\pi\theta, \sin 2\pi\theta)$。我们应该非常熟悉，极坐标中 0 和 2π 是相同点；这里它们只是归一化为 0 和 1。令 θ 从 0 到 1，那么“环绕”0 是表示 $\mathbb{S}^1$ 的一个简便方法，因为这不需要像式（4.5）那样弯曲。

等同这个概念看起来具有欺骗性，因为流形的定义要求它必须是 $\mathbb{R}^m$ 的子集。这不是问题，因为 Whitney 定理阐述到：任何 n 维流形都可以嵌入 $\mathbb{R}^{2n+1}$。等同只是减少了形象化所需要的维数。在运动规划算法的实现中应用等同也是非常方便的。

二维流形 将笛卡儿积应用于一维流形就可以定义一些重要的二维流形。二维流形 $\mathbb{R}^2$ 由 $\mathbb{R}\times\mathbb{R}$ 产生。乘积 $\mathbb{R}\times\mathbb{S}^1$ 定义了一个等价于无限圆柱面的流形。乘积 $\mathbb{S}^1\times\mathbb{S}^1$ 定义了一个等价于一个环面的流形（油炸圈饼的表面）。

是否可以定义任意其他二维流形？如图 4.5 所示，用等同方法可以产生新的流形。从一个与 $\mathbb{R}^2$ 同胚的开正方形 $M=(0,1)\times(0,1)$开始。令（x, y）表示平面内的一个点。对所有 $y\in(0,1)$，等同（0, y）～（1, y），并将这些点加入 M，这样就可以产生一个平圆柱。这个结果如图 4.5 所示，发生等同的地方已用箭头指示。

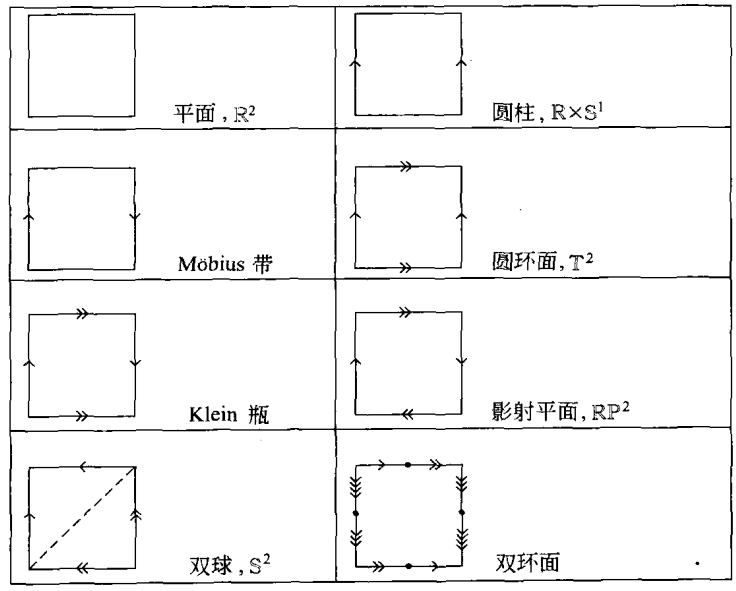


图 4.5 一些二维流形，它们是通过将沿着一个正方形区域的边界的点对等同起来而得到的

将一条纸带的一个短边扭转 180°，然后将两端连接起来就可以构造一个 Möbius 带。它与圆柱面不同胚。也可以将扭转等同起来构造 Möbius 带，就像对于 $y\in(0,1)$，使（0, y）～（1,1–y）。在这种情况下，箭头方向相反。Möbius 带有一个著名的特性：单侧的（用铅笔跟踪纸带，你将到达纸带的两侧），并且是不可定向的（如果试图在平面上画它，如果不使用等同这个技巧，那么它将总是有一个扭转）。

1 这通常被更正式地进行定义，称为“商拓扑”。

所有这些情况都有一个边界。接下来的几个流形将不再有边界，即使它们可能是有界的。如果你处在它们任何一个中间，那意味着你将永远沿着任何轨迹行走，且不会遇到边界。这似乎就像我们的物理世界是无边界的，但是这仅仅是一个错觉。对这个世界来说，存在多个明显的互不同胚的可能性。在更高维数，这种可能性是宇宙论的范畴，它是采用拓扑来刻画宇宙结构的天体物理学的一个分支。

通过等同$(0, y)\sim(1,y)$就可以构造一个圆柱面，再用$(0, x)\sim(x,1)$将上面和下面等同起来就构成一个圆环面。注意到必须包含点(0,0)，并将(0,0)与其他 3 点等同起来。图 4.5 用双箭头指示上面和下面的等同。所有的等同点都必须加入 M。这里没有扭转。所产生的圆环面就像 20 世纪 80 年代的太空视频游戏中出现的宇宙。飞船从一个方向飞离屏幕，又从其他某个地方出现，就像等同所描述的那样。

增加扭转可以产生两个有趣的流形。构造圆环所进行的等同就像构造 Möbius 带一样，只是侧面不扭转而已。这产生的流形称为 Klein 瓶，它可以嵌入 $\mathbb{R}^4$，就像一个闭二维曲面的内部与外部是相同的！从某种意义上，这与 Möbius 带相似。现在假设在两侧、上面和下面都存在扭转。这导致非常奇异的流形：真正的射影平面 $\mathbb{RP}^2$。该空间等价于 $\mathbb{R}^3$ 中通过原点的所有直线的集合。对运动规划来说，三维 $\mathbb{RP}^3$ 是最重要的流形之一。

令 $\mathbb{S}^2$ 表示单位球，其定义：

$$\mathbb{S}^2 = \left\{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 + z^2 = 1\right\} \tag{4.6}$$

其他表示 $\mathbb{S}^2$ 的方法是进行图 4.5 的最后一行中所示的等同。一条点划线指示赤道可能出现的地方，如果我们想给地球造一堵歪扭的墙。地极在左上到右下角。图 4.5 所示最后一个例子是双环面，它是双孔圆环的面。

高维流形 二维流形的构造技术可以扩展到高维情形。$\mathbb{R}^n$ 是一个 n 维流形。$\mathbb{S}^1$ 的 n 个复制的笛卡儿积可以构造 n 维圆环面 $\mathbb{T}^n$。注意到 $\mathbb{S}^1 \times \mathbb{S}^1 \neq \mathbb{S}^2$。因此 $\mathbb{T}^n$ 用 $\left(\mathbb{S}^1\right)^n$ 表示。对于正整数 i 和 j，且 $i+j=n$，笛卡儿积 $\mathbb{R}^i \times \mathbb{T}^j$ 可以构造不同类型的 n 维圆柱面。更高维数的球可以定义为：

$$\mathbb{S}^n = \left\{(x \in \mathbb{R}^{n+1} \mid \|x\| = 1\right\} \tag{4.7}$$

其中 $\|x\|$ 表示 x 的欧氏范数，n 是正整数。通过等同方体$(0,1)^n$的面可以构造许多有趣的空间（或者多面体的面），特别是如果允许不同类型的扭曲。用这种方法可以定义一个 n 维射影空间。例如 Lens 空间是一个可以通过等同多面体的面来构造的流形簇[837]。

由于在运动规划中的重要性，关于流形的更多详细内容在射影空间上给出。n 维实数射影空间 $\mathbb{RP}^n$ 的定义是 $\mathbb{R}^{n+1}$ 中经过原点的所有直线的集合。每条直线作为 $\mathbb{RP}^n$ 中的一个点。采用式（4.7）中 $\mathbb{S}^n$ 的定义，注意到 $\mathbb{R}^{n+1}$ 中每条这样的直线与 $\mathbb{S}^n \subset \mathbb{R}^{n+1}$ 在两个地方相交。这两个交点称为对径点，即它们在 $\mathbb{S}^n$ 中离得尽可能得远。每条线的交点对也是唯一的。如果我们将 $\mathbb{S}^n$ 中所有对径点对等同起来，那么可以在每条通过 $\mathbb{R}^{n+1}$ 的原点的直线与球上的每个对径点对之间定义一个同胚。也就是说所产生的流形 $\mathbb{S}^n/\sim$ 与 $\mathbb{RP}^n$ 同胚。

另外一个阐述等同的方法是：$\mathbb{RP}^n$ 仅仅是 $\mathbb{S}^n$ 的上半部分，其中每个赤道点等同于它的对径点。因此如果你试图走进南半球，你将发现你自己在世界的另一边向北行走。可以将 $\mathbb{RP}^2$ 的特殊情况和 $\mathbb{S}^2$ 的上半部分形象化。假设将图 4.5 $\mathbb{RP}^2$ 中的一个正方形变形成一个圆盘，其中相对的点是等同的，那么结果仍然表示 $\mathbb{RP}^2$。圆盘的中心现在被抬高离开平面形

成了 $\mathbb{S}^2$ 的上半部分。

4.1.3 路径与连通

运动规划的核心是确定是否可以从空间中的某个地方到达空间中的另外一个地方。在第 2 章中，我们采取一系列行动可以使一种状态到达另一种状态。对运动规划，与此相似的是通过连续路径连接位形空间的一个点与另外一个点。在离散规划中，图的连通性非常重要。本节介绍拓扑空间中的连通性。

路径 令 X 是一个拓扑空间，我们的目的仍然是流形。一条路径是一个连续函数 $\tau:[0,1]\to X$。可以选择 $\mathbb{R}$ 作为 τ 的值域。一条路径是一个函数，而不是点的集合。路径上的每个点由 $\tau(s)$ 给出，$s\in[0,1]$。这是对应用第 2 章的规划时所访问的状态序列的推广。可以定义阶段的可数集合，被访问的状态表示为 $x_1, x_2, \cdots$。目前采用 $\tau(s)$，其中 s 代替阶段索引。为清晰起见，我们使用 $x(s)$，而不是 $\tau(s)$，其中 $s\in[0,1]$。

连通与路径连通 一个拓扑空间称为是连通的，如果它不能表示成两个没有交集的非空开集合的并集。该定义非常一般，如果 X 是连通的，这并不意味着在 X 中的任何点对之间都存在路径，例如拓扑学家的正弦曲线：

$$X=\left\{(x,y)\in\mathbb{R}^2 \mid x=0 \text{ or } y=\sin(1/x)\right\} \tag{4.8}$$

考虑绘制 X 的图形。$\sin(1/x)$部分产生 y 轴附近的振荡，频率趋近于无穷。它与 y 轴合并后，这个空间是连通的，但是没有从正弦曲线到达 y 轴的路径。

如何才能避免这种问题呢？基本的方法是使用连通的定义中的路径定义。一个拓扑空间 X 被认为是路径连通，如果对于所有 $x_1, x_2\in X$，存在一条路径 τ 使 $\tau(0)=x_1$，$\tau(1)=x_2$。可以看到如果 X 是路径连通的，那么在以前定义的意义下它也是连通的。

另外一个避免这种问题的方法是限制所考虑的拓扑空间的种类。这里采用这种方法，假设所有拓扑空间是流形。在这种情况下，不会发生类似式（4.8）所描述的奇怪的事情[1]，那么连通与路径连通的定义是一致的[454]。因此我们简称一个空间是连通的。但是有时在连通的定义不满足时，我们应该说 X 是路径连通的。

单连通 既然已经建立了连通的概念，那么下一步是表达不同种类的连通。这可以使用同伦的概念，这个概念被直观地认为是一种将一条路径不断“扭曲”或者“变形”成为另外一条路径的方法，如图 4.6(a)所示。

两条路径 τ_1 和 τ_2 是同伦的（端点固定），如果存在一个连续函数 $h:[0,1]\times[0,1]\to X$ 满足以下 4 个条件：

（1）（从第一条路径开始）对于所有 $s\in[0,1]$，$h(s,0)=\tau_1(s)$。

（2）（到第二条路径结束）对于所有 $s\in[0,1]$，$h(s,1)=\tau_2(s)$。

（3）（保持开始点固定）对于所有 $t\in[0,1]$，$h(0,t)=h(0,0)$。

（4）（保持结束点固定）对于所有 $t\in[0,1]$，$h(1,t)=h(1,0)$。

参数 t 可以认为是一个圆形把手，转动它将路径逐渐从 τ_1 变形到 τ_2。前两个条件分别指示 $t=0$ 产生 τ_1，$t=1$ 产生 τ_2。后两个条件指示路径终点保持固定。

1 拓扑学家的正弦曲线不是一个流形，因为所有包含（0,0）的开集也包含正弦曲线的某些点。这些开集与 $\mathbb{R}$ 不是同胚的。

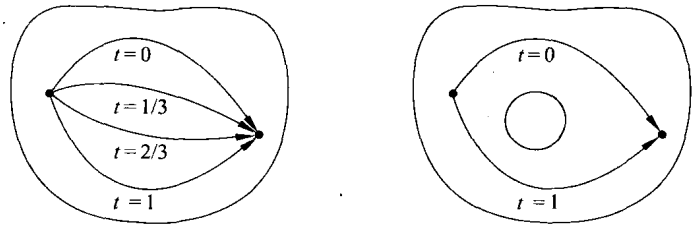


图 4.6 (a) 同伦连续地将一条路径弯曲到另外一条路径中；(b) 在 $\mathbb{R}^2$ 中的一个孔上，路径的像不能连续地被弯曲；因为它引起不连续；在这种情况下，两条路径不是同伦的

在扭曲过程中，路径的像不会产生不连续的跳变。在 $\mathbb{R}^2$ 中，这可以防止同伦从孔中穿过，如图 4.6(b)所示。防止同伦从一些孔中穿过的关键是 h 必须是连续的。在高维情况，存在一些不同种类的孔。例如对于 $\mathbb{R}^3$，假设空间像一块包含气泡的瑞士奶酪。同伦可以在气泡的周围运动，但是不能经过一个贯串整个奶酪的孔。将同伦推广到更高维曲面，那么就可以刻画出现在更高维数情况下的气泡和其他类型的孔，这一点与路径不同[442]。

同伦定义了从某个 $x_1 \in X$ 到某个 $x_2 \in X$ 的所有路径的集合上的等价关系。“等价路径”的概念经常在运动规划、控制论等领域中出现。假设 X 是路径连通的。如果所有路径落入同一个等价类中，那么称 X 是单连通的；否则称 X 是多连通的。

群 由同伦所诱导的等价关系开始进入代数拓扑领域，代数拓扑是用代数方法刻画拓扑空间结构的一个数学分支，例如用群的概念。这些群对运动规划来说有重要的含义。因此这里对它进行简要介绍。首先定义群的概念，一个群是能够进行“$\circ$”二元运算的一个集合 G，它满足如下关于群的公理：

（1）（封闭性）：对于任何 $a, b \in G$，积 $a \circ b \in G$。

（2）（结合性）：对于任何 $a, b, c \in G$，有 $(a \circ b) \circ c = a \circ (b \circ c)$。因此可以不需要圆括号，它们的积可以记为 $a \circ b \circ c$。

（3）（单位元素）：一个元素 $e \in G$ 称为单位元素，如果对于 $e \in G$，它使 $e \circ a = a$，$a \circ e = a$。

（4）（逆元素）：对于每个元素 $e \in G$，存在一个元素 a^{-1}，称为 a 的逆，使 $a \circ a^{-1} = e$ 和 $a^{-1} \circ a = e$。

这里有一些这方面的例子。

例 4.7（群的简单例子） 整数 $\mathbb{Z}$ 的集合是一个关于加法的群，单位元素是 0，每个 i 的逆是 $-i$。去掉 0 的有理数的集合 $\mathbb{Q} \setminus 0$ 是关于乘法的群，单位元素是 1，每个元素的 q 的逆是 $1/q$（0 被去掉，避免被 0 除）。 ■

一些群具有可交换性，即对任意 $a, b \in G$，$a \circ b = b \circ a$。这种情况的群称为可交换的。我们会遇见可交换的和不可交换的群的例子。一个可交换群的例子是 $\mathbb{R}^n$ 上的向量加法。所有三维旋转的集合是一个不可交换群的例子。

基本群 从路径空间以及同伦所产生的等价关系可以构造一个有趣的群。基本群 $\pi_1(X)$（或者第一同伦群）与任何拓扑空间 X 相关联。$f(0) = f(1)$的一条（连续）路径称为一个环。令某 $x_b \in X$ 作为一个基点。对于某任意且固定的基点 x_b，考虑使 $f(0) = f(1) = x_b$ 的所有的环

的集合。通过定义如下二元运算就可以产生群。令 $\tau_1:[0,1]\to X$ 和 $\tau_2:[0,1]\to X$ 是两个有相同基点的环路。将它们的积 $\tau=\tau_1\circ\tau_2$ 定义为：

$$\tau(t)=\begin{cases}\tau_1(2t) & \text{如果 } t\in[0,1/2)\\ \tau_2(2t-1) & \text{如果} t\in[1/2,1]\end{cases} \tag{4.9}$$

这产生了一个连续环路，因为 τ_1 终止于 x_b，而 τ_2 开始于 x_b。从某种意义上讲，这两条路径是首尾相接的。

假设由同伦所诱导的等价关系应用于所有经过固定基点 x_b 的路径的集合。为一个类选择一条特殊的路径这种做法已经不再重要了；可以采用任何表示方法。当环的集合作为一个群来阐述时，也需要用到等价关系。群运算通常是在路径的等价集合上进行的。

当从不同等价类中产生的两条路径用运算。来首尾相接时会发生什么？产生的路径是不是和前两条路径同伦呢？如果原始的两条路径来自于同一个同伦类，那么所产生的路径是同伦的吗？回答通常都是否定的。基本群描述路径的等价类之间的关系，并且刻画 X 的连通性。因为基本群是基于路径的，所以它与运动规划之间有非常紧密的联系。

例 4.8（一个单连通空间） 假设一个拓扑空间 X 是单连通的。在这种情况下，从基点 x_b 开始的所有环路是同伦的，它们构成一个等价类。那么 $\pi_1(X)=1_G$，这是一个仅由单位元素组成的群。■

例 4.9（$\mathbb{S}^1$ 的基本群） 假设 $X=\mathbb{S}^1$。在这种情况下，对于每个 $i\in\mathbb{Z}$，存在一个路径的等价类。如果 $i>0$，那么意味着路径沿逆时针方向环绕 $\mathbb{S}^1$ i 次，然后回到 x_b。如果 $i<0$，那么路径沿顺时针方向环绕 i 次。如果 $i=0$，那么路径等价于保持在 x_b 点上的路径。那么关于加法运算的基本群是 $\mathbb{Z}$。如果 τ_1 沿逆时针环绕 i_1 次，τ_2 沿逆时针环绕 i_2 次，那么 $\tau=\tau_1\circ\tau_2$ 属于一个环类，这个类的环沿逆时针环绕 i_1+i_2 次。考虑加法的逆。如果一条路径绕 $\mathbb{S}^1$ 七次，并且与沿相反方向绕 $\mathbb{S}^1$ 七次的路径结合在一起，那么所得到的结果与保持在 x_b 点上的路径是同伦的。因此 $\pi_1(\mathbb{S}^1)=\mathbb{Z}$。■

例 4.10（$\mathbb{T}^n$ 的基本群） 对于环面 $\pi_1(\mathbb{T}^n)=\mathbb{Z}^n$，其中 $\mathbb{Z}^n$ 的第 i 个分量就是环路环绕 $\mathbb{T}^n$ 的第 i 个分量的次数。这具有直观意义，因为 $\mathbb{T}^n$ 仅仅是 n 个圆的笛卡儿积。从平面的一个单连通的子集开始，钻 n 个相互隔离的圆孔就可以得到基本群 $\mathbb{Z}^n$。这经常出现在移动机器人避免与平面中的 n 个相互隔离的障碍物发生碰撞的情况中。■

到现在为止，基本群似乎只是简单地跟踪路径环绕孔的次数。下一个例子将产生一些非常奇怪的行为，这些行为有助于说明代数拓扑中的一些有趣结构。

例 4.11（$\mathbb{RP}^2$ 的基本群） 假设射影平面 $X=\mathbb{RP}^2$。那么在环路空间上只有两个等价类。绕偶数次的所有路径都是同伦的。同理，缠绕奇数次的所有路径也是同伦的。这种奇怪的行为在图 4.7 中被加以说明。因此产生的基本群只有两个元素：$\pi_1(\mathbb{RP}^2)=\mathbb{Z}_2$，这是一个 2 阶循环群，它对应于模 2 加。这具有直观意义，因为群跟踪的是整数的和是奇数还是偶数，在本例中也就是穿越 $\mathbb{RP}^2$ 正方形的总次数。对于 $\mathbb{RP}^3$ 来说，基本群是相同的，这将出现在第 4.2.2 节，因为它与三维旋转同胚。因此三维旋转的集合也只有两个路径类。■

遗憾的是，两个拓扑空间可能有相同的基本群，即使它们不是同胚的。例如 $\mathbb{Z}$ 是 $\mathbb{S}^1$、圆柱 $\mathbb{R}\times\mathbb{S}^1$ 以及 Mobius 带的基本群。在最后一种情况下，基本群并不表示空间中存在一个“扭转”。另一个问题是具有连通性的空间可能被声明为单连通的。球 $\mathbb{S}^2$ 的基本群仅仅是 1_G，

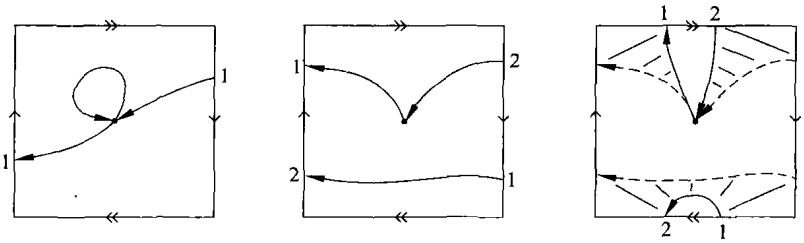


图 4.7 说明为什么 $\pi_1(\mathbb{RP}^n)=\mathbb{Z}_2$；整数 1 和 2 指示当一条路径到达边界时它在什么地方继续；(a) 显示两条不等价的路径；(b) 显示一条绕两次的路径；(c) 这与一条环路同伦，它根本没有缠绕；最后，出现在底部的部分路径被拉向顶部；它最后收缩为一个任意小的环

这与 $\mathbb{R}^2$ 的基本群相同。想象球上的环路；可以看到它们全部落入一个等价类中。因此 $\mathbb{S}^2$ 是单连通的。基本群忽略了 $\mathbb{R}^3$ 中的气泡，因为同伦可以缠绕这些气泡来弯曲路径。通过定义二阶同伦群，一些问题可以得到解决。例如两个变量的连续函数 $[0,1]\times[0,1]\to X$ 可以用来代替一条路径。所产生的同伦产生一类能够弯曲的面或者曲面，这就产生了缠绕 $\mathbb{R}^3$ 中气泡的同伦群 $\pi_2(X)$。这种思路可以扩展到二维以上情况来检测更高维空间中一些不同类型的孔。这就会产生了高阶同伦群的概念。空间上一个比单连通更强的概念是它的所有阶的同伦群等于单位群。这就会防止出现任何类型的孔，并且意味着空间 X 是可缩的，这种可缩性意味着可以构造一种同伦，它可以将 X 缩小到一个点[442]。在平面中，可缩性和单连通是等价的；但是在诸如运动规划中出现的高维空间中，可缩性应该用来指示空间没有内部障碍物（孔）。

另外一种方法是采用同调来产生群，同调基于胞腔复形的结构，而不是同伦映射。这个学科非常复杂，但是在拓扑定理的证明中非常有用。更多关于代数拓扑的阅读材料见本章结尾的文献综述。

4.2 位形空间

本节定义第 3 章中变换所产生的流形。如果机器人有 n 个自由度，变换的集合通常是一个 n 维流形。这种流形称为机器人的位形空间，记为 C 空间。在本书中，C 空间可以认为是一个特殊的状态空间。为了解决运动规划问题，算法必须在 C 空间中对搜索进行引导。C 空间是一个很强的抽象，它将第 3 章中的复杂模型和变换转变为计算一条穿越流形的路径这样的一般问题。通过为此而设计的算法，它们被应用于各种类型机器人和变换中。在第 4.3 节将障碍物引入位形空间，问题会变得复杂，但是在第 4.2 节没有障碍物。

4.2.1 二维刚体：$SE(2)$

第 3.2.2 节的式（3.35）表示如何通过相似变换矩阵 T 对 $\mathbb{R}^2$ 中的刚体进行变换。本章的任务是刻画所有可能的刚体变换的集合的特征。这会是哪种流形呢？这里将给出答案和解释。因为可以选择任何 $x_t, y_t\in\mathbb{R}$ 来进行平移，这独自产生了一个流形 $M_1=\mathbb{R}^2$。任何旋

转 $\theta\in[0,2\pi)$也都可以单独使用。因为 2π 与 0 产生相同旋转，所以可以认为它们是等同的，这就将二维旋转的集合变成流形 $M_2=\mathbb{S}^1$。为了得到对应于所有刚体运动的流形，只需要使 $C=M_1\times M_2=\mathbb{R}^2\times\mathbb{S}^1$。问题的答案是：$C$ 空间是一种圆柱面。

现在给出更详细的说明。主要目的是说明这种简单的、直观的论述将不能应用于三维情况。我们的方法是介绍一些二维情况所使用的技术方法，它们比较容易理解，然后在第 4.2.2 节扩展到三维情况。

矩阵群 第一步首先将变换的集合当作一个群，除可以当作拓扑空间之外[1]。我们从矩阵集合产生几个重要的群，根本上是为了产生对于运动规划来说非常重要的 $n\times n$ 旋转矩阵的群。所有非奇异 $n\times n$ 实数矩阵的集合称为关于矩阵乘法的一般线性群，记为 $GL(n)$。每个矩阵 $A\in GL(n)$都有逆矩阵 $A^{-1}\in GL(n)$，它们相乘得到单位矩阵 $AA^{-1}=I$。矩阵必须非奇异，其原因与从 $\mathbb{Q}$ 中剔除掉 0 是一样的。在矩阵代数中与被 0 除这种情况相似的是不能转换奇异矩阵。

从一个群 G_1 中除去一些元素得到一个子群 G_2，通过这种方式可以构造一些有趣的群。为了得到一个子群，G_2 必须是 G_1 的子集，并且满足关于群的公理。通过构造子群，我们得到旋转矩阵的集合。$GL(n)$的一个重要的子群是正交群 $O(n)$，它是所有满足 $AA^T=I$ 的矩阵 $A\in GL(n)$ 的集合，其中 A^T 表示矩阵 A 的转置。这些矩阵有正交列（内积为零），其行列式总是 1 或者 −1。因此 AA^T 就是每对列向量的内积。如果列不同，结果一定为 0；如果相同，结果为 1，因为 $AA^T=I$。特殊的正交群 $SO(n)$是 $O(n)$的子群，其中每个矩阵的行列式为 1。$SO(n)$的另外一个名字是 n 维旋转矩阵群。

群的链 $SO(n)\leqslant O(n)\leqslant GL(n)$中的“$\leqslant$”表示“子群于”。每个群都可以看作一个拓扑空间。元素为实值的所有 $n\times n$ 矩阵的集合（它不是关于乘法的群）与 $\mathbb{R}^{n^2}$ 是同胚的，因为矩阵中 n^2 个元素可以独立选取。对于 $GL(n)$，奇异矩阵被剔除，但是因此可以得到一个 n^2 维流形。对于 $O(n)$，表达式 $AA^T=I$ 对应于 n^2 个不得不满足的代数方程。这就大大地降低了维数。但是一些方程是多余的。列的内积只有 $\binom{n}{2}$ 个，并且有 n 个方程要求内积为 1。这产生了总共 $n(n+1)/2$ 个独立的方程。每个独立的方程将流形的维数降低一维，并且 $O(n)$的维数是 $n^2-n(n+1)/2=n(n-1)/2$，也就是 $\binom{n}{2}$。为得到 $SO(n)$，需要增加一个约束 $\det A=1$，该约束消除 $O(n)$中一半的元素，但保持维数不变。

例 4.12（矩阵子群） 说明 $n=2$ 的情况。所有 2×2 矩阵的集合为：

$$\left\{\begin{pmatrix}a & b\\ c & d\end{pmatrix}\middle|\, a,b,c,d\in\mathbb{R}\right\} \tag{4.10}$$

它与 $\mathbb{R}^4$ 同胚。所有非奇异 2×2 矩阵的集合可以形成群 $GL(2)$，它引入约束 $ad-bc\neq0$。奇异矩阵的集合形成边界在 $\mathbb{R}^4$ 中的一个三维流形，但是 $\mathbb{R}^4$ 的所有其他元素在 $GL(2)$中；因此 $GL(2)$是一个四维流形。

下面加入约束 $AA^T=I$ 得到 $O(2)$。变成：

$$\begin{pmatrix}a & b\\ c & d\end{pmatrix}\begin{pmatrix}a & c\\ b & d\end{pmatrix}=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \tag{4.11}$$

1 本节考虑的群实际上是李（Lie）群，因为它们是平滑流形[64]。但是我们不用这个名称，因为还没有定义平滑结构的概念。熟悉李群的读者能够认出大多数即将出现的概念。关于李群的更详细的内容将出现在第 15.4.3 节和第 15.5.1 节。

这直接产生4个代数方程：

$$a^2+b^2=1 \tag{4.12}$$

$$ac+bd=0 \tag{4.13}$$

$$ca+db=0 \tag{4.14}$$

$$c^2+d^2=1 \tag{4.15}$$

其中式（4.14）是多余的。有两种方程。式（4.13）给出的方程迫使列的内积为0。这样的方程仅有一个，因为对应 $n=2$ 来说，$\binom{n}{2}=1$。式（4.12）和式（4.15）给出的另外两个约束迫使行(row)成为单位向量。单位向量有两个，因为 $n=2$。所产生的流形的维数为 $\binom{n}{2}=1$，因为我们从 $\mathbb{R}^4$ 开始，且式（4.12）、式（4.13）和式（4.15）使我们失去三个维数。这个流形看起来像什么呢？想象有两个不同的二维单位向量(a, b)和(c, d)，(a, b)可以选择任何值，只要 $a^2+b^2=1$。这看起来像 $\mathbb{S}^1$，但是(a, b)和(c, d)的内积必须为0。因此对于(a, b)的每个值，c 和 d 有两种选择：（1）$c=b$ 和 $d=-a$；（2）$c=-b$ 和 $d=a$，这就是两个圆。流形是 $\mathbb{S}^1 \sqcup \mathbb{S}^1$，其中 $\sqcup$ 表示不相交的集合的并集。当然这种流形是不连通的，因为不存在一条从一个圆到另一个圆的路径。

最后一步是要求 $\det A=ad-bc=1$ 得到所有二维旋转矩阵的集合 $SO(2)$。如果没有这个条件，矩阵产生刚体的旋转后的镜像。约束使 c 和 d 选择 $c=-b$ 和 $d=a$。这就放弃了 $O(2)$ 中的一个圆，得到 $SO(2)$的一个单独的圆。我们已经得到我们所知的：$SO(2)$与 $\mathbb{S}^1$ 同胚。采用极坐标，参数化这个圆可以得到第3.2.2节给出的标准的二维旋转矩阵（3.31）。 ■

特殊欧氏群 既然已经描述了旋转群 $SO(n)$的特征，接下来将平移和旋转都包括进来。这对应于所有如下形式的$(n+1)\times(n+1)$旋转矩阵的集合：

$$\left\{ \begin{pmatrix} R & v \\ 0 & 1 \end{pmatrix} \middle| R \in SO(n),\, v \in \mathbb{R}^n \right\} \tag{4.16}$$

这看起来像式（3.52）和式（3.56）的一般化，其中 n 分别为 $n=2$ 和 $n=3$。矩阵中的 R 完成 $\mathbb{R}^n$ 中 n 维物体的旋转，v 完成物体的平移。其结果是一个群 $SE(n)$，它称为特殊欧氏群。作为一个拓扑空间，$SE(n)$与 $\mathbb{R}^n \times SO(n)$是同胚的，因为旋转矩阵和平移向量可以独立选择。对于 $n=2$ 的情况，这意味着 $SE(2)$与 $\mathbb{R}^2 \times \mathbb{S}^1$ 同胚，这验证了本节开始时的论述。因此能够在平面内平移和旋转的二维刚体的 C 空间为：

$$C=\mathbb{R}^2 \times \mathbb{S}^1 \tag{4.17}$$

更精确地，应该用 $\cong$ 代替 $=$，表示 C 可以是任意与 $\mathbb{R}^2 \times \mathbb{S}^1$ 同胚的空间；但是我们通常避免使用这个符号。

阐述 C 空间 考虑 C 的拓扑含义。因为 $\mathbb{S}^1$ 是多连通的，$\mathbb{R} \times \mathbb{S}^1$ 和 $\mathbb{R}^2 \times \mathbb{S}^1$ 也是多连通的。很难对 C 进行形象化，因为它是三维流形；但是等同可以非常好地阐述它。从开单位立方体$(0,1)^3 \subset \mathbb{R}^3$ 开始。将形式为$(x, y, 0)$和$(x, y, 1)$的边界点包含进来，并对所有 $x, y \in (0,1)$ 进行等同$(x, y, 0) \sim (x, y, 1)$。这意味着当沿 x 和 y 方向运动时，C 空间有一个“边界”；但是沿 z 方向运动会产生环绕。

对于运动规划算法来说理解到这种环绕的存在是非常重要的。例如考虑 $\mathbb{R} \times \mathbb{S}^1$，因为它比较容易形象化。想象 $C=\mathbb{R} \times \mathbb{S}^1$ 的路径规划问题，如图4.8所示。假设将顶和底等同起来产生一个圆柱面，并且中间有一个障碍物。假设任务是发现一条从 q_I 到 q_G 的路径。如果顶和底不等同，那么不可能将 q_I 连接到 q_G；但是如果算法意识到给它的是一个圆柱面，那

么任务就非常简单。一般情况下，理解 C 的拓扑是非常重要的，否则将会丢失潜在的解。

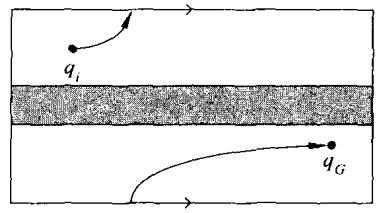


图 4.8 一个规划算法可能不得不穿过等同边界来发现一条路径解

下一节将讨论 n=3 的 $SE(n)$。主要的困难在于确定 $SO(3)$的拓扑。在本书中不考虑 n>3 的情况。

4.2.2 三维刚体：$SE(3)$

读者可能已经料想到为三维刚体所定义的 C 是二维情况的扩展；尽管如此，三维旋转更加复杂。所产生的 C 空间是一个六维流形 $C=\mathbb{R}^3\times\mathbb{RP}^3$。其中三维来自于平移，其他三维来自于旋转。

本节的主要任务是确定 $SO(3)$的拓扑。在第 3.2.3 节中，我们用偏航角、俯仰角、滚动角产生旋转矩阵。这些角度便于形象化，也很容易在软件中实现变换和产生 DH 参数。但是它们只与单个旋转有关，而我们目前的问题是刻画所有旋转的集合的特征。用 α、β 和 γ 来表示旋转的集合是可能的，但是这会引起严重问题。有些情况下，非零角度产生的却是单位旋转矩阵，这等价于 $\alpha=\beta=\gamma=0$。还有一些情况，偏航角、俯仰角、滚动角的值产生相同的旋转矩阵。这些问题破坏了拓扑，在运动规划中会引起理论和实际的困难。

考虑应用第 4.2.1 节的矩阵群的概念。一般线性群 $GL(3)$与 $\mathbb{R}^9$ 同胚。施加约束条件 $AA^T=I$ 就可以确定正交群 $O(3)$。有 $\binom{3}{2}=3$ 个独立方程，它们要求不同的列是正交的，并且三个独立方程迫使每列的内积为 1。这意味着 $O(3)$的维数是 3，这与我们的直觉相吻合，因为第 3.2.3 节有三个旋转参数。为了得到 $SO(3)$，加入最后一个约束 $\det A=1$。回忆例 4.12 中 $SO(2)$包含两个圆，约束 $\det A=1$ 选择其中之一。在 $O(3)$情况下，流形是 $\mathbb{S}^3\sqcup\mathbb{S}^3$，$\det A=1$ 选择其中之一。但是还存在一个问题：这些球上的对径点产生相同的旋转矩阵。当采用四元数来表示 $SO(3)$时会看见这种情况。

使用复数来表示 $SO(2)$ 在引入四元数表示三维旋转之前，先用复数表示二维旋转。令单位复数是指任意复数 $a+bi$，其中 $a^2+b^2=1$。

所有单位复数的集合构成关于乘法的一个群。将会看到它与 $SO(2)$是“相同的”群。需要对这个思路进行更加精确的描述。两个群 G 和 H 被认为是“相同的”，如果它们是同构的，即存在一个双射函数：$f:G\to H$ 使得对于所有 $a,b\in G$，$f(a)\circ f(b)=f(a\circ b)$。这意味着我们可以在 G 中进行某些运算，然后将结果映射到 H，在 H 中进行更多的计算，并且毫无困难地将结果映射回 G。集合 G 和 H 只是表达相同群的两个可选方法。

单位复数与 $SO(2)$同构。为清晰起见，回顾复数可以用极坐标的形式表示为 $re^{i\theta}$，单位复数就是 $e^{i\theta}$。通过用 $e^{i\theta}$ 表示旋转矩阵（3.31），那么可以在二维旋转矩阵和单位复数之间构造一个双射。

如果用复数表示旋转，那么旋转就可以用代数运算来进行。两个旋转的组合就是进行矩阵相乘。等价的运算是复数的相乘。假设二维机器人首先旋转 θ_1，然后旋转 θ_2。在极坐标情况下，复数相乘得到 $e^{i\theta_1}e^{i\theta_2}=e^{i(\theta_1+\theta_2)}$，这清楚地表示了一个 $\theta_1+\theta_2$ 旋转。如果单位复数用直角坐标表示，那么 a_1+b_1i 和 a_2+b_2i 所对应的两个旋转的组合得到 $(a_1a_2-b_1b_2)+(a_1b_2+a_2b_1)i$。虽然常用复数来表达多项式方程的解，但是我们没有这样做，而只是借用它们良好的代数特性。任何时候我们都可以将复数 $a+bi$ 转变为等价的旋转矩阵：

$$R(a,b)=\begin{pmatrix} a & -b \\ b & a \end{pmatrix} \tag{4.18}$$

因为 $a^2+b^2=1$，所以这里仅有一个独立参数需要确定。因此单位复数的集合和 $SO(2)$ 是相同的流形，$SO(2)$是一个圆 $\mathbb{S}^1$（“相同”是在同胚意义下）。

四元数 复数表示二维旋转的方式被改进用四元数来表示三维旋转。令$\mathbb{H}$表示四元数的集合，其中每个四元数 $h\in\mathbb{H}$ 用 $h=a+bi+cj+dk$ 表示，$a,b,c,d\in\mathbb{R}$。一个四元数是一个四维向量。符号 i,j,k 表示四元数的 3 个“虚数”分量。定义如下关系：$i^2=j^2=k^2=ijk=-1$，从它可以得到 $ij=k$、$jk=i$ 和 $ki=j$。那么两个四元数 $h_1=a_1+b_1i+c_1j+d_1k$ 和 $h_2=a_2+b_2i+c_2j+d_2k$ 的乘法可以得到 $h_1\cdot h_2=a_3+b_3i+c_3j+d_3k$，其中：

$$\begin{aligned} a_3&=a_1a_2-b_1b_2-c_1c_2-d_1d_2 \\ b_3&=a_1b_2+a_2b_1+c_1d_2-c_2d_1 \\ c_3&=a_1c_2+a_2c_1+b_2d_1-b_1d_2 \\ d_3&=a_1d_2+a_2d_1+b_1c_2-b_2c_1 \end{aligned} \tag{4.19}$$

使用这种运算，$\mathbb{H}$就是一个关于四元数乘法的群。但是该乘法是不可交换的。三维旋转也是如此。

为方便起见，四元数乘法可以用向量点积和叉积表示。令 $v=[b\quad c\quad d]$是一个三维向量，它表示四元数最后 3 个分量。$h_1\cdot h_2$ 的第一个分量是 $a_1a_2-v_1\cdot v_2$。最后 3 个分量由三维向量 $a_1v_2+a_2v_1-v_1\times v_2$ 给出。

单位复数之于 $SO(2)$的方式与四元数之于 $SO(3)$的方式相同，这意味着$\mathbb{H}$受限于 $a^2+b^2+c^2+d^2=1$。注意到这形成了一个子群，因为单位四元数的乘法产生单位四元数，并且满足其他关于群的公理。

下面描述从单位四元数到 $SO(3)$的映射。将单位四元数 $h=a+bi+cj+dk$ 映射到矩阵：

$$R(h)=\begin{pmatrix} 2(a^2+b^2)-1 & 2(bc-ad) & 2(bd+ac) \\ 2(bc+ad) & 2(a^2+c^2)-1 & 2(cd-ab) \\ 2(bd-ac) & 2(cd+ab) & 2(a^2+d^2)-1 \end{pmatrix} \tag{4.20}$$

已经证明该矩阵是正交的，并且 $\det R(h)=1$。因此它属于 $SO(3)$。如下 h 表示图 4.9 所示的旋转：

$$h=\cos\frac{\theta}{2}+\left(v_1\sin\frac{\theta}{2}\right)i+\left(v_2\sin\frac{\theta}{2}\right)j+\left(v_2\sin\frac{\theta}{2}\right)k \tag{4.21}$$

遗憾的是，这种表示并不是唯一的。从式（4.20）可以证明 $R(h)=R(-h)$。图 4.10 给出了一个几何解释。四元数 h 和 $-h$ 代表相同的旋转，因为关于 v 旋转 θ 等价于关于 $-v$ 旋转 $2\pi-\theta$。考虑关于 v 旋转 $2\pi-\theta$ 的四元数表示形式。实部是：

$$\cos\left(\frac{2\pi-\theta}{2}\right)=\cos\left(\pi-\frac{\theta}{2}\right)=-\cos\frac{\theta}{2}=-a \tag{4.22}$$

i、j、k 分量是：

$$-v\sin\left(\frac{2\pi-\theta}{2}\right)=-v\sin\left(\pi-\frac{\theta}{2}\right)=-v\sin\left(\frac{\theta}{2}\right)=\begin{bmatrix}-b & -c & -d\end{bmatrix} \tag{4.23}$$

这样就构造了四元数$-h$。因此 h 和$-h$ 表示相同的旋转。幸运的是，这是仅有的问题，式(4.20)给出的映射是从单位四元数集合到 $SO(3)$的二对一映射。

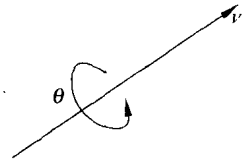


图 4.9 任意三维旋转都可以认为是一个绕由单位方向向量 $v=[v_1 \quad v_2 \quad v_3]$给定的轴转动角度 θ 的旋转

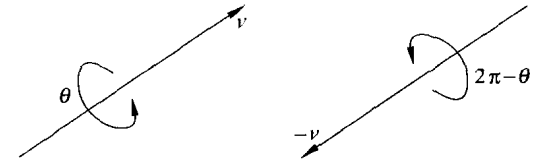


图 4.10 对某些旋转，可以有两种方法进行编码

通过等同可以解决这个问题。因为约束 $a^2+b^2+c^2+d^2=1$，单位四元数的集合与 $\mathbb{S}^3$ 是同胚的。在这一点上，无关于四元数的代数特征。将每个 h 想象为 $\mathbb{R}^4$ 的一个元素，约束 $a^2+b^2+c^2+d^2=1$ 迫使点留在 $\mathbb{S}^3$ 上。使用等同，对于所有单位四元数，声明 $h\sim -h$。这意味着 $\mathbb{S}^3$ 的对径点被等同。由第 4.1.2 节可知：当对径点被等同时，$\mathbb{RP}^n\cong\mathbb{S}^n/\sim$。因此 $SO(3)\cong\mathbb{RP}^3$，即 $\mathbb{R}^4$ 中所有通过原点的直线的集合，但是这很难形象表示。图 4.5 中的 $\mathbb{RP}^2$ 的表示可以扩展到 $\mathbb{RP}^3$。从 $(0,1)^3\subset\mathbb{R}^3$ 开始，构造 3 个不同类型的等同，每个等同对应于立方体的一个相对面，并且将所有的点增加到流形中。每种等同都需要一个扭转（如果没有扭转，那么得到的将会是 $\mathbb{T}^3$）。例如在 z 方向，对所有 $x,y\in[0,1]$，使$(x,y,0)\sim(1-x,1-y,1)$。

得到旋转唯一性的方法是要求处在 $\mathbb{S}^3$ 的“上半部”。例如要求 $a\geqslant 0$，由于 $\mathbb{S}^3$ 的赤道上的对径点产生的缘故，所以只要 $a=0$ 时的边界情况得到正确处理。如果 $a=0$，那么要求 $b\geqslant 0$。但是如果 $a=b=0$，那么要求 $c\geqslant 0$，这是因为诸如 $(0,0,-1,0)$ 和 $(0,0,-1,0)$ 这样的点是相同的旋转。最后如果 $a=b=c=0$，那么仅允许 $d=1$。如果有这些限制，那么很重要的是记住 $\mathbb{RP}^3$ 的连通性。如果一条路径穿越 $\mathbb{S}^3$ 的赤道，那么它必须映射到“北半球”的相应位置。当它到达赤道时，它必须向对径点移动。这些概念容易形象描述，如果你去掉一维，并且想象它们在 $\mathbb{S}^2\subset\mathbb{R}^3$ 中，就像第 4.1.2 节结尾描述的那样。

使用四元数乘法 旋转的表示归结起来就是取 $\mathbb{S}^3$ 上的点，并遵守对径点给出 $SO(3)$的相同元素这样一个事实。从某种意义上，这与四元数的代数特征无关。只是意味着 $SO(3)$ 可以用 $\mathbb{S}^3$ 中的点来表示，就像用 $\mathbb{S}^1$ 的点表示 $SO(2)$一样（在 $SO(3)$上忽略对径点的等同问题）。

尽管如此，引入四元数运算的一个重要的原因是单位四元数群与 $SO(3)$也是同构的。即一系列旋转可以用四元数乘法进行连乘，而不是用矩阵乘法。这是很重要的，因为与矩阵相乘比较而言，四元数乘法的运算量小得多。在任意点上，用式（4.20）可以将结果转换到矩阵形式；当然这是不必要的。世界上的一个点$(x,y,z)\in\mathbb{R}^3$ 可以直接用四元数运算进行变换。当然这需要一个与共轭复数相类似的一种表示。对于任何 $h=a+bi+cj+dk\in\mathbb{H}$，令 $h^*=a-bi-cj-dk$ 表示 h 的共轭。对于任意点$(x,y,z)\in\mathbb{R}^3$，令 $p\in\mathbb{H}$ 表示四元数 $0+xi+$

$yj+zk$。那么旋转后的点(x,y,z)由$h \cdot p \cdot h^*$确定。产生的四元数的i,j,k分量是旋转后的点的新坐标。它等价于用矩阵$R(h)$旋转后的点(x,y,z)。

从旋转矩阵获得四元数参数 从第3.2.3节可知：给定一个旋转矩阵（3.34），用atan2函数可以直接确定偏航角、俯仰角和滚动角。那么四元数参数也可以直接从矩阵产生。这是式（4.20）中函数的逆过程[1]。

给定旋转矩阵（3.43），计算四元数参数[212]。第一个分量是：

$$a=\frac{1}{2}\sqrt{r_{11}+r_{22}+r_{33}+1} \tag{4.24}$$

如果$a \neq 0$，那么：

$$b=\frac{r_{32}-r_{23}}{4a} \tag{4.25}$$

$$c=\frac{r_{13}-r_{31}}{4a} \tag{4.26}$$

$$d=\frac{r_{21}-r_{12}}{4a} \tag{4.27}$$

如果$a=0$，那么会发生以前所提及的过赤道问题。这种情况下：

$$b=\frac{r_{13}r_{12}}{\sqrt{r_{12}^2r_{13}^2+r_{12}^2r_{23}^2+r_{13}^2r_{23}^2}} \tag{4.28}$$

$$c=\frac{r_{12}r_{23}}{\sqrt{r_{12}^2r_{13}^2+r_{12}^2r_{23}^2+r_{13}^2r_{23}^2}} \tag{4.29}$$

$$d=\frac{r_{13}r_{23}}{\sqrt{r_{12}^2r_{13}^2+r_{12}^2r_{23}^2+r_{13}^2r_{23}^2}} \tag{4.30}$$

如果$r_{12}=r_{23}=0$或者$r_{13}=r_{23}=0$或者$r_{12}=r_{13}=0$，那么该方法也会失败。这些方程对应于旋转矩阵（3.39）、（3.40）或者（3.41），它们可以提前确定。

特殊欧氏群 既然$SO(3)$的表示中复杂的部分已经解决，那么$SE(3)$的表达就简单了。$SE(3)$中矩阵的一般形式由式（4.16）给出，其中$R \in SO(3)$且$v \in \mathbb{R}^3$。因为$SO(3) \cong \mathbb{RP}^3$，并且平移可以独立选择，那么在$\mathbb{R}^3$中平移和旋转的刚体的$C$空间是：

$$C=\mathbb{R}^3 \times \mathbb{RP}^3 \tag{4.31}$$

这是一个六维流形。就像预料的那样，C的维数就是空间中自由移动物体的自由度的个数。

4.2.3 物体的链与树

如果有多个物体可以独立运动，那么它们的C空间可以用笛卡儿积来组合。令C_i表示$\mathcal{A}_i$的C空间。如果在$\mathcal{W}=\mathbb{R}^2$或者$\mathcal{W}=\mathbb{R}^3$中有n个自由运动物体，那么：

$$C_i=C_1 \times C_2 \times \cdots \times C_n \tag{4.32}$$

如果物体连接起来形成一个运动链或者运动树，那么每个C空间必须逐一进行考虑，无法简化该过程。一般来说，需要注意的一件事是对于典型关节来说它不可能在整个范围内运动。例如旋转关节不能旋转到任意角度$\theta \in [0,2\pi)$。如果θ不能环绕$\mathbb{S}^1$，那么该关节

1 因为这个函数是二对一的，所以四元数只有被限制在上半球以后才有逆过程。

的 C 空间与 $\mathbb{R}$ 同胚，而不是与 $\mathbb{S}^1$ 同胚。对球形关节也有相似的情形。一个典型球关节由于机械原因而不能完成 $SO(3)$中的任意定向。在这种情况下，C 空间不是 $\mathbb{RP}^3$，因为 $SO(3)$损失了部分空间。

另外一个复杂性是第3.3.2节介绍的DH参数本来是为了有利于坐标系的安排以及变换的计算，但是它没有考虑拓扑。例如表示球形机器人手腕的一个普通方法是构造 3 个零长度连杆，每个连杆都像一个旋转关节。如果运动的范围受限，这可能不会有问题，但是在一般情况下，也会出现类似于用偏航角、俯仰角、滚动角表示 $SO(3)$时所出现的问题。可能有多种方法表达同一手臂位形。

下面给出的几个例子有助于确定物体链或者树的 C 空间。假设 $\mathcal{W}=\mathbb{R}^2$，有一个由 n 个物体通过旋转关节连接而成的链。假设第一个关节能够围绕一个固定点旋转（例如围绕一个钉子进行旋转）。如果每个关节能够进行全范围的运动 $\theta_i \in [0,2\pi)$，那么 C 空间为：

$$C=\mathbb{S}^1\times\mathbb{S}^1\times\cdots\times\mathbb{S}^1=\mathbb{T}^n \tag{4.33}$$

但是如果每个关节运动范围限制到 $\theta_i \in (-\pi/2,\pi/2)$，那么 $C=\mathbb{R}^n$。如果 $SE(2)$中的任意变换都可以应用于 $\mathcal{A}_1$，那么还需要另外一个 $\mathbb{R}^2$。在关节运动受限的情况下，产生 $\mathbb{R}^{n+2}$。如果关节能够任意定向，那么 $C=\mathbb{R}^2\times\mathbb{T}^n$。如果有棱柱形关节，那么每个关节对 C 空间的贡献是 $\mathbb{R}$。

回忆图 3.12，对于 $\mathcal{W}=\mathbb{R}^3$，有 6 个不同类型的关节。旋转关节和棱柱形关节的运动行为和 $\mathcal{W}=\mathbb{R}^2$ 中一样。每个螺旋关节贡献 $\mathbb{R}$。圆柱关节贡献 $\mathbb{R}\times\mathbb{S}^1$，除非旋转运动受限。平面关节贡献 $\mathbb{R}^2\times\mathbb{S}^1$，因为 $SE(2)$中的任何变换都是可能的。如果它的旋转运动受限，那么它贡献 $\mathbb{R}^3$。最后，球形关节理论上可以贡献 $\mathbb{RP}^3$。在实际情况下很少发生。运动受限后它更可能贡献 $\mathbb{R}^2\times\mathbb{S}^1$ 或者 $\mathbb{R}^3$。注意到如果第一个关节是一个自由运动体，那么它贡献 $\mathbb{R}\times\mathbb{RP}^3$。

运动树的处理方法与运动链的方法相同。一个还没有提及的问题是连杆之间可能发生碰撞。到目前为止一直忽略了这个问题，考虑这个问题很明显会增加许多非常复杂的约束条件。第 4.3 节的概念可以用来解决这个问题以及 $\mathcal{W}$ 中的障碍物。考虑这些类型的限制以及所得的空间的路径连通性确实是运动规划的主要内容。

4.3 位形空间障碍物

第 4.2 节定义机器人变换的流形 C 时并没有考虑任何碰撞约束。本节从 C 中删除引起机器人与障碍物发生碰撞，机器人的某些连杆与其他连杆发生碰撞的位形。C 中被删除的部分称为障碍区域。剩余的空间就是路径解必须穿越的空间。一个运动规划算法必须在剩余空间中发现一条从初始位形到达目标位形的路径。最后采用第 3 章的模型以及本章前面部分的内容，可以精确描述运动规划问题。

4.3.1 基本运动规划问题

刚体的障碍区域 假设在 $\mathcal{W}=\mathbb{R}^2$ 或者 $\mathcal{W}=\mathbb{R}^3$ 中包含障碍区域 $\mathcal{O}\subset\mathcal{W}$。假设定义了一

个刚体机器人 $\mathcal{A} \subset \mathcal{W}$；稍后处理多连杆情况。假设$\mathcal{A}$和$\mathcal{O}$表示成第 3.1 节的半代数模型（模型包括多边形和多面体）。令 $q \in C$ 表示$\mathcal{A}$的位形，其中对于 $\mathcal{W}=\mathbb{R}^2$， $q=(x_t, y_t, \theta)$；对于 $\mathcal{W}=\mathbb{R}^3, q=(x_t, y_t, z_t, h)$（$h$ 表示单位四元数）。

障碍区域 $C_{obs} \subseteq C$ 定义为：

$$C_{obs}=\{q \in C \mid \mathcal{A}(q) \cap \mathcal{O} \neq \phi\} \tag{4.34}$$

它是所有位形 q 的集合，在那里机器人 $\mathcal{A}(q)$ 与障碍区域$\mathcal{O}$相交。因为$\mathcal{O}$和$\mathcal{A}(q)$是$\mathcal{W}$中的闭集合，那么该障碍区域是 C 中闭集合。

剩余位形称为自由空间，定义为 $C_{free}=C/C_{obs}$。因为 C 是拓扑空间且 C_{obs} 是闭的，那么 C_{free} 必须是一个开集合。这意味着机器人可以到达距离障碍物任意近的地方。如果$\mathcal{A}$与$\mathcal{O}$发生“接触”：

$$\text{int}(\mathcal{O}) \cap \text{int}(\mathcal{A}(q)) \neq \phi \quad \text{且} \quad \mathcal{O} \cap \mathcal{A}(q) \neq \phi \tag{4.35}$$

那么 $q \in C_{obs}$（int 表示内部）。式（4.35）表示只有它们的边界是相交的。

到达距离任意近的想法在机器人技术中没有实际意义，但是它却可以清楚表述运动规划问题。因为 C_{free} 是开的，表述诸如最短路径等某些最优化问题是不可能的。在这种情况下，应该采用闭包 $cl(C_{free})$，如第 7.7 节所描述。

多物体的障碍区域 如果机器人包括多个物体，情况就更加复杂。式（4.34）的定义仅意味着机器人不与障碍物发生碰撞，但是如果机器人包括多体，那么还需要考虑避免不同连杆之间的碰撞。令机器人建模为 m 个连杆的族$\{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_m\}$，这些连杆可以用关节连接起来，也可以不用关节连接。给整个连杆族指定一个位形向量 q。每个连杆 i 记为$\mathcal{A}_i(q)$，即使 q 的有些参数与连杆$\mathcal{A}_i$的运动无关。例如在一个运动链中，第二连杆的位形不依赖于第九个连杆和第十个连杆之间的角度。

令 P 表示碰撞对的集合，每个碰撞对$(i,j) \in P$ 表示一个连杆索引对 $i, j \in \{1,2,\cdots, m\}$且 $i \neq j$。如果(i,j)在 P 中出现，这意味着在位形向量 q 中不允许出现$\mathcal{A}_i$和$\mathcal{A}_j$，因为对于 q 来说 $\mathcal{A}_i(q) \cap \mathcal{A}_j(q) \neq \phi$。通常 P 并不表示所有碰撞对，因为相邻连杆由于关节的连接而一直保持连接状态。P 的一个通常的定义是每个连杆必须避免与任何不相连接的连杆发生碰撞。对于 m 个物体，P 的大小通常为 $O(m^2)$；但是实际情况下，通过对连杆组进行几何分析，有可能删除一些碰撞对。一些连杆对之间的碰撞在 C 上是不会发生的，它们就不需要出现在 P 中。

使用 P，将机器人的自碰撞问题加入 C_{obs} 的定义中，得到：

$$C_{obs}=\left(\bigcup_{i=1}^{m}\{q \in C \mid \mathcal{A}_i(q) \cap \mathcal{O} \neq \phi\}\right) \cup \left(\bigcup_{[i,j] \in P}\{q \in C \mid \mathcal{A}_i(q) \cap \mathcal{A}_j(q) \neq \phi\}\right) \tag{4.36}$$

因此如果至少有一个连杆与$\mathcal{O}$发生碰撞，或者 P 表示的连杆对中至少有一个对发生碰撞，那么该位形 $q \in C$ 在 C_{obs} 中。

基本运动规划的定义 到现在已经为运动规划问题的定义进行了足够多的准备。图 4.11 对该问题进行了阐述。主要的困难在于不能简单而有效地构造 C_{obs} 或 C_{free} 的边界或者实体表示。

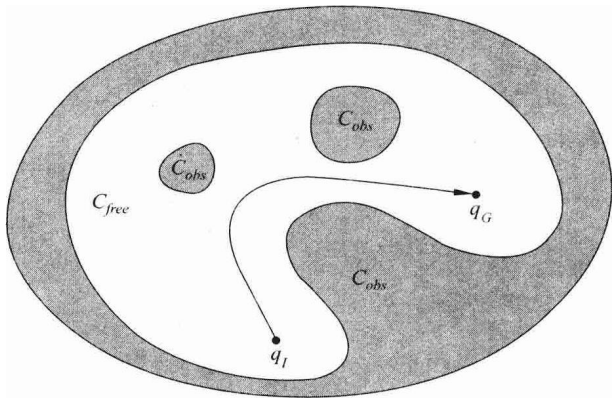


图 4.11 用 C-空间的思路，基本运动规划问题在概念上非常简单：任务是在 C_{free} 中寻找一条从 q_I 到 q_G 的路径：整个水滴表示 $C = C_{obs} \cup C_{free}$

表述 4.1（钢琴搬运工问题）

1. 一个世界 $\mathcal{W}$，其中 $\mathcal{W} = \mathbb{R}^2$ 或者 $\mathcal{W} = \mathbb{R}^3$。

2. 世界中有一个半代数障碍区域 $\mathcal{O} \subset \mathcal{W}$。

3. 在 $\mathcal{W}$ 中定义一个半代数机器人。它可以是刚体机器人或者 m 个连杆 $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_m$ 的族。

4. 位形空间 C 是所有可能应用于机器人的变换的集合。以此为基础可以产生 C_{obs} 和 C_{free}。

5. 位形 $q_I \in C_{free}$ 为初始位形。

6. 位形 $q_I \in C_{free}$ 为目标位形。初始位形和目标位形通常称为疑问对（或者疑问），记为 (q_I, q_G)。

7. 一个完整算法必须计算一条连续路径 $\tau : [0,1] \rightarrow C_{free}$，使 $\tau(0) = q_I, \tau(1) = q_G$，或者正确地报告这样的路径不存在。

从文献[820]可以看到这个问题是 PSPACE 复杂度的，即 NP 复杂度。主要问题在于 C 的维数是无界的。

4.3.2 显式建模 C_{obs}：平移情况

理解如何构造 C_{obs} 是非常重要的。在某些算法中，特别是第 6 章中的组合方法，构造 C_{obs} 是解决问题的第一步。而其他一些算法，特别是第 5 章中的采样规划算法，有助于理解为什么应该避免进行这种构造。

C_{obs} 最简单的情形是 $C = \mathbb{R}^n$，其中 n=1,2,3，并且机器人是仅限于进行平移运动的刚体。在这种情况下，C_{obs} 可以表示为一种卷积。对于任何两个集合 $X, Y \subset \mathbb{R}^n$，它们的 Minkowski 差[1]定义为：

$$X \ominus Y = \{x - y \in \mathbb{R}^n \mid x \in X, y \in Y\} \tag{4.37}$$

其中 x–y 只是 $\mathbb{R}^n$ 上的向量相减。X 和 Y 之间的 Minkowski 差分可以认为是 X 和–Y 之间的

1 在有些书中，如数学和图形处理等，Minkowski 差或者 Minkowski 减的定义会有所不同。所以，有些作者喜欢用 Minkowski 和 $\oplus$ 来定义所有运算，这个和在所有书中的定义是一致的。按照这个传统，我们定义 $X \oplus (-Y)$，这等价于 $X \ominus Y$。

Minkowski 和。只需要在式（4.37）将 X 和 Y 相加就可以得到 Minkowski 和⊕。用 $-y$ 代替每个 $y \in Y$ 可以得到 $-Y$ 集合。

根据 Minkowski 差，$C_{free} = \mathcal{O} \ominus \mathcal{A}(0)$。为了清楚地看到，可以考虑一个一维例子。

例 4.13（一维 C 空间障碍物） 在图 4.12 中，机器人 $\mathcal{A}$=[−1,2]和障碍区域 $\mathcal{O}$ = [0,4]在一维世界 $\mathcal{W} = \mathbb{R}$ 中是一个区间。机器人取反得到的 $-\mathcal{A}$ 是区间[−1,2]。对 $\mathcal{O}$ 和 $-\mathcal{A}$ 求 Minkowski 和，可以得到 C 空间障碍 C_{obs}=[−2,5]。 ■

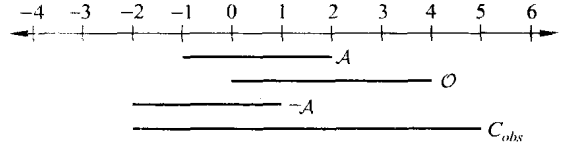


图 4.12 一个一维 C-空间障碍物

Minkowski 差通常被认为是一个卷积。它甚至可以定义成在微分方程和系统论中的形式。作为一个一维例子，令 $f : \mathbb{R} \to \{0,1\}$ 是一个函数：$f(x)$=1 当且仅当 $x \in \mathcal{O}$。相似地，令 $g : \mathbb{R} \to \{0,1\}$ 是一个函数：$g(x)$=1 当且仅当 $x \in \mathcal{A}$。卷积：

$$h(x) = \int_{-\infty}^{\infty} f(\tau) g(x - \tau) \mathrm{d}\tau \tag{4.38}$$

产生一个函数 h，如果 $x \in C_{obs}$，那么 $h(x)$=1；否则，$h(x)$=0。

一个多边形 C 空间障碍 在包含一个凸多边形障碍物 $\mathcal{O}$ 和一个凸多边形机器人 $\mathcal{A}$ 的二维世界中[660]有计算 C_{obs} 的简单算法。这通常称为星算法。对于这个问题，C_{obs} 也是一个凸多边形。回顾到，非凸障碍物和机器人可以建模为凸的组成部分的并集。如果将 C_{obs} 当成凸的组成部分的并集，它的每个凸的组成部分对应于一个与 $\mathcal{O}$ 的凸组成部分发生碰撞的 $\mathcal{A}$ 的凸组成部分，那么下面介绍的概念也会在非凸情况用到。

方法是基于以角度为基础对多边形的边的法线进行排序。关键在于 C_{free} 的每条边是一条从 $\mathcal{A}$ 或者 $\mathcal{O}$ 平移的边。实际上，从 $\mathcal{A}$ 或者 $\mathcal{O}$ 平移的每条边在 C_{obs} 的构造中仅使用一次。仅有的问题是确定 C_{obs} 的这些边的顺序。令 $\alpha_1, \alpha_2, \cdots, \alpha_n$ 表示逆时针顺序围绕 $\mathcal{A}$ 的边的向内法线的角度。令 $\beta_1, \beta_2, \cdots, \beta_n$ 表示 $\mathcal{O}$ 的向外法线的角度。以围绕 $\mathbb{S}^1$ 循环的顺序对这两种角度的集合进行排序后，以遇到它们的先后顺序使用这些与排序的法线相对应的边，这样就可以递增构造 C_{obs}。

例 4.14（三角形机器人和四边形障碍物） 为了加深理解，考虑一个三角形机器人和一个矩形障碍物的情形，如图 4.13 所示。$\mathcal{A}$ 上的黑点表示它的体坐标的原点。考虑机器人以它们经常接触的方式围绕障碍物滑动，如图 4.14(a)所示。这对应于遍历 ∂C_{obs}（C_{obs} 的边界）中的所有位形。$\mathcal{A}$ 的原点描绘出 C_{obs} 的边，如图 4.14(b)所示。有 7 条边，每条边对应于 $\mathcal{A}$ 的一条边或者 $\mathcal{O}$ 的一条边。法线方向的定义如图 4.15(a)所示。当如图 4.15(b)所示排序后，C_{obs} 的边可以递增构造。 ■

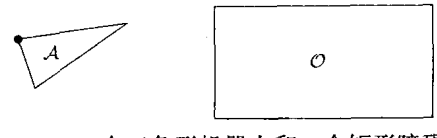


图 4.13 一个三角形机器人和一个矩形障碍物

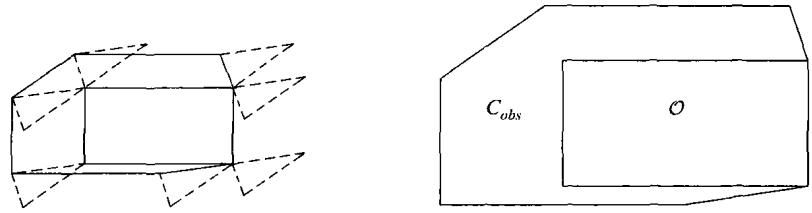


图 4.14 (a) 在障碍物周围滑动机器人，并保持它们之间相互接触；(b) $\mathcal{A}$的原点从 C_{obs} 开始所绘制的边

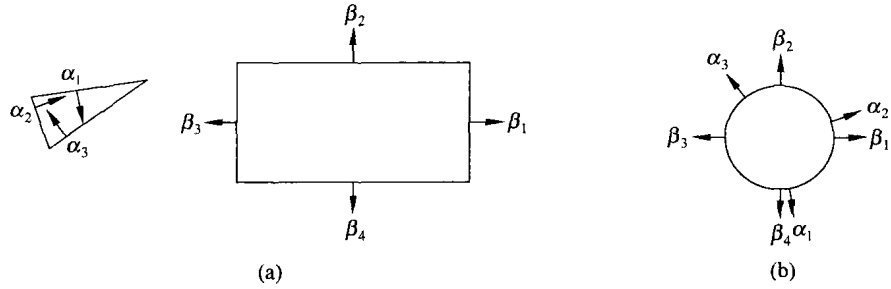


图 4.15 (a) $\mathcal{A}$的边向内的法线和$\mathcal{O}$的边向外的法线；(b) 沿着 $\mathbb{S}^1$ 对边的法线进行排序。这给出 C_{obs} 中的边的顺序

算法的运算时间是 $O(n+m)$，其中 n 是$\mathcal{A}$的边的数量，m 是$\mathcal{O}$的边的数量。在线性时间内就可以对角度进行排序，因为它们已经以逆时针顺序出现在$\mathcal{A}$和$\mathcal{O}$周围；只需要将它们合并。如果两条边共线，那么它们可以首尾相接成 C_{obs} 的一条边。

计算 C_{obs} 的边界 快速地识别 C_{obs} 的每条边。它还根据半平面构造 C_{obs} 的一个实体表示。这要求定义 $n+m$ 个线性方程（假设没有共线的边）。

有两种不同方法生成 C_{obs} 的边，如图 4.16 所示[284,660]。EV 类接触表示$\mathcal{A}$的边与$\mathcal{O}$的顶点相接。它给 C_{obs} 生成 n 条边，分别对应于$\mathcal{A}$的每条边。VE 类接触表示$\mathcal{A}$的顶点与$\mathcal{O}$的边相接触。这给 C_{obs} 生成 m 条边。边的法线之间的关系如图 4.16 所示。对于 EV 类，向内的法线指向障碍物的有共同顶点的边的向外法线之间。相似地，对应 VE 类，$\mathcal{O}$的向外的法线指向$\mathcal{A}$的向内法线之间。

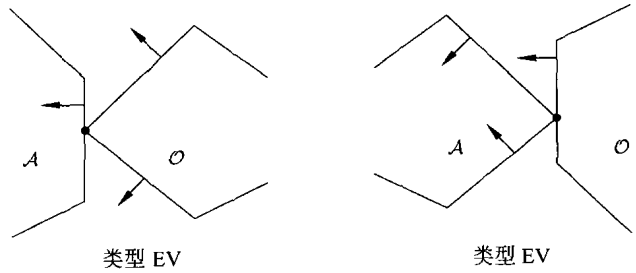


图 4.16 两种不同类型的接触，它们生成 C_{obs} 的不同类型的边[282,660]

使用如图 4.15(b)所示的顺序，当遇到$\mathcal{A}$的法线时，发生 EV 类接触；当遇到$\mathcal{O}$的法线时，发生 VE 类接触。任务是确定每个接触的线性方程。考虑 EV 类接触；VE 类接触可以采用类似的方法处理。除了法线的方向约束外，$\mathcal{O}$的接触顶点还必须落在$\mathcal{A}$的接触边上。回忆到，

凸障碍物是用半平面的交集来构造的。根据半平面就可以定义 C_{obs} 的每条边；因此，仅需要确定$\mathcal{O}$的顶点是否在$\mathcal{A}$的接触边上。当 n 和 v 垂直时，这个条件成立，如图 4.17 所示，产生约束条件 $n \cdot v = 0$。

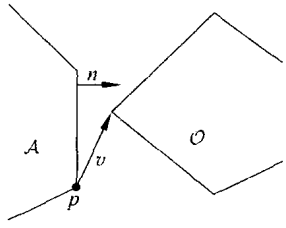


图 4.17 当 n 和 v 垂直时发生接触

注意到法线向量 n 不依赖于$\mathcal{A}$的位形，因为机器人不能旋转。向量 v 依赖于点 p 的平移 $q=(x_t, y_t)$。因此条件记为 $n \cdot v(x_t, y_t)=0$ 更合理。变换方程对平移是线性的；因此 $n \cdot v(x_t, y_t)=0$ 更是 C 中一条直线的方程。例如如果$\mathcal{A}(0,0)$的点 p 的坐标是(1,2)，那么位形 (x_t, y_t)下的 p 的表达式是$(1+x_t, 2+y_t)$。令 $f(x_t, y_t)=n \cdot v(x_t, y_t)$。令 $H=\{(x_t, y_t) \in C \mid f(x_t, y_t) \leqslant 0\}$。注意到不在 H 中的任何位形必须在 C_{free} 中。半平面 H 用来定义 C_{obs} 的边。为每个 EV 类和 VE 类接触所产生的半平面的交集可以完全刻画障碍区域 C_{obs} 的特征。如期所料，这在有 $n+m$ 个边的 C 中产生了一个凸多边形。

例 4.15（C_{obs} 的边界） 考虑为如图 4.18 所示的机器人和障碍物构造一个 C_{obs} 的几何模型。假设$\mathcal{A}$的方向已经固定，并且 $C=\mathbb{R}^2$。在这种情况下，C_{obs} 是一个有七条边的凸多边形。发生接触的条件如图 4.19 所示。顺序按照法线在 $\mathbb{S}^1$ 周围出现的顺序（使用$\mathcal{A}$的向内法线以及$\mathcal{O}$的向外法线）。C_{obs} 的边及其相应接触类型如图 4.19 所示。 ■

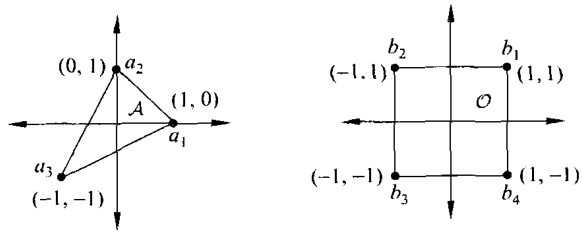


图 4.18 考虑为这个例子构造障碍区域

Type	Vtx.	Edge	n	v	Half-Plane
VE	a_3	b_4-b_1	[1, 0]	$[x_t-2, y_t]$	$\{q \in C \mid x_t-2 \leqslant 0\}$
VE	a_3	b_1-b_2	[0, 1]	$[x_t-2, y_t-2]$	$\{q \in C \mid y_t-2 \leqslant 0\}$
EV	b_2	a_3-a_1	[1, −2]	$[-x_t, 2-y_t]$	$\{q \in C \mid -x_t+2y_t-4 \leqslant 0\}$
VE	a_1	b_2-b_3	[−1, 0]	$[2+x_t, y_t-1]$	$\{q \in C \mid -x_t-2 \leqslant 0\}$
EV	b_3	a_1-a_2	[1, 1]	$[-1-x_t, -y_t]$	$\{q \in C \mid -x_t-y_t-1 \leqslant 0\}$
VE	a_2	b_3-b_4	[0, −1]	$[x_t+1, y_t+2]$	$\{q \in C \mid -y_t-2 \leqslant 0\}$
EV	b_4	a_2-a_3	[−2, 1]	$[2-x_t, -y_t]$	$\{q \in C \mid 2x_t-y_t-4 \leqslant 0\}$

图 4.19 按照边的法线沿着 $\mathbb{S}^1$ 出现的顺序显示的各种接触条件（用$\mathcal{A}$的向内法线和$\mathcal{O}$的向外法线）

多面体 C 空间障碍物 许多以前的思路都可以扩展到仅能在包含多面体障碍物的三维

空间中进行平移的多面体机器人。如果$\mathcal{A}$和$\mathcal{O}$是凸的，那么产生的C_{obs}是凸多面体。

有三种不同类型的接触，每种类型都可以导致C中的半空间：

（1）FV 类：$\mathcal{A}$的一个面与$\mathcal{O}$的一个顶点。

（2）VF 类：$\mathcal{A}$的一个顶点与$\mathcal{O}$的一个面。

（3）EE 类：$\mathcal{A}$的一个边与$\mathcal{O}$的一个边。

如图 4.20 所示，每个半空间定义了多面体C_{obs}的一个面。可以在$O(n+m+k)$时间内构造出C_{obs}，其中n是$\mathcal{A}$的面的数量，m是$\mathcal{O}$的面的数量，k是C_{obs}的面的数量，最大为nm[414]。

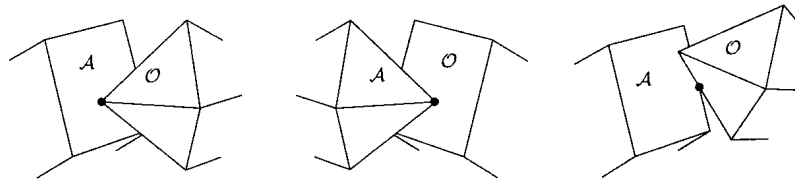


图 4.20 三种不同类型的接触，它们生成C_{obs}的不同类型的面

4.3.3 显式建模 C_{obs}：一般情形

遗憾的是，C_{obs}为多边形或者多面体这种情况是非常有限的。大多数问题产生极度复杂的C空间障碍物。值得庆幸的一点是对于任何用半代数模型定义的机器人和障碍物来说，C_{obs}可以用半代数模型来表示，即使在应用第 3.2 节～第 3.4 节中的任何变换以后。但是对其他类型的变换，如扭曲柔性材料[32,580]，这可能不正确。

考虑二维空间中凸多边形机器人和凸多边形障碍物。假设$SE(2)$中的任何变换都可以应用到$\mathcal{A}$；因此$C=\mathbb{R}^2\times\mathbb{S}^1$和$q=(x_t,y_t,\theta)$。任务是定义一个代数本原的集合，这些本原组合起来是可以定义C_{obs}的。再次强调，区分 EV 和 VE 是非常重要的。考虑如何为 EV 类接触构造代数本原；VE 类接触可以用相似方法进行处理。

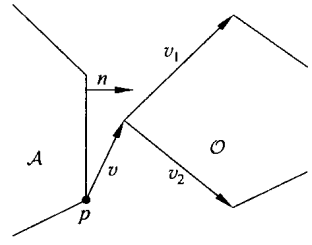


图 4.21 当允许旋转时有助于构造C_{obs}的一个说明

对于仅能够进行平移的情况，通过对边法线进行排序，我们能够确定所有 EV 类型的接触。对有旋转的情况，法线的排序依赖于θ。这意味着 EV 类接触的适用性依赖于机器人的指向θ。回忆到，$\mathcal{A}$的向内法线必须指向包含接触顶点的$\mathcal{O}$的向外法线之间，如图 4.21 所示。这个约束可以用向量v_1和v_1的内积来表达。关于法线方向的论述可以等价地表述为n与v_1之间、n与v_1之间的角度必须小于$\pi/2$。使用内积，这意味着$n\cdot v_1\geqslant 0$和$n\cdot v_2\geqslant 0$。就像平移的情况，接触条件是$n\cdot v=0$。观察到n依赖于θ。对应任何$q\in C$，如果$n(\theta)\cdot v_1\geqslant 0$，$n(\theta)\cdot v_2\geqslant 0$，$n(\theta)\cdot v(q)>0$，那么$q\in C_{free}$。令$H_f$表示满足这些条件的位形的集合。这些条件意味着一个点在C_{free}中，并且任何其他类型 EV 接触和 VE 类型接触也意味着更多的点在C_{free}中。平常地，$H_f\subset C_{free}$，即补集$C\setminus H_f$是C_{obs}的超集合（因此$C_{obs}\subset C\setminus H_f$）。令$H_A=C\setminus H_f$。使用本原：

$$H_1=\{q\in C\,|\,n(\theta)\cdot v_1\leqslant 0\} \tag{4.39}$$

$$H_2 = \{q \in C | n(\theta) \cdot v_2 \leqslant 0\} \tag{4.40}$$

$$H_3 = \{q \in C | n(\theta) \cdot v(q) \leqslant 0\} \tag{4.41}$$

$$H_A = H_1 \cup H_2 \cup H_3$$

我们知道，$C_{obs} \subseteq H_A$，但是 H_A 可能包含 C_{free} 中的点。这种状况类似于第 3.1.1 节中对从半平面构造凸多边形模型所做的论述说明。在目前状况下，只知道 H_A 外面的任何位形一定在 C_{free} 中。对于每个可能的 EV 类和 VE 类接触，如果 H_A 还与所有其他相应集合相交，那么结果就是 C_{obs}。每个接触都有机会删除一部分 C_{free}。最后，足够多的 C_{free} 片被删除，以使剩下的位形一定在 C_{obs} 中。对于任何 EV 类接触，$(H_1 \cup H_2) \setminus H_3 \subseteq C_{free}$。VE 类接触与此相似。可以构造一个与第 3.1.1 节的定义相似的逻辑谓词来确定是否 $q \in C_{obs}$，所用时间与 C_{obs} 本原的数量成线性关系。

还有一个重要的问题。$n(\theta)$的表达式不是多项式，因为在 $SO(2)$的旋转矩阵中有 $\cos\theta$ 和 $\sin\theta$ 项。如果能够用多项式代替这些表达式，那么一切都可以解决，因为法线向量（不是单位法线）的表达式以及内积都是线性函数，因此多项式的变换还是多项式。如此替代可以用球极投影来实现（见文献[591]）；但是一个更简便的方法是使用复数表示旋转。回忆到，当使用 $a+bi$ 表示旋转时，$SO(2)$的每个旋转矩阵由式（4.18）表示，3×3 的相似变换矩阵变为：

$$T(a,b,x_t,y_t) = \begin{pmatrix} a & -b & x_t \\ b & a & y_t \\ 0 & 0 & 1 \end{pmatrix} \tag{4.42}$$

使用这个矩阵对点[x y 1]进行变换得到$(ax-by+x_t, bx+ay+y_t)$。因此$\mathcal{A}$上任何变换后的点是 a,b,x_t,y_t 的线性函数。

这是构造线性函数的一个简单技巧，但是所付出的代价是什么？这依赖于 a 和 b，而不是 θ。这将 C 的维数从 3 增加到 4，并且$C=\mathbb{R}^4$。但是必须增加一个代数本原将 a 和 b 约束在单位圆上。

使用复数，得到每个 EV 类接触和 VE 类接触在 $\mathbb{R}^4$ 中的本原。通过定义 $C=\mathbb{R}^4$，可以得到 EV 类接触的如下代数本原：

$$H_1 = \{(x_t,y_t,a,b) \in C | n(x_t,y_t,a,b) \cdot v_1 \leqslant 0\} \tag{4.43}$$

$$H_2 = \{(x_t,y_t,a,b) \in C | n(x_t,y_t,a,b) \cdot v_2 \leqslant 0\} \tag{4.44}$$

$$H_3 = \{(x_t,y_t,a,b) \in C | n(x_t,y_t,a,b) \cdot v(x_t,y_t,a,b) \leqslant 0\} \tag{4.45}$$

这产生 $H_A = H_1 \cup H_2 \cup H_3$。为了保护 C 的 $\mathbb{R}^2 \times \mathbb{S}^1$ 拓扑，集合：

$$H_s = \{(x_t,y_t,a,b) \in C | a^2+b^2-1=0\} \tag{4.46}$$

与 H_A 相交。集合 H_s 对所有 EV 类接触和 VE 类接触都是不变的，因此它仅需要考虑一次。

例 4.16（C_{obs} 的非线性边界） 考虑给例 4.15 描述的模型增加旋转。在这种情况下，必须考虑所有可能的成对的边的接触。对于这个例子，有 12 个 EV 类接触和 12 个 VE 类接触。每个接触产生 3 个代数本原。包含 H_s 后，这个简单的例子产生了 73 个本原。我们这里仅为一个接触产生本原，而不是产生所有本原。考虑 a_3 和 b_4–b_1 之间的 VE 类接触。向外法线 n 保持固定在 n =[1,0]。向量 v_1 和 v_2 从与 a_3 相邻的边中产生，它们是 a_3–a_2 和 a_3–a_1。

注意到 a_1、a_2 和 a_3 依赖于位形。使用二维相似变换式（3.35），在位形 (x_t, y_t, θ) 下的 a_1 为 $(\cos\theta + x_t, \sin\theta + y_t)$。使用 $a+bi$ 表示旋转，a_1 表达式变为 $(a + x_t, b + y_t)$。a_2 和 a_3 的表达式分别为 $(-b + x_t, a + y_t)$ 和 $(-a + b + x_t, -b - a + y_t)$。那么有 $v_1 = a_2 - a_3 = [a - 2b, 2a + b]$ 和 $v_2 = a_1 - a_3 = [2a - b, a + 2b]$。注意到 v_1 和 v_2 仅依赖于 $\mathcal{A}$ 的指向。假设 v 的指向是从 b_4 到 a_3，那么 $v = a_3 - b_4 = [-a + b + x_t - 1, -a - b + y_t + 1]$。容易计算内积 $v_1 \cdot n$、$v_2 \cdot n$ 和 $v \cdot n$ 形成 H_1、H_2 和 H_3 作为代数本原。

可以产生一个有趣的现象。仅有的非线性本原是 $a^2 + b^2 = 1$。因此 C_{obs} 可以认为是 $\mathbb{R}^4$ 上的一个与圆柱面相交的线性多面体（像一个多面体，但是多一维）。 ■

三维刚体 对于可以应用 $SE(3)$ 中的任何变换的三维刚体，也可以应用相同的一般原则。四元数参数变成非常适合表示 $SO(3)$ 的方法，因为使用式（4.20）可以避免所有三角函数，就像在 $SO(2)$ 中那样。遗憾的是，式（4.20）和式（4.18）对位形变量来说不是线性的，但是它至少是多项式。这就能够形成 C_{obs} 的半代数模型。$SE(3)$ 中产生 FV 类、VF 类和 EE 类接触。从所有接触条件中，可以产生对应于每个 C_{obs} 片的多项式。这些片是有 7 个变量的多项式：$x_t, y_t, z_t, a, b, c, d$。再次，特别的本原必须与所有其他本原相交；这里，增加了使用单位四元数这个约束条件。这将维数从 7 维减少到 6 维。应该加入约束条件来抛弃一半 $\mathbb{S}^3$，这是由于 $\mathbb{S}^3$ 上对径点等同的缘故，所以 $\mathbb{S}^3$ 是冗余的。

物体的链与树 对于物体的链与树，思路从概念上讲也是相同的，但是代数变得更加麻烦。回忆到通过相似变换矩阵的积可以得到每个连杆的变换，就像式（3.53）和式（3.57）中分别给出的二维和三维情况一样。如果 $SO(2)$ 的旋转采用复数表示、$SO(3)$ 采用四元数表示，那么每个矩阵由多项式组成。矩阵乘积以后，可以得到关于位形变量的多项式表达式。因此可以构造一个半代数模型。对于每个连杆，需要考虑所有的接触类型。从例 4.15 和例 4.16 进行外推，可以想象没有人愿意用手工来完成所有这些，因为它完全可以自动进行。对于已有的解决运动规划问题的理论算法来说，自动构造这些表示的能力是非常重要的，可参见第 6.4 节。

如果用 DH 参数来表述 $\mathcal{W} = \mathbb{R}^3$ 的运动链，那么转化到四元数表示可能是不方便的。避免这种情况的一个方法是使用复数方法表示每个以位形变量形式出现的变量 θ_i 和 α_i。这是可以进行的，因为在变换矩阵中仅有 cos 和 sin 出现。它们可以分别用复数的实部和虚部代替。维数将会增加，但是在所有复数必须有单位模的约束条件下，维数又会适当减少。

4.4 闭 运 动 链

本节继续第 3.4 节的讨论。假设一个连杆族排列成环状。在这种情况下，C 空间变得异常复杂，因为关节角度必须保持环的封闭。这导致诸如式（3.80）和图 3.26 所示的约束，其中一些连杆相互之间必须保持在特定相对位置。考虑满足这些约束条件的所有位形的集合。这是一个流形吗？非常遗憾，回答是否定的。但是从被称为簇的代数几何的意义上讲，C 空间属于一个良好空间族。代数几何刻画多项式的解集合的特征。稍后可见，所有运动学都可以表示为多项式。因此所产生的约束条件是一个多项式系统，这并没有什么奇怪的，多项式的解集代表闭运动连杆组的 C 空间。尽管这里考虑的代数簇不一定是流形，但是它

们可以剖分为有限个匹配良好的流形的族[1]。

遗憾的是，闭链中簇的参数表示仅在少数简单情况中是可行的。簇的拓扑极其难以刻画。更糟糕的是，文献[492]已经证明对于每个封闭的、有界的、可以嵌入 $\mathbb{R}^n$ 的实数代数簇来说，存在这样一个连杆组，它的 C 空间与它是同胚的。这些问题意味着在大多数时间里，运动规划算法需要直接用到隐含多项式。对第 6.4.2 节的代数方法，这不会带来任何概念上的困难，因为这些方法直接用到多项式。基于采样的方法通常依赖于对位形进行有效采样的能力，然而在没有参数表示的情况下，采样方法不容易进行适应性改进而应用于簇。第 7.4 节将介绍最新的方法，该方法将基于采样的规划算法扩展应用于封闭链中出现的各种簇。

4.4.1 数学概念

为了理解簇的概念，定义多项式及其解是非常有益的，这些定义比第 3 章中的定义更加正式。

场 多项式通常定义在场上，场是代数的另外一个研究对象。场与群相似，但是它有更多的运算和公理。下面给出场的定义，记住几个熟悉的场的例子：有理数 $\mathbb{Q}$；实数 $\mathbb{R}$；复平面 $\mathbb{C}$。可以验证这些场满足如下六个公理。

场是一个集合 $\mathbb{F}$，它有两个二元运算：$\cdot:\mathbb{F}\times\mathbb{F}\to\mathbb{F}$（称为乘法），$+:\mathbb{F}\times\mathbb{F}\to\mathbb{F}$（称为加法），运算满足如下公理：

（1）（结合律）对于所有 $a,b,c\in\mathbb{F}$，$(a+b)+c=a+(b+c)$和$(a\cdot b)\cdot c=a\cdot(b\cdot c)$。

（2）（交换律）对于所有 $a,b,c\in\mathbb{F}$，$a+b=b+a$ 和 $a\cdot b=b\cdot a$。

（3）（分配律）对于所有 $a,b,c\in\mathbb{F}$，$a\cdot(b+c)=a\cdot b+a\cdot c$。

（4）（恒等式）对于所有 $a\in\mathbb{F}$，存在 $0,1\in\mathbb{F}$，使 $a+0=a\cdot 1=a$。

（5）（加法逆元）对每个 $a\in\mathbb{F}$，存在某 $b\in\mathbb{F}$，使 $a+b=0$。

（6）（乘法逆元）对每个 $a\in\mathbb{F}$，除 $a=0$ 以外，存在某 $c\in\mathbb{F}$，使 $a\cdot c=1$。

将这些公理与第 4.2.1 节的群的定义相比较。可以认为场是两种不同类型的群，其中之一关于乘法，而另外一个关于加法。场通常要求交换律；因此我们不能用四元数建立场。出现分配律是因为两个不同的运算之间存在交集，这对群来说是不可能的。

多项式 假设有 n 个变量 $x_1,x_2,\cdots,x_n$。场 $\mathbb{F}$ 上的一个单项式是一个乘积的形式：

$$x_1^{d_1}\times x_2^{d_2}\times L\times x_n^{d_n} \tag{4.47}$$

其中所有指数 $d_1,d_2,\cdots,d_n$ 是正整数。单项式总的阶数为 $d_1+\cdots+d_n$。

变量为 $x_1,x_2,\cdots,x_n$、系数在 $\mathbb{F}$ 中的一个多项式 f 是有限个单项式的线性组合，单项式的系数也在 $\mathbb{F}$ 中。多项式可以表示为：

$$\sum_{i=1}^{m} c_i m_i \tag{4.48}$$

其中 m_i 是式（4.47）表示的一个单项式，$c_i\in\mathbb{F}$ 是系数。如果 $c_i\neq 0$，那么每个 $c_i m_i$ 称为一项。注意到 f 每一项的指数 d_i 可能是不同的。f 的总阶数是 f 的单项式中的最大总阶数。变

1 这被称为 Whitney 分层。

量为 $x_1,x_2,\cdots,x_n$、系数在 $\mathbb{F}$ 中的所有多项式的集合记为 $\mathbb{F}[x_1,x_2,\cdots,x_n]$。

例 4.17（多项式） 定义对应于我们对多项式的直觉概念。例如假设 $\mathbb{F}=\mathbb{Q}$。$\mathbb{Q}[x_1, x_2, x_3]$中的一个多项式的例子是：

$$x_i^4-\frac{1}{2}x_1x_2x_3^3+x_1^2x_2^2+4 \tag{4.49}$$

其中 1 是一个有效的单项式；因此，$\mathbb{F}$ 的任何元素可以单独作为一项出现，例如式（4.49）中 $4\in\mathbb{Q}$。由于第二项的缘故，式（4.49）的总阶数是 5。一个等价的多项式可以用更合适的变量表示。使用 x, y 和 z 作为变量，那么就产生：

$$x^4-\frac{1}{2}xyz^3+x^2y^2+4 \tag{4.50}$$

它属于 $\mathbb{Q}[x,y,z]$。 ■

多项式的集合 $\mathbb{F}[x_1,\cdots, x_n]$实际上是一个关于加法的群；但它不是一个场。即使多项式可以相乘，但是一些多项式没有乘法逆元。因此集合 $\mathbb{F}[x_1,\cdots, x_n]$经常被称为多项式的可交换环。一个可交换环是一个有二元运算的集合，它满足关于场的每个公理，除乘法逆元的公理以外。

簇 给定一个场 $\mathbb{F}$ 以及整数 n，$\mathbb{F}$ 上的 n 维仿射空间是集合：

$$\mathbb{F}^n=\{(c_1,\cdots,c_n)|c_1,\cdots,c_n\in\mathbb{F}\} \tag{4.51}$$

对本节来说，一个仿射空间可以认为是一个向量空间（精确定义见文献[441]）。因此 $\mathbb{F}^n$ 好像是标量场 $\mathbb{F}$ 的向量版本。我们熟悉的例子有 $\mathbb{Q}^n$、$\mathbb{R}^n$ 和 $\mathbb{C}^n$。

如果用 $\mathbb{F}$ 的元素代替每个变量，并用场的运算来计算表达式，那么 $f\in\mathbb{F}[x_1, x_2,\cdots, x_n]$ 中的一个多项式可以转变成为一个函数：

$$f:\mathbb{F}^n\to\mathbb{F} \tag{4.52}$$

这可以记为 $f(a_1,\cdots,a_n)\in\mathbb{F}$，其中 a_i 表示 $\mathbb{F}$ 中的用来代替变量 x_i 的元素。

这里就产生了一个有趣的问题。对于给定的 f，$\mathbb{F}^n$ 的什么元素使得 $f(a_1,\cdots, a_n)=0$？我们能够问“什么元素使得 f 为某非零值”这个问题，但是这是不必要的，因为可以重新定义一个为零的多项式。例如，$\mathbb{R}^2$ 的什么元素使得 $x^2+y^2=1$？这个我们所熟知的 $\mathbb{S}^1$ 的方程可以重新表述为：$\mathbb{R}^2$ 的什么元素使得 $x^2+y^2-1=0$？

令 $\mathbb{F}$ 是一个场，$\{f_1,\cdots,f_k\}$ 是 $\mathbb{F}[x_1, x_2,\cdots, x_n]$中多项式的集合。该集合：

$$V(f_1,\cdots,f_k)=\{(a_1,\cdots,a_n)\in\mathbb{F}|f_i(a_1,\cdots,a_n)=0,1\leqslant i\leqslant k\} \tag{4.53}$$

称为由 $f_1,\cdots,f_k$ 定义的（仿射）簇。一个有趣的情况是簇的交集和并集都还是簇。因此它们的行为就像第 3.1.2 节中的半代数集合，但是对簇来说，仅允许等式约束。考虑簇 $V(f_1,\cdots,f_k)$ 与 $V(g_1,\cdots, g_l)$。它们的交集是：

$$V(f_1,\cdots,f_k)\cap V(g_1,\cdots,g_l)=V(f_1,\cdots,f_k,g_1,\cdots,g_l) \tag{4.54}$$

因为 $\mathbb{F}^n$ 的每个元素必须为$(f_1,\cdots,f_k, g_1,\cdots, g_l)$中的每个多项式产生一个零值。

为了得到并集，只需要多项式相乘。例如考虑由下式定义的簇 $V_1,V_2\in\mathbb{F}$：

$$V_1=\{(a_1,\cdots,a_n)\in\mathbb{F}|f_1(a_1,\cdots,a_n)=0\} \tag{4.55}$$

$$V_2=\{(a_1,\cdots,a_n)\in\mathbb{F}|f_2(a_1,\cdots,a_n)=0\} \tag{4.56}$$

通过形成多项式 $f=f_1f_2$ 就可以得到集合 $V_1\cup V_2\subset\mathbb{F}$。注意到如果 $f_1(a_1,\cdots,a_n)=0$ 或者

$f_2(a_1,\cdots,a_n)=0$，那么 $f(a_1,\cdots,a_n)=0$。因此 $V_1 \cup V_2$ 是一个簇。簇 V_1 和 V_2 用单个多项式定义，但是相同的思路也可以应用于任何簇。如果有多个多项式，那么 $V(\cdot)$ 的自变量中必须包含所有 $f_i g_i$。

4.4.2 $\mathbb{R}^2$ 上的运动链

为了说明一个概念，首先仔细学习一个简单情况。令 $\mathcal{W}=\mathbb{R}^2$，并假设有一个连杆链 $\mathcal{A}_1,\mathcal{A}_2,\cdots\mathcal{A}_n$，就像例 3.3 中 n=3。假设第一个连杆通过旋转关节连接到原点，每个连杆 $\mathcal{A}_i$ 通过旋转关节连接到 $\mathcal{A}_{i-1}$。这使 C 空间：

$$C=\mathbb{S}^1\times\mathbb{S}^1\times\cdots\times\mathbb{S}^1=\mathbb{T}^n \tag{4.57}$$

是 n 维环面。

二连杆 如果只有两个连杆 $\mathcal{A}_1$ 和 $\mathcal{A}_2$，那么 C 空间可以形象地表示为一个正方形，它的对面是等同的。每个坐标 θ_1 和 θ_2 的范围从 0 到 2π。假设每个连杆的长度为 1。这产生 a_1=1。一个点 $(x,y)\in\mathcal{A}_2$ 被变换为：

$$\begin{pmatrix}\cos\theta_1 & -\sin\theta_1 & 0\\ \sin\theta_1 & \cos\theta_1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\theta_2 & -\sin\theta_2 & 1\\ \sin\theta_2 & \cos\theta_2 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}x\\ y\\ 1\end{pmatrix} \tag{4.58}$$

为了得到多项式，可以应用第 4.2.2 节的技术用 $a_i=\cos\theta_i$ 和 $b_i=\sin\theta_i$ 代替三角函数，其中的约束是 $a_i^2+b_i^2=1$。这产生：

$$\begin{pmatrix}a_1 & -b_1 & 0\\ b_1 & a_1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}a_2 & -b_2 & 1\\ b_2 & a_2 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}x\\ y\\ 1\end{pmatrix} \tag{4.59}$$

其中的约束是对于 i =1,2 必须满足 $a_i^2+b_i^2=1$。这保护了 C 的环面拓扑，但是现在 C 空间嵌入 $\mathbb{R}^4$。每个点的坐标是 $(a_1,b_1,a_2,b_2)\in\mathbb{R}^4$；但是仅有两个自由度，因为每对 a_i,b_i 必须在单位圆上。

将式（4.59）中的矩阵相乘产生多项式 $f_1,f_2\in\mathbb{R}[a_1,b_1,a_2,b_2]$，分别为：

$$f_1=xa_1a_2-ya_1b_2-xb_1b_2+ya_2b_1+a_1 \tag{4.60}$$

$$f_2=-ya_1a_2+xa_1b_2+xa_2b_1-yb_1b_2+b_1 \tag{4.61}$$

对于 x 和 y 坐标。注意到多项式变量是位形参数，而 x 和 y 不是多项式的变量。对于给定的点 $(x,y)\in\mathcal{A}_2$，所有系数都已确定。

零维的簇 现在加入一个运动封闭约束。将处在 $\mathcal{W}$ 的(1,1)点的机器人 $\mathcal{A}_2$ 的体坐标系中的(1,0)点固定。将 x =1 和 y =1 代入式（4.60）和式（4.61）产生约束：

$$f_1=a_1a_2-b_1b_2+a_1=1 \tag{4.62}$$

$$f_2=a_1b_2+a_2b_1+b_1=1 \tag{4.63}$$

这产生簇：

$$V(a_1a_2-b_1b_2+a_1-1,a_1b_2+a_2b_1+b_1-1,a_1^2+b_1^2-1,a_2^2+b_2^2-1) \tag{4.64}$$

它是 $\mathbb{R}^4$ 的子集。稍微修改了多项式，因为每个约束必须写成 f= 0 的形式。

尽管式（4.64）表示二连杆的受约束位形空间，但是它并不十分清晰。没有一个明确的特征（例如参数化），就会使运动规划问题复杂化。从图 4.22 可以看出它仅有两个解。

它们分别发生在 $\theta_1=0$、$\theta_2=\pi/2$ 和 $\theta_1=\pi/2$、$\theta_2=-\pi/2$ 时。根据多项式变量(a_1,b_1,a_2,b_2)，这两个解是(1,0,0,1)和(0,1,0,−1)。可以将这些解代入式（4.64）中的每个多项式来验证结果是 0。因此这个簇表示 $\mathbb{R}^4$ 中的两个点。也可以解释为环面 $\mathbb{S}^1\times\mathbb{S}^1$ 上的两个点。

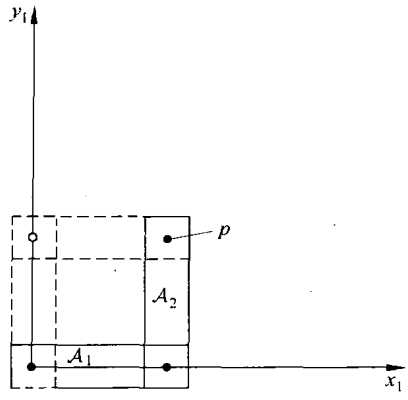


图 4.22 两个位形支撑(1,1)处的点 p

解集的维数为 0 并没有什么奇怪的，如式（4.64）所示，其中有 4 个独立的约束和 4 个变量。根据不同的选择，簇有可能为空。例如将点$(1,0)\in\mathcal{A}_2$ 移到$(1000,0)\in\mathcal{W}$实际上是不可能的。

一维簇 当有解的一个连续统时，会出现更加有趣、复杂的情形。例如，如果去掉一个约束条件，那么可以得到一个一维解集。假设图 4.22 的例子中仅对一个变量有约束。从直觉上看，这将产生一个一维簇。设置 x 的坐标为 0，产生：

$$a_1a_2-b_1b_2+a_1=0 \tag{4.65}$$

并且 y 可以取任何可能的值。如图 4.23(a)所示，点 p 必须沿着 y 轴。这等价于一个三连杆组，其中第三个关节为棱柱形关节并沿着 y 轴。如图 4.23(b)显示所产生的簇 $V(a_1a_2-$

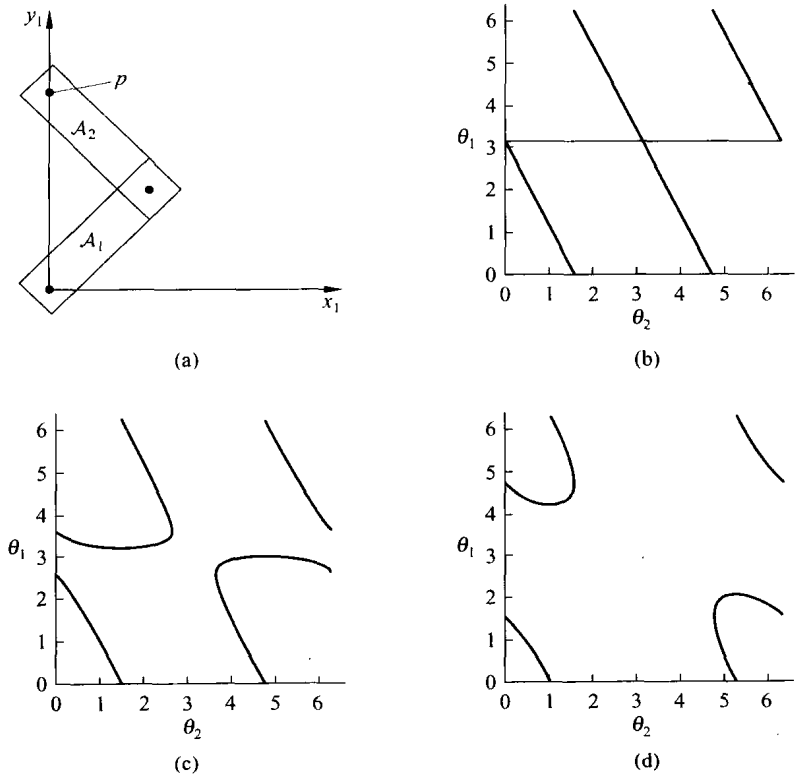


图 4.23 一个约束施加在$\mathcal{A}_2$的点 p 上，如(a)所示；(b)、(c)和(d)中的曲线分别刻画了 $f_1=0$、$f_1=1/8$ 和 $f_1=1$ 情况下的簇

$b_1b_2 + a_1)$，但是为便于形象说明，图形绘制在 $\theta_1 - \theta_2$ 坐标系下将维数从 4 减少到 2。为了正确解释图 4.23，回忆到拓扑是 $\mathbb{S}^1 \times \mathbb{S}^1$，即它的上面和下面是等同的，并且两边也是等同的。图 4.23(b)的中心对应于 $(\theta_1,\theta_1)=(\pi,\pi)$，它防止簇变成流形。因此即使对于这样一个简单的例子，也会失去流形的良好结构。观察到在 (π,π)，连杆完全重叠，$\mathcal{A}_2$ 上的点 p 被置于 $\mathcal{W}$ 中的(0,0)点。图 4.23(b)中的水平线对应于保持两个连杆重叠，改变 θ_1 角可以在它们之间进行摆动。对角线对应于沿如图 4.23(a)所示位形的运动。注意到连杆和 y 轴总是形成一个等边角，那么解集是任何满足 $\theta_2=\pi-\theta_1$ 的 θ_1, θ_2 对。这就是为什么图 4.23(b)所示的对角曲线是线性的原因，图 4.23(c)和图 4.23(d)分别显示有如下约束的簇：

$$a_1a_2 - b_1b_2 + a_1 = \frac{1}{8} \tag{4.66}$$

$$a_1a_2 - b_1b_2 + a_1 = 1 \tag{4.67}$$

在这些情况下，$\mathcal{A}_2$ 上的点(0,1)必须分别服从于 x=1/8 和 x=1。簇是与 $\mathbb{S}^1$ 同胚的流形。从图 4.23(b)到图 4.23(d)的顺序可以想象为的簇收缩为一个小圆的动画的一部分。最后，对于 $a_1a_2 - b_1b_2 + a_1 = 2$，它收缩到一个点，因为当 $\theta_1 = \theta_2 = 0$ 时才有唯一解。除此之外则无解，簇是空集。因此通过允许改变一个约束，可以得到四个不同的拓扑：（1）连接于一点的两个圆；（2）一个圆；（3）一个点；（4）空集。

三连杆 因为对于一维以上的情形，仍然可能进行形象说明，所以假设有 3 个连杆 $\mathcal{A}_1$、$\mathcal{A}_2$ 和 $\mathcal{A}_3$。C 空间可以形象地表示为一个三维立方体，立方体的相对面是等同的。每个坐标 θ_i 的范围为 0～2π。假设每个连杆的长度为 1，即 a_1=a_2=1。点 $(x,y)\in\mathcal{A}_3$ 被变换为：

$$\begin{pmatrix}\cos\theta_1 & -\sin\theta_1 & 0\\ \sin\theta_1 & \cos\theta_1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\theta_2 & -\sin\theta_2 & 10\\ \sin\theta_2 & \cos\theta_2 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\theta_3 & -\sin\theta_3 & 10\\ \sin\theta_3 & \cos\theta_3 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}x\\ y\\ 1\end{pmatrix} \tag{4.68}$$

为了得到多项式，令 $a_i=\cos\theta$，$b_i=\sin\theta$，这样就得到：

$$\begin{pmatrix}a_1 & -b_1 & 0\\ b_1 & a_1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}a_2 & -b_2 & 10\\ b_2 & a_2 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}a_3 & -b_3 & 10\\ b_3 & a_3 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}x\\ y\\ 1\end{pmatrix} \tag{4.69}$$

其中还必须满足约束 $a_i^2 + b_i^2 = 1$，i=1, 2, 3。这保持了 C 的环面拓扑，但它是嵌入在 $\mathbb{R}^6$ 中。矩阵相乘产生多项式 $f_1, f_2 \in \mathbb{R}[a_1,b_1,a_2,b_2,a_3,b_3]$：

$$f_1 = 2a_1a_2a_3 - a_1b_2b_3 + a_1a_2 - 2b_1b_2a_3 - b_1a_2b_3 + a_1 \tag{4.70}$$

$$f_2 = 2b_1a_2a_3 - b_1b_2b_3 + b_1a_2 + 2a_1b_2a_3 + a_1a_2b_3 \tag{4.71}$$

分别对应 x, y 坐标。

考虑增加一个约束：

$$2a_1a_2a_3 - a_1b_2b_3 + a_1a_2 - 2b_1b_2a_3 - b_1a_2b_3 + a_1 = 0 \tag{4.72}$$

它使点 $(1,0)\in\mathcal{A}_3$ 沿着 y 轴。那么所产生的簇是一个有趣的流形，如图 4.24 所示（立方体的侧面是等同的）。

增加 f_1 的值会引起簇的收缩。f_1=7/8 和 f_1=2 的情形，如图 4.25 所示。在 f_1=1 时，簇不是流形，但是它可以变成 $\mathbb{S}^2$。最后这个球缩小到点 f_1=3，对于 f_1>3，簇是空的。

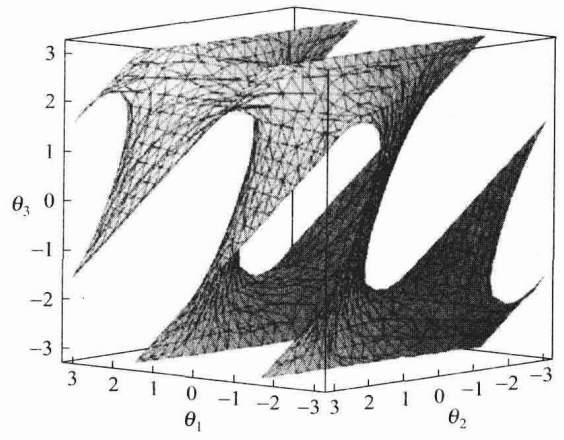


图 4.24 f_1=0 的三连杆链的簇是一个二维流形

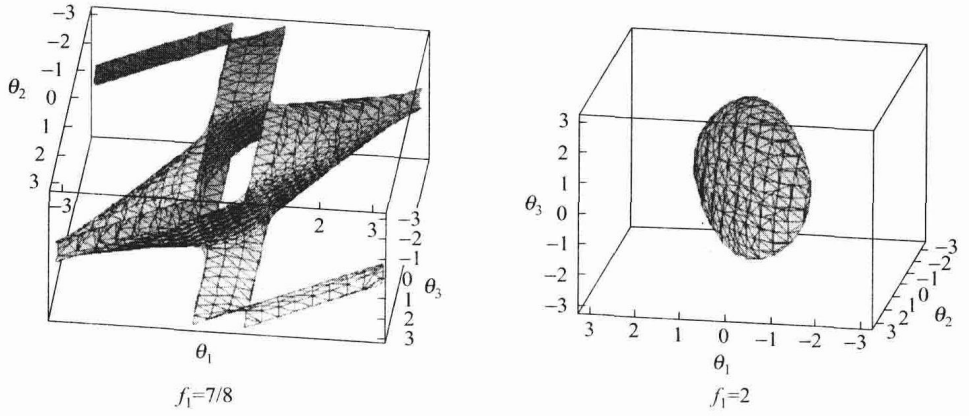


图 4.25 如果 f_1>1，那么簇收缩；如果 1<p<3，那么簇是一个球；在 f_1=0 上它是一个点，并且对于 f_1>3，它将完全消失

我们可以约束 p 点的 y 坐标来得到 f_2=0，而不是通过约束 f_1=0。这产生另外一个二维簇。如果同时施加这两个约束，那么结果是两个原始簇的交集。例如假设 f_1=1 和 f_2=0。这等价于一个四杆机构[312]，其中第四个连杆 A_4 沿着 x 轴从 0 到 1。这个簇：

$$\begin{aligned} V(&2a_1a_2a_3 - a_1b_2b_3 + a_1a_2 - 2b_1b_2a_3 - b_1a_2b_3 + a_1 - 1,\\ &2b_1a_2a_3 - b_1b_2b_3 + b_1a_2 + 2a_1b_2a_3 + a_1a_2b_3) \end{aligned} \tag{4.73}$$

如图 4.26 所示。使用 $\theta_1,\theta_2,\theta_3$ 坐标，解可以表示为线段的一个族。对所有 $t\in[0,\pi]$，在点 $(0,2t,\pi)$、$(t,2\pi-t,\pi+t)$、$(2\pi-t,\ t,\ \pi-t)$、$(2\pi-t,\ \pi,\ \pi+t)$处有解。注意到，这个簇仍然不是流形。选择不同长度的连杆可以生成四杆机构的一族有趣的簇。对于使用球形关节的二维和三维情况，这些机构的拓扑已经确定（见文献[701]）。

4.4.3 定义一般连杆组的簇

我们现在介绍定义簇的一般方法。记住以前的例子将有助于对问题表述的理解。一般情况下，每个约束都可以认为是一个如下形式的论述：

一个点 $p\in\mathcal{A}_j$ 的第 i 个坐标值在 $\mathcal{A}_k$ 的体坐标系中需要固定为 x。

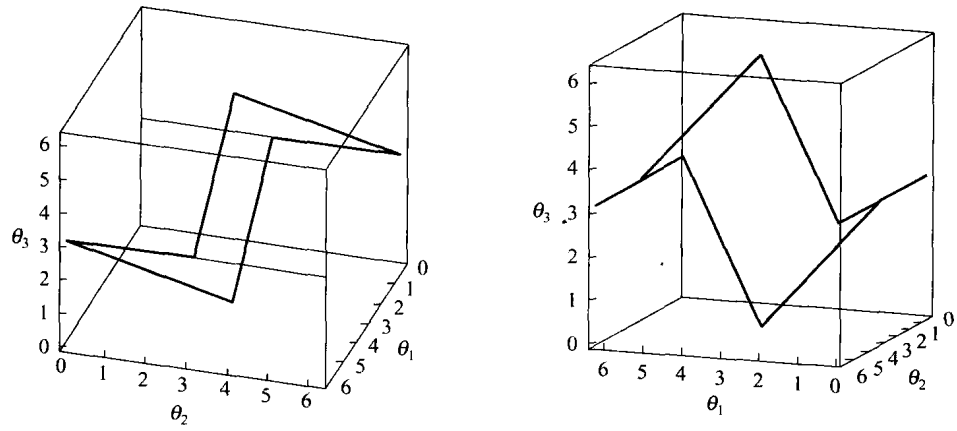


图 4.26 如果施加两个约束 f_1=1 和 f_2=0，那么簇相交得到解的一个一维集合：这个例子等价于一个四杆机构

对于图 4.23(b)中的簇来说，点 $p \in \mathcal{A}_2$ 的第 1 个坐标在 $\mathcal{A}_1$ 的体坐标系中固定为 0。一般情况是必须使一个点关于 $\mathcal{A}_1$ 以外的其他连杆的体坐标系保持固定；第 4.4.2 节的例子不是这种一般情况。

假设 n 连杆 $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n$ 在 $\mathcal{W} = \mathbb{R}^2$ 或者 $\mathcal{W} = \mathbb{R}^3$ 中运动。为方便起见，指定连杆 $\mathcal{A}_1$ 为第 3.4 节中所定义的根。一些连杆成对连接形成关节。从连杆和关节可以构造一个连杆组的图 $\mathcal{G}(V, E)$。$\mathcal{G}$的每个顶点表示 L 中的一个连杆。$\mathcal{G}$中的每条边表示一个关节。这种定义似乎有不足之处，特别是在平面内，因为连杆通常看起来像边，而关节看起来像顶点。所以重新进行安排也是可能的，但是这样不容易扩展到一个连杆有两个以上关节的情况。如果两个以上连杆连接在同一个点上，那么每个连杆产生$\mathcal{G}$的一条边。

确定表达簇的多项式约束的步骤如下：

（1）定义连杆组的图$\mathcal{G}$，图的一个连杆对应一个顶点，一个关节对应一条边。如果一个关节连接多于两个物体，那么必须指定一个物体为交汇点，如图 4.27 和图 4.28(a)所示。在图 4.28 中，连杆 4、13 和 23 被指定为交汇点。

（2）将一个连杆指定为根 $\mathcal{A}_1$。这个连杆在$\mathcal{W}$中可以是固定的，也可以对它进行变换。在后一种情况下，变换的集合根据$\mathcal{W}$的维数可以是 $SE(2)$或者 $SE(3)$。这使得整个连杆组的运动不依赖于它的内部运动。

（3）构造连杆组图$\mathcal{G}$的生成树可以消除环路。这意味着沿着一条路径从根可以到达每个顶点（或者连杆）。可以使用任何生成树。图 4.28(b)显示了删除点画线所表示的边以后所生成的树。

（4）应用第 3.4 节的技术来设置体坐标系以及对连杆树所进行的变换。

（5）对$\mathcal{G}$的每一条没有在 T 中出现的边，写下对应连杆之间的约束的集合。在图 4.28(b)中，可以看出在四个连杆对之间需要约束：14-15、21-22、23-24 和 19-23。

这可能是最需要技巧的部分。如图 3.27 所示，约束可以表述成式（3.81）。这等价于得到图 4.26 中例子的做法，这意味着实际上有两个约束，每个 x 和 y 坐标都有一个约束。如果所有关节是旋转的，图 4.27 中的例子也是如此。假设两个物体 $\mathcal{A}_i$ 和 $\mathcal{A}_k$ 必须刚性连接。这要求增加一个约束来防止相互旋转。这可以通过选择 $\mathcal{A}_j$ 上的另外一个点并保证它的其中

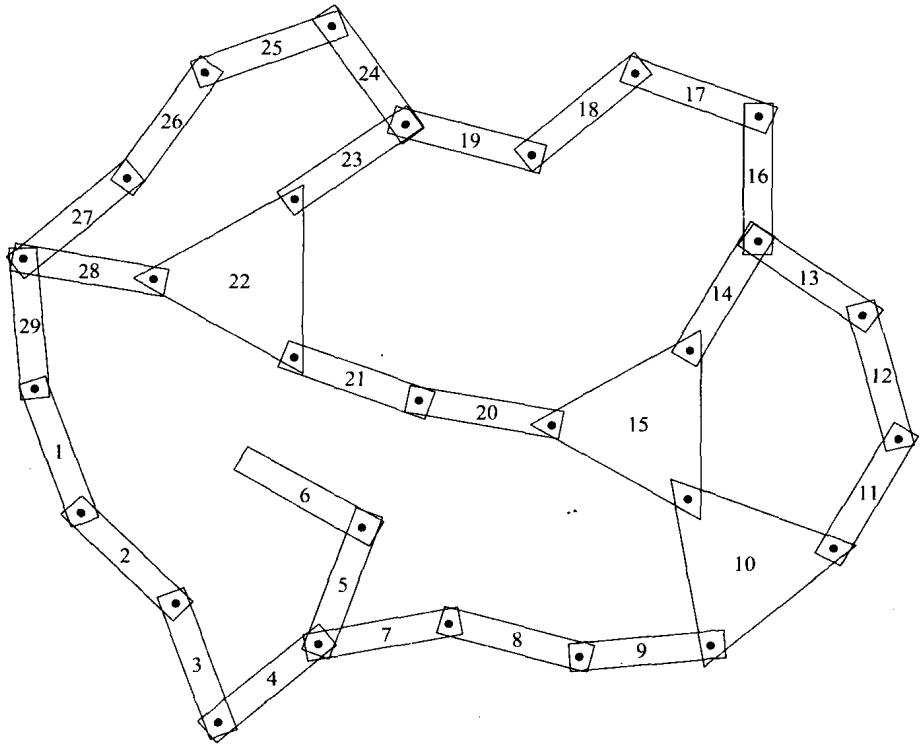


图 4.27 一个有 29 个连杆、几个环的复杂连杆组，其中有些连杆与两个以上的物体相连接，并且有些物体连接两个以上连杆；i 指示连杆 $\mathcal{A}_i$

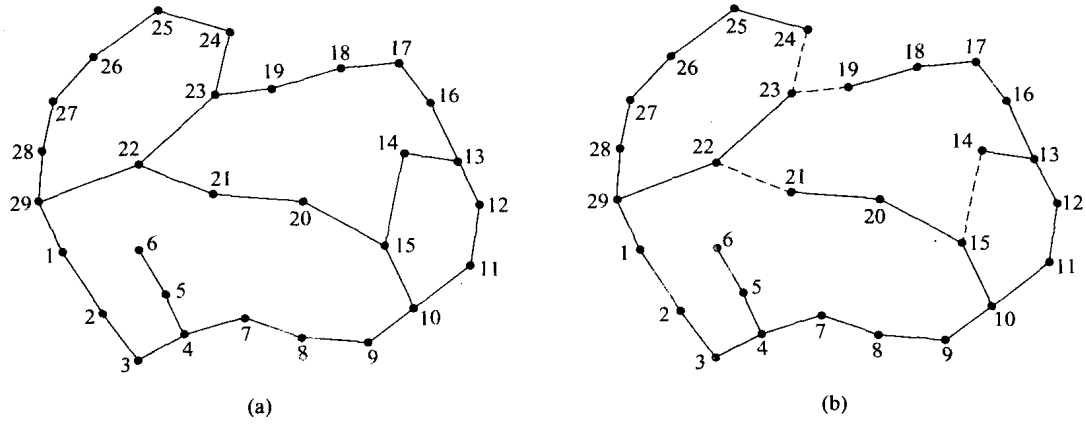


图 4.28 产生连杆组的图的一种方法，本图对应于图 4.27 中的连杆组；(b)用点线表示的被删除的边

一个坐标处在 $\mathcal{A}_k$ 的体坐标系的正确位置。如果增加 4 个方程，每个点对应两个方程，那么它们其中之一是冗余的，因为 $\mathcal{A}_j$ 关于 $\mathcal{A}_k$ 仅可能有三个自由度（来自于 $SE(2)$的维数）。

一个相似但更加复杂的情形发生在 $\mathcal{W}=\mathbb{R}^3$ 情况下。固定一个点产生三个约束。如果一个点被固定，那么 $\mathcal{A}_j$ 关于 $\mathcal{A}_k$ 可以进行 $SO(3)$中的任何旋转。这意味着 $\mathcal{A}_j$ 和 $\mathcal{A}_k$ 是通过球形关节连接的。如果它们用旋转关节连接，那么还需要另外两个约束，这两个约束可以根据第二个点的坐标来进行选择。如果 $\mathcal{A}_j$ 和 $\mathcal{A}_k$ 是刚性连接，那么需要一个来自于第三个点的约束。

独立约束不多于六个，因为这是 $SE(3)$的维数。

（6）将三角函数转换到多项式。对于任何二维变换，都可以采用复数来代替。如果选择 DH 参数来表示三维情况，那么 $\cos\theta_i$ 和 $\sin\theta_i$ 表达式可以用一个复数来表示，$\cos\alpha_i$ 和 $\sin\alpha_i$ 表达式可以用另外一个复数来表示。如果在参数化中直接使用 $SO(3)$的旋转矩阵，那么可以采用四元数表示。在所有这些情况下，都可以得到多项式表达式。

（7）将约束以多项式方程 $f=0$ 的形式列出。为了写出簇的描述，所有多项式必须置为 0，如第 4.4.2 节中介绍的例子。

是否可能从独立约束的个数来确定簇的维数？回答是否定的。这可以从第 4.4.2 节中的连杆链看出；根据特定的方程，它们会产生各种维数的簇。现在能够进行维数的计算，但是要求更多的手段方法(见本章结尾的参考文献)。但是有一种提供自由度数的上界的方法。假设生成树中连杆组总的自由度是 m。每个独立约束至多可以减少一个自由度。因此如果有 l 个独立自由度，那么簇的维数不会超过 $m-l$。一般方法的这种表达式是 Kutzbach 准则；其平面情况是 Grübler 公式[312]。

最后一个需要关注的情况是障碍区域 C_{obs}。一旦簇被等同，障碍区域和运动规划的定义式（4.34）和表述 4.1 就不需要进行改变。但是位形空间 C 必须重新定义为满足封闭约束的位形的集合。

进一步阅读

第 4.1 节介绍了拓扑的基本定义和概念。对它进行进一步学习将有助于加深对位形空间的理解。有许多关于拓扑的书籍，其中许多都会令人生畏，需要读者具备一定层次的数学积累。作为一个深入浅出地介绍拓扑的文献，读者可以参考文献[538]。另外一本循循善诱地介绍拓扑的是文献[499]。一本适合研究生的教科书可以从 Internet 上得到[442]。其他文献包括[38,454]。为了理解拓扑中许多概念定义的目的，文献[912]是很有帮助的。第 4.1.2 节介绍的流形比在多数文献中介绍得要更加简单，因为大多数材料中都介绍了平滑流形，它的可微性要求使它本身更加复杂化（本章并不需要如此）；平滑流形参见第 8.3.2 节。在一个拓扑图上移动的点的位形空间，可参见文献[5]。

第 4.2 节介绍了基本 C-空间定义。更多关于矩阵群及其拓扑特征，参见文献[64]，它提供了向关于李群理论的过渡。更多关于工程中的四元数，参见文献[212,566]。第 4.2 节的其余部分和第 4.3 节的大多数内容来源于文献[591]。文献[222]中也介绍了 C-空间。更多关于计算 C_{obs}，可参考关于位图的文献[516,739]，组合方法见第 6 章和文献[867]。

第 4.4 节中大多数表示是来自于文献[252]中对代数簇的介绍，文献[252]中甚至还包括机器人的例子；它也介绍了确定簇的维数的方法。许多算法出现在文献[707]中。关于用闭运动链设计而成的机器人更详细的内容，参见文献[696]。

习　题

1. 考虑集合 $X=\{1,2,3,4,5\}$。令 X、ϕ、$\{1,3\}$、$\{1,2\}$、$\{2,3\}$、$\{1\}$、$\{2\}$和$\{3\}$为 X 的所有子集的族，指定 X 为一个开集合。

（a）X是一个拓扑空间吗？

（b）如果{1,2,3}加入开集合的族中，那么它是一个拓扑空间吗？试进行说明。

（c）闭集是什么（假设{1,2,3}作为一个开集合而被包括在内）？

（d）X的任意子集既不是开的，也不是闭的吗？

2. 奇异拓扑的连续函数：

（a）对于例 4.4 中的奇异拓扑，给出连续函数 $f:X\to X$ 的一个例子。

（b）刻画所有可能的连续函数集合的特点。

3. 对于俄语字母 А、Б、В、Г、Д、Е、Ё、Ж、З、И、Й、К、Л、М、Н、О、П、Р、С、Т、У、Ф、Х、Ц、Ч、Ш、Щ、Ъ、Ы、Ь、Э、Ю、Я，确定哪一对是同胚的。在解决问题之前，将每个字母想象成 $\mathbb{R}^2$ 的一个一维子集并相应地把它们描绘出来。

4. 证明同胚能够在所有拓扑空间的族上产生一个等价关系。

5. 对于一个在 $\mathbb{R}^3$ 中能够平移和旋转的圆柱形的杆来说，它的 C-空间维数是多少？如果杆绕其中心轴旋转，假设杆的位置和方向在任何可以检测的方式下都是不变的。用更简单空间（诸如 $\mathbb{S}^1$、$\mathbb{S}^2$、$\mathbb{R}^2$、P^2 等）的一个笛卡儿积来表达杆的 C-空间。你的理由是什么？

6. 令 $\tau_1:[0,1]\to\mathbb{R}^2$ 是一条单位圆环形路径，它被定义为 $\tau_1(s)=\big(\cos(2\pi s),\sin(2\pi s)\big)$。令 $\tau_2:[0,1]\to\mathbb{R}^2$ 为另外一个环形路径 $\tau_2(s)=\left(-2+3\cos(2\pi s),\frac{1}{2}\sin(2\pi s)\right)$。这条路径是中心在(–2,0)的椭圆。试说明 τ_1 和 τ_2 是同伦的（是指用一个能够将 τ_1 “形态”到 τ_2 的加性参数来构造一个连续函数）。

7. 证明同伦在从某 $x_1\in X$ 到某 $x_2\in X$ 所有路径的集合上产生一个等价关系，其中 x_1 和 x_2 可以任意选择。

8. 对于一个能够在一个二维视频游戏中平移和旋转的航天器，试确定其 C-空间。屏幕的左右边缘被等同，上下边缘也被等同。在等同中没有任何“扭转”。

9. 对第 4.3.3 节中 H_A 进行推导，但是考虑的是 VE 类型接触。

10. 试确定在一个巨大球体上行驶的汽车的 C-空间（诸如没有高山和海洋的地球）。假设球体足够大使得它的曲率可以被忽略（例如汽车水平地停在没有起伏的地球球面上）。提示：它不是 $\mathbb{S}^2\times\mathbb{S}^1$。

11. 假设$\mathcal{A}$和$\mathcal{O}$都定义为等边三角形，其坐标为(0,0)、(2,0)、$(1,\sqrt{3})$。确定 C-空间障碍物。指定它的所有顶点的坐标，指出每条边相应的接触类型。

12. 试说明对于所有单位四元数来说，矩阵（4.20）都是一个有效的旋转矩阵。

13. 试说明在一个场 $\mathbb{F}$ 上的多项式集合 $\mathbb{F}(x_1,\cdots,x_n)$ 是一个关于加法的群，其中 $x_1,\cdots,x_n$ 是变量。

14. 四元数：

（a）定义一个单位四元数 h_1，它表示围绕由向量 $\left[\frac{1}{\sqrt{3}}\quad\frac{1}{\sqrt{3}}\quad\frac{1}{\sqrt{3}}\right]$ 给定的轴旋转 $-\frac{\pi}{2}$。

（b）定义一个单位四元数 h_2，它表示围绕由向量[0　1　0]给定的轴旋转 π。

（c）假设进行 h_1 所表示的旋转，然后进行 h_2 所表示的旋转。这种旋转组合可以表示为绕一个向量给定的轴的一次旋转。给出这个轴和绕该轴所旋转的角度。

15. 对于一个能够实现除了恒等以外的任意定向的球形关节，它能够给 C-空间产生什么拓扑空间？

16. 假设在一个三维世界上自由地漂浮着 5 个多面体物体。它们都能够旋转和平移。如果把它们当作“一个”组合机器人，那么所产生的 C-空间的拓扑是什么（假设物体互相没有连接）？维数是多少？

17. 通过要求整个机器人包含在 G 中，假设在 C-空间中定义一个目标区域 $G \subseteq \mathcal{W}$。例如一辆汽车可能不得不全部停泊在停车位中的一个空间中。

（a）给出与式（4.34）相似的 C_{goal} 的定义，但是 $\mathcal{A}$ 包含在 G 中。

（b）对于 $\mathcal{A}$ 和 G 都是凸多边形的情况，设计一个有效计算 C_{goal} 的算法。

18. 图 4.29(a)显示了由单位正方形的边的等同所定义的 Möbius 带。想象用剪刀沿着两条点画线剪这个带子。描述所产生的拓扑空间。它是流形吗？试给予解释。

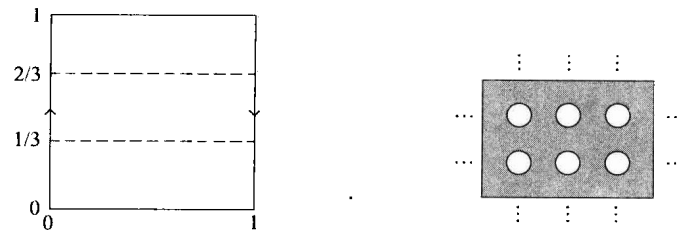


图 4.29 (a) 切开 Möbius 带以后会得到什么拓扑空间；(b) 将孔拉扯到平面外以后会得到一个流形吗

19. 考虑图 4.29(b)，它显示了 $\mathbb{R}^2$ 中这样的点的集合：对于 x 和 y 都是整数的每个 (x, y)，以 (x, y) 为中心、半径为 1/4 的闭圆盘被去掉以后所余下的点的集合。

（a）剩余的点的集合是一个流形吗？试给予解释。

（b）现在去掉半径为 1 的圆盘，是否能够得到一个流形？

（c）最后，去掉半径为 3/2 的圆盘，是否能够得到一个流形？

20. 证明：如图 4.26 所示的解曲线正确地阐述了式（4.73）给出的簇。

21. 对于一个立方体和正四面体，给出 C_{obst} 的面的数量，假设 C 是 $SE(3)$。每个接触类型可以得到多少个面？

22. 沿着第 4.2 节分析矩阵子群的思路，确定 4×4 旋转矩阵的群 $SO(4)$ 的维数。你能够刻画这种拓扑空间的特征吗？

23. 假设给定球形关节的一个运动链。试说明在每个相似旋转矩阵中如何将式（4.20）用作旋转部分，这与用 DH 不同。解释对于运动规划应用来说为什么更愿意采用式（4.20）。

24. 假设对如图 3.29 所示的机械施加一个约束：c 固定在位置(10,10)。如果用复数来表示旋转，请用多项式方程来表达这个约束。

25. 拼图玩具由 18 个通心面条形状的关节组成，它们连接形成一个环。关节之间的每个附件形成一个旋转关节。每个连杆是一个弯管，它们约是一个圆的 1/4。保持这个环所产生的簇的维数是多少？假设玩具可以放置在 $\mathbb{R}^3$ 中任意位置，它的位形空间是什么（包括内部自由度）？

实现

26. 计算 *C*-空间障碍物：

（a）用第 4.3.2 节的算法来构造一个凸多边形 *C*-空间障碍物。

（b）现在允许机器人在平面中旋转。对于任意凸多边形和障碍物，由于不同的类型 EV 和类型 VE 接触变得活跃的缘故，*C*-空间障碍物根本地发生了改变。计算该处的方位角。

（c）将 *C*-空间障碍物的变化做成动画，用机器人方位角作为动画中的时间轴。

27. 考虑从如图 4.5 所示的流形的原点开始，以一个特定的角度离开的“直线”路径（左下角），其中角度是图的输入。直线必须服从等同；因此当线遇到正方形的边时，它可能继续向上。如果需要直线充满整个空间，而不是只形成一个有限图形（例如一个片段、条或者一个格片），研究所需要的条件。

第 5 章　基于采样的运动规划

在第 4.3.1 节的表述 4.1 中讨论运动规划主要有两个基本原理。本章提出其中之一——基于采样的运动规划，简要描述如图 5.1 所示。主要的思路是避免像第 4.3 节那样显式构造 C_{obs}，而是采用采样方式引导搜索探索 C 空间。碰撞检测模块使能这种探索，该模块将运动规划算法当作“黑箱”。这使得与特定几何模型无关的规划算法得以发展。碰撞检测模块处理诸如模型是不是半代数集合、立体三角形、非凸多面体等这一类问题。这种普通的原理近年来非常成功地解决了机器人、制造业、生物应用等涉及成千上万几何本原的问题。采用显式表示 C_{obs} 的方法，这些问题是不可能得到解决的。

完备性　为基于采样的算法定义几个概念是非常有用的。这些算法存在的缺陷是它们并不能保证问题一定有解。一种算法是完备的，如果对于任何输入，该算法都能够在有限的时间内正确报告是否有解。如果有解，那么算法必须在有限的时间内返回一个解。第 6 章中的组合运动规划方法具有这种完备性，但基于采样的规划方法不具有这种完备性。取而代之的是相对较弱的完备性概念。这时会用到稠密性的概念。它意味着当迭代次数趋近于无穷时采样点逼近任何位形。能够进行稠密采样的确定性方法称为分辨率完备的。即如果有解，那么算法将在有限的时间内得到解；但是如果无解，那么算法可能永远进行下去。如果方法是基于随机采样，且以概率 1 稠密，那么这样的算法是以概率完备的，即如果取足够多的点，那么发现解的概率是 1。与此非常有关的信息是收敛的速度，它通常很难确定。

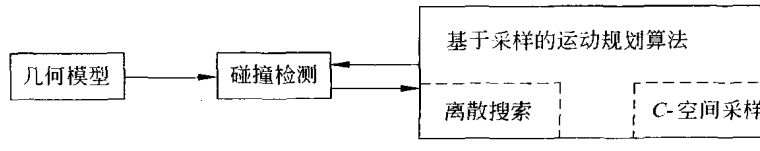


图 5.1　基于采样的规划原理将碰撞检测用作一个“黑箱”，“黑箱”将运动规划和几何模型、运动模型分离开；进行 C-空间采样和离散规划（例如搜索）

第 5.1 节提出度量空间和测度空间的概念，它们是几乎所有采样规划方法的基础。第 5.2 节提出采样的一般概念以及性能准则，它们对于分析采样规划算法的性能是十分有效的。第 5.3 节简要介绍碰撞检测算法，取得对规划算法的可用信息以及得到这些信息所付出的计算代价的理解。第 5.4 节集成采样以及离散规划技术提出定义解决运动规划问题的算法的架构。从给定(q_I, q_G)对这个意义上，这些方法可以被认为是单疑问问题，算法进行搜索直到发现一个解（或者报告搜索失败）。第 5.5 节着重于快速探索随机树（RRT）以及快速探索稠密树（RDT），它们用来发展有效的单疑问规划算法。第 5.6 节介绍多疑问算法，算法通过预处理来构造数据结构，该结构稍后用来得到许多初始-目标对的有效解。在这种情况下，假设障碍区域$\mathcal{O}$对于每个疑问都保持不变。

5.1 C-空间中的距离与体积

实质上，所有基于采样的规划算法都要求能够测量 C 中两个点之间的距离。在多数情况下，这产生了第 5.1.1 节介绍的度量空间。第 5.1.2 节给出了运动规划的例子。在这些算法中定义 C 的子集的体积是非常重要的。这用到第 5.1.3 节介绍的测度空间。第 5.1.4 节介绍不变测度，它应该在任何可能的时候加以应用。

5.1.1 度量空间

在 $\mathbb{R}^n$ 中很容易定义欧氏距离。但是为了在任何 C 上定义距离函数，必须满足相应的公理以使 C 上的距离函数与欧氏距离相一致。

如下定义和公理用来构造将拓扑空间转换到度量空间的函数[1]。度量空间(X, ρ)是一个有函数 $\rho: X \times X \rightarrow \mathbb{R}$ 的拓扑空间 X，使得对于任意 $a, b, c \in X$：

1. 非负性：$\rho(a, b) \geqslant 0$。
2. 自反性：$\rho(a, b)=0$ 当且仅当 $a = b$。
3. 对称性：$\rho(a, b)=\rho(b, a)$。
4. 三角不等式：$\rho(a, b)+\rho(b, c) \geqslant \rho(a, c)$。

函数 ρ 定义了度量空间中点与点之间的距离，关于 ρ 的 4 个条件都与我们直觉的距离相一致。从 a 到 c 的距离总是小于或等于通过 a 和 c 之间的一个中间点 b 后的总距离，从这个意义上，最后一个条件意味着 ρ 是最优的。

L_p 度量 $\mathbb{R}^n$ 上最重要的一族度量由下式给出，对于任意 $p \geqslant 1$：

$$\rho(x, x') = \left(\sum_{i=1}^{n} \left| x_i - x_i' \right|^p \right)^{\frac{1}{p}} \tag{5.1}$$

对每个 p，式（5.1）称为 L_p 度量。3 个最常见的情况是：

（1）L_2：欧氏度量，即 $\mathbb{R}^n$ 上的欧氏距离。

（2）L_1：通常称为 Manhattan 度量，因为在 $\mathbb{R}^2$ 上它对应于沿着轴向栅格运动所得到的路径的长度。例如从(0,0)到(2,5)的距离是 7，即“向东移动 2 个单元”，然后“向北移动 5 个单元”。

（3）L_∞：L_∞ 度量是 p 趋近于无穷时，采用式（5.1）的极限形式来定义的。结果为：

$$L_\infty(x, x') = \max_{1 \leqslant i \leqslant n} \left\{ \left| x_i - x_i' \right| \right\} \tag{5.2}$$

这是正确的，因为 p 值越大，式（5.1）的和式中的最大项也更占优势。

L_p 度量可以从向量空间的范数中产生，$\mathbb{R}^n$ 上的 L_p 范数定义如下：

---

1 有些拓扑空间不是可度量的，这意味着不存在满足公理的函数。许多度量定理给出拓扑空间可度量的充分条件[454]，运动规划中出现的任何空间都是可度量的。

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \tag{5.3}$$

p=2 的情形就是常见的欧氏范数，例如假设向量空间是 $\mathbb{R}^n$，令 $\|\cdot\|$ 表示标准的欧氏范数，L_2 度量就是 $\rho(x, y)=\|x-y\|$。任何 L_p 度量都可以用向量相减的形式表示。

度量子空间　通过验证可以看出，度量空间(X, ρ)的任何子空间 $Y \subset X$ 是一个度量空间，如果 ρ 的定义域限制为 $Y \times Y$。用 $\mathbb{R}^m$ 上的任何 L_p 度量都可以得到第 4 章的流形和簇上的度量。

度量空间的笛卡儿积　将度量很容易扩展到笛卡儿积，因为 C 空间经常由笛卡儿积来构造，特别是在多体情况下。令(X, ρ_x)和(X, ρ_y)表示两个度量空间。为笛卡儿积 $Z= X \times Y$ 构造一个度量空间(X, ρ_z)，通过定义度量 ρ_z：

$$\rho_z(z,z') = \rho_z(x,y,x',y') = c_1\rho_x(x,x') + c_2\rho_y(y,y') \tag{5.4}$$

其中 $c_1>0$ 和 $c_2>0$ 是任意正实值常数，$x, x' \in X$，$y, y' \in Y$。每个 $z \in Z$ 用 $z=(x, y)$表示。

也可以产生 Z 的其他形式的度量，例如：

$$\rho_z(z,z') = \left(c_1 \left[\rho_x(x,x') \right]^p + c_2 \left[\rho_y(y,y') \right]^p \right)^{\frac{1}{p}} \tag{5.5}$$

就是对于任意正整数 p 的一种度量。$c_1>0$ 和 $c_2>0$ 还必须是任意正常数。当然 c_1 和 c_2 也可能有其他选择，理解到这一点是很重要的。

5.1.2　运动规划中重要的度量空间

这里提出一些在运动规划中经常用到的度量空间。

例 5.1（采用复数的 $SO(2)$度量）　如果 $SO(2)$用单位复数表示，那么可以回顾到，由 $\{(a,b) \in \mathbb{R}^2 \mid a^2 + b^2 = 1\}$ 给定的 C 空间是 $\mathbb{R}^2$ 的子集。度量可以用 $\mathbb{R}^2$ 上的任何 L_p 度量。对任何点对(a_1, b_1)和(a_2, b_2)采用欧氏度量：

$$\rho(a_1,b_1,a_2,b_2) = \sqrt{(a_1-a_2)^2 + (b_1-b_2)^2} \tag{5.6}$$

■

例 5.2（进行角度比较的 $SO(2)$度量）　以前的 $SO(2)$度量不给出沿着圆运动的距离。而是计算 $\mathbb{R}^2$ 中连接两个点的线段的长度。有时可能并不想要这种度量。那么通过比较角度 θ_1 和 θ_2 就可以得到另外一种度量。但是在这种情况下需要注意等同问题，因为沿着圆运动有两种方法从 θ_1 到达 θ_2。那么在度量定义中就出现了最小值：

$$\rho(\theta_1,\theta_2) = \min\left\{ |\theta_1 - \theta_2|, 2\pi - |\theta_1 - \theta_2| \right\} \tag{5.7}$$

其中 $\theta_1, \theta_2 \in [0, 2\pi]/\sim$。对于两点$(a_1, b_1)$和$(a_2, b_2)$，这也可以用复数 $a + bi$ 的形式表示成两个向量之间的角度：

$$\rho(a_1,b_1,a_2,b_2) = \cos^{-1}(a_1a_2 + b_1b_2) \tag{5.8}$$

■

例 5.3（$SE(2)$度量）　采用子空间的原理，容易得到 $SE(2)$度量。采用 $SO(2)$的复数形式，$SE(2)$的每个元素是一个点$(x_t, y_t, a, b) \in \mathbb{R}^4$。欧氏度量或者 $\mathbb{R}^4$ 上的任何其他 L_p 度量都可以用来产生度量。■

例 5.4（采用四元数的 *SO*(3)度量） 对于 *SO*(3)，情况会变得非常复杂。单位四元数形成 $\mathbb{R}^4$ 的一个子集 $\mathbb{S}^3$。因此可以采用任何 L_p 度量来定义 $\mathbb{S}^3$ 上的度量，但是这不是 *SO*(3)上的度量，因为需要等同对径点。令 $h_1, h_2 \in \mathbb{R}^4$ 表示两个单位四元数（这里将其阐述为 $\mathbb{R}^4$ 的元素，忽略四元数代数）。考虑等同，度量为：

$$\rho(h_1, h_2) = \min\{\|h_1 - h_2\|, \|h_1 + h_2\|\} \tag{5.9}$$

其中 min 的两个变量分别对应于 h_1 到 h_2、$-h_2$ 的距离。出现 h_1+h_2 是因为 h_2 被取反产生它的对径点$-h_2$。

在 *SO*(2)情况下，式（5.9）的度量似乎会被扭曲，因为它测量的是截取 $\mathbb{S}^3$ 内部的线段的长度，这不同于沿曲面运动。解决这个问题就给出 *SO*(3)的一个非常自然的度量，该度量基于“球形线性插值”。这是将连接点的线段向外推至 $\mathbb{S}^3$ 上。通过降低维数，很容易将它形象化。假设要计算$\mathbb{S}^3$ 上两个点的距离。如果这些点在赤道上，那么球形线性插值方法产生的距离与沿着赤道移动的路径成正比，这与截取 $\mathbb{S}^2$ 内部的线段不同（对于不在赤道上的点，采用通过该点的大圆）。

用两个四元数间的内积很容易定义这种度量。回顾到，对于 $\mathbb{R}^n$ 中的单位向量 v_1 和 v_2，有 $v_1 \cdot v_2 = \cos\theta$，其中 θ 是两个向量间的角度。需要用这个角度来给出沿着 $\mathbb{S}^3$ 的正确距离。所产生的度量是式（5.8）的简单扩展。两个四元数间沿 $\mathbb{S}^3$ 的距离是：

$$\rho_s(h_1, h_2) = \cos^{-1}(a_1a_2 + b_1b_2 + c_1c_2 + d_1d_2) \tag{5.10}$$

其中 $h_i = (a_i, b_i, c_i, d_i)$。考虑等同，那么产生度量：

$$\rho(h_1, h_2) = \min\{\rho_s(h_1, h_2), \rho_s(h_1, -h_2)\} \tag{5.11}$$

■

例 5.5（另外一种 *SE*(2)度量） 对于一些 C 空间，会出现联系不同种类的量的问题。例如定义在 *SE*(2)上的任何度量都必须比较平面距离与三角关系。例如，即使 c_1=c_2=1，用弧度表示的 $\mathbb{S}^1$ 的范围是[0,2π]，用角度表示的范围为[0,360]。如果这两种情况采用相同的常数 c_2，可以得到两种完全不同的度量。应用于 $\mathbb{R}^2$ 和 $\mathbb{S}^1$ 的单位也完全不兼容。 ■

例 5.6（机器人位移度量） 考虑机器人的位移，有时这种不兼容问题可以得到解决。对于任意两种位形 $q_1, q_2 \in C$，机器人位移度量定义为：

$$\rho(q_1, q_2) = \max_{a \in A}\{\|a(q_1) - a(q_2)\|\} \tag{5.12}$$

其中 $a(q_i)$是机器人 $\mathcal{A}$ 处在位形 q_i 时点 a 的位置。直觉地，当机器人从位形 q_1 运动到 q_2 时，机器人位移度量所产生的是机器人的任何部分的最大位移量。计算这种度量的复杂性和有效性取决于特定机器人的几何模型和运动。对于能够平移和旋转的凸多面体机器人来说，仅检查顶点就足够了。该度量看起来非常理想，但是对于大多数情况，尚未得到有效的算法。 ■

例 5.7（$\mathbb{T}^n$ 度量） 在环面 $\mathbb{T}^n$ 上构造度量。诸如式（5.4）和式（5.5）这样的笛卡儿积可以扩展到 $\mathbb{S}^1$ 的每个副本上（一个副本对应一个参数 θ_i）。这产生 n 个系数 $c_1, c_2, \cdots, c_n$。用机器人位移可以确定这些系数。例如，如果机器人是一个连杆链，给第一连杆的变化赋予更大的权重是有意义的，因为第一连杆的运动使能整个链运动。当最后一个参数发生变化，仅仅最后一个连杆运动，在这种情况下，应该给予较小的权重。 ■

例 5.8（*SE*(3)度量） 将笛卡儿积应用于 $\mathbb{R}^3$ 的度量以及 *SO*(3)的度量，就可以形成 *SE*(3)

的度量，如式（5.11）。还剩下系数需要确定。这些问题在第 5.3.4 节还会出现，在那里将会介绍更多关于机器人位移的内容。■

伪度量　许多运动规划算法所使用的函数有点像距离函数，但是不满足所有度量公理。如果使用这些距离函数，那么称它们为“伪度量”。产生伪度量的一般原则是将距离定义为某准则的最优尚需代价（回顾第 2.3 节中的离散尚需代价函数）。当在第 14 章考虑微分约束时，这将变得更加重要。

在连续情况下，代价对应于机器人运动的距离，或者所消耗的能量。有时得到的伪度量是不对称的。例如汽车下坡比上坡需要更少的能量。假设汽车只能向前驾驶。那么它可能运动一个短的距离就可以从 q_1 到 q_2，但是它可能不得不运动一个较长的距离才能从 q_2 到达 q_1，因为它不能向反方向运动。这些问题将出现在第 13.1.2 节和第 15.3.1 节描述的 Dubins 车中。

机器人伪度量的一个重要的例子是势函数，它是第 5.4.3 节介绍的随机势场方法的重要组成部分。其思路是构造标量函数来估计与目标之间的距离；但是可能还有试图将机器人排斥到远离障碍物的附加项。在一般情况下这将产生距离函数的局部最小值，这可能会导致势函数违反三角不等式。

5.1.3　基本测度理论的定义

本节简要介绍如何定义度量空间中的体积。提出定义诸如积分或者概率密度等概念的基础。测度理论是一个先进的数学课题，它超出了本书的研究范围；但是简要介绍一些基本定义也是非常有价值的，因为它们有时会出现在基于采样的规划中。

测度可以认为是一个函数，它产生度量空间(X, ρ)的子集的实数值。理想地，对任何子集 $A \subseteq X$，我们希望产生一个非负值 $\mu(A) \in [0, \infty]$。遗憾的是由于 Banach-Tarski 悖论的缘故，如果 $X = \mathbb{R}^n$，那么当我们试图给一些子集定义体积时就会发生矛盾。如果 X 是有限的，那么这种情况就不会发生。因此很难使这个问题形象化；关于构造奇异不可测集合，参见文献[839]。由于存在这个问题，通过定义避免产生自相矛盾的集合的子集族，我们就得到了一个工作区。X 的一个子集族$\mathcal{B}$称为 sigma 代数，如果它满足如下公理：

1. 空集属于$\mathcal{B}$。
2. 如果 $B \in \mathcal{B}$，那么 $X \setminus B \in \mathcal{B}$。
3. 对于$\mathcal{B}$中可数个集合的族，它们的并集属于$\mathcal{B}$。

最后两个条件意味着$\mathcal{B}$中可数个集合交集也属于$\mathcal{B}$。$\mathcal{B}$中这样的集合称为可测集。

一个称为 Borel 集的 sigma 代数可以由任何度量空间(X, ρ)产生。从 X 中的所有开球的集合开始。对于任何 $x \in X$ 以及 $r \in (0, \infty)$，有如下形式的集合：

$$B(x, r) = \{x' \in X \mid \rho(x, x') < r\} \tag{5.13}$$

Borel 集$\mathcal{B}$可以通过 sigma 代数从这些开球进行构造。例如 $\mathbb{R}^2$ 上的开正方形属于$\mathcal{B}$，因为它构造为可数个球的并集（需要无穷多，因为弯曲的球必须收敛到覆盖正方形的直的边）。通过使用 Borel 集，可以避免不可测集带来的麻烦。

例 5.9（Borel 集）　$\mathcal{B}$的一个简单的例子用 $\mathbb{R}$ 来构造。对任意 $x_1, x_2 \in \mathbb{R}$，$x_1 < x_2$，开球仅仅是所有开区间$(x_1, x_2) \subset \mathbb{R}$ 的集合。■

使用$\mathcal{B}$，将一个测度μ定义为一个函数$\mu:\mathcal{B}\rightarrow[0,\infty]$，使满足测度公理：

1. 对于空集ϕ，有$\mu(\phi)=0$。

2. 对于可数个两两不相交的、可测集合的任何族E_1、E_2、E_3、…，令E表示它们的并集，测度μ必须满足：

$$\mu(E)=\sum_i \mu(E_i) \tag{5.14}$$

其中i是整个族的数量。

例 5.10（Lebesgue 测度） 最常见的测度是 Lebesgue 测度，它已经成为$\mathbb{R}$中长度、$\mathbb{R}^2$中面积、$\mathbb{R}^n$中体积（$n\geqslant 3$）的标准概念。Lebesgue 测度的一个重要概念就是零测度（measure zero）集合的存在。对于任意可数集合A，Lebesgue 测度产生$\mu(A)=0$。例如点$\{1\}\subset\mathbb{R}$的总长度是多少？任何单点的长度必须为零。为了满足测度公理，诸如{1,3,4,5}的集合也必须有零测度。甚至$\mathbb{R}$中的无穷子集$\mathbb{Z}$和$\mathbb{Q}$也有零测度。如果集合$A\subseteq\mathbb{R}^n$的维数为m，其中某整数$m<n$，那么按照$\mathbb{R}^n$上 Lebesgue 测度，$\mu(A)=0$。例如集合$\mathbb{S}^2\subset\mathbb{R}^2$有零测度，因为球面没有体积。但是如果测度空间被限制在$\mathbb{S}^2$，且定义了表面积，那么就得到非零测度。 ■

例 5.11（计数测度） 如果(X,ρ)是有限的，那么可以定义计数测度。在这种情况下，测度可以定义在X的整个幂集合上。对于$A\subset X$，计数测度集合产生$\mu(A)=|A|$，即A中元素的数量。可以验证它满足测度公理。 ■

例 5.12（概率测度） 测度论将离散和连续概率理论统一为一个整体。可以定义测度μ来产生概率质量。概率的性质与测度公理一致，所以它也产生一个测度空间。分别用来定义连续和离散情况下随机变量的期望的积分和求和这两种运算被统一到测度论积分中。 ■

测度理论可以用来定义积分的一般概念，这种积分比经典微积分中所学的黎曼积分更加强大。最重要的概念之一是 Lebesgue 积分。积分区域的划分并不局限于区间，可以划分为任意可测集。定义要求可测函数以保证函数定义域划分为可测集。进一步学习内容见文献[348,549,839]。

5.1.4 使用正确的测度

因为可能还有其他一些度量和测度，所以有时似乎无法进行“正确”选择。这可能有点令人沮丧，因为基于采样的规划算法对测度的依赖性很强。对包含$SO(N)$的某些变换群来说，有一种自然测度，称为 Haar 测度。从 Haar 测度可以得到好的度量，但遗憾的是，对度量的选择仍然具有随意性。

基本要求是当用群的元素对集合进行变换时，测度不应该发生变化。正式地，令G表示一个实值矩阵群，μ表示G上的一个测度。如果对于任何可测子集$A\subseteq G$以及任何元素$g\in G$，有$\mu(A)=\mu(gA)=\mu(Ag)$，那么μ称为G的 Haar 测度[1]。gA表示通过乘积ga得到的所有矩阵的集合，$a\in A$。相似地，Ag表示所有ag形式的乘积。

例 5.13（$SO(2)$的 Haar 测度） 通过将旋转参数化为$[0,1]/\sim$，其中 0 和 1 是等同的，

1 这种测度从尺度上是唯一的，并对于任意局部紧致拓扑群来说这种测度是存在的。

并且令 μ 表示单位区间上的 Lebesgue 测度。那么就可得到 $SO(2)$的 Haar 测度，为更加清晰地看到这种不变性，考虑区间[1/4,1/2]，它产生旋转矩阵的一个集合 $A \subset SO(2)$。这对应于从 $\theta = \pi/2$ 到 $\theta = \pi$ 的所有旋转的集合。测度产生 $\mu(A)$=1/4。现在考虑用旋转矩阵 $g \in SO(2)$ 乘以每个矩阵 $a \in A$ 产生 Ag。假设 g 是 $\theta = \pi$ 时的旋转矩阵。集合 Ag 是从 $\theta = 3\pi/2$ 到 $\theta = 2\pi = 0$ 的所有旋转矩阵的集合。测度 $\mu(A)$=1/4 保持不变。gA 的不变性可以采用相似的方法进行检验。g 变换将区域平移到[0,1]/～。因为测度是基于区间长度的，所以它关于平移是不变的。注意到 μ 乘以一个固定常数（例如 2π）后并不影响不变性。

一个不变度量可以用 $SO(2)$上的 Haar 测度来定义。对任意点 x_1, $x_2 \in [0,1]$，令 $\rho = \mu([x_1, x_2])$，其中$[x_1, x_2]$是以 x_1 和 x_2 为端点的最短（最小测度）区间。这种度量已经在例 5.2 中给出过。

为了得到不是 Haar 测度的例子，令 μ 表示[0,1]上的概率质量，并定义任意非均匀分布概率密度函数（均匀分布产生 Haar 测度）。区间的任何移动都将改变概率质量，从而导致不同测度。

不使用 Haar 测度会导致 $SO(2)$的某些部分的权重比其他部分大。有时可能希望产生这种差异，但是知道如何消除该差异也非常重要。这些思路是显而易见的，但是在 $SO(3)$和一些其他群的情况下，很难消除这个差异并得到 Haar 测度。 ■

例 5.14（$SO(3)$的 Haar 测度） 对 $SO(3)$来说，需要采用四元数。如果采用单位四元数，那么 $SO(3)$用 $\mathbb{S}^3$ 进行参数化，但是对径点是等同的。可以看到 $\mathbb{S}^3$ 的表面积是 Haar 测度（因为 $\mathbb{S}^3$ 是三维流形，它可以被认为是表面“体积”）。从第 5.2.2 节可以看到 $SO(3)$上均匀随机采样必须利用 $\mathbb{S}^3$ 上的均匀概率密度。这就对应于 Haar 测度。如果 $SO(3)$用欧拉角来表示，将无法得到 Haar 测度。这无意中会引入一个差异；在没有特别原因的情况下，$SO(3)$中的一些旋转的权重比其他旋转的权重更大。 ■

5.2 采 样 理 论

5.2.1 目的与基本概念

运动规划的状态空间 C 是不可数无限的，所以最多只能在可数个采样点上考虑基于采样的规划算法。如果算法永远不停地运行，这可能是可数无限的，但是在实际情况下我们希望在仅考虑有限个采样点以后算法尽早结束。C 的势与集合之间的不匹配激发了研究人员对采样技术的研究。一旦定义了采样分量，那么就可以用第 2 章的离散规划方法，但是它们的性能依赖于对 C-空间的采样方式。

因为基于采样的规划算法经常被尽快终止，那么采样顺序的选择就显得非常关键。因此需要对采样集合和采样序列加以区别。从采样序列可以构造唯一的采样集合，但是从一个采样集合却能够构造许多序列。

稠密 考虑在 C 上构造一个无限采样序列。那么希望这个序列具备什么特征呢？最好是序列最终到达 C 中每个点，但是这是不可能的，因为 C 是不可数无限的。但这个序列可能到达距离 C 中每个元素任意近的地方（假设 $C \subseteq \mathbb{R}^m$）。在拓扑中这就是稠密的概念。令

U 和 V 是拓扑空间的任意子集。集合 U 在 V 中是稠密的，如果 $cl(U)=V$（回忆第 4.1.1 节中的集合的闭包）。这意味着给 U 增加边界点就产生 V。一个简单的例子是 $(0,1)\subset\mathbb{R}$ 在 $[0,1]\subset\mathbb{R}$ 中是稠密的。一个更加有趣的例子是有理数集合 $\mathbb{Q}$ 在 $\mathbb{R}$ 中是可数的、稠密的。为什么？对于任意实数，例如 $\pi\in\mathbb{R}$，存在一个分数序列收敛于它。这个分数序列必须是 $\mathbb{Q}$ 的一个子集。如果一个序列（不同于集合）的基集是稠密的，那么它也是稠密的。采样方法最少需要产生一个稠密序列。诸如采样的均匀性和规则性等更强的要求则在稍后进行解释。

一个随机序列可能是稠密的 假设 $C=[0,1]$。得到一个稠密序列的最简单的一种方法是随机进行采样。假设 $I\subset[0,1]$是一个长度为 e 的区间。如果独立随机地选取 k 个采样点[1]，那么它们都没有落入 I 的概率是 $(1-e)^k$。当 k 趋近于无穷时，这个概率趋于零。这意味着[0,1]上任何非零长度区间不包含一个采样点的概率趋于零。这里有一个技术问题，独立、随机采样得到的无穷序列以概率 1 稠密，而不同于确保稠密。这是概率论中处理不可数无穷序列的成果之一。例如，如果在[0,1]中随机选取一个数，那么取到 $\pi/4$ 的概率为 0；但是仍然是有可能选取到的（概率仅仅是 Lebergue 测度，零测度集合的 Lebergue 测度是零）。对于运动规划，这种技术性没有实际含义；但是如果 k 并不很大，那么仅仅得到概率保证，而不是绝对收敛，这并不令人满意。下面介绍的序列一定是稠密的，因为它是确定性的。

Van der Corput 序列 荷兰数学家 Van der Corput 于 1935 年给出这样一个序列[952]。它展示了一些对应用来说非常理想的品质，同时它的思路也非常简单。然而遗憾的是，它仅仅定义在单位区间上。将它的许多品质扩展到更高维数空间的要求激发产生了下面将看见的正式的品质测度和采样技术。

为了解释 Van der Corput 序列，令 $C=[0,1]/\sim$，其中 0～1 可以作为 $SO(2)$进行说明。假设我们希望将 16 个采样点放置在 C 中。一个理想的选择是集合 $S=\{i/16|0\leqslant i\leqslant 16\}$，该集合的点的间距为 1/16。这意味着 C 中任何点与最近的采样点之间的距离不会超过 1/32。如果我们希望产生一个包含 S 的序列，将如何做？最好的顺序是什么？如果我们不能保证 16 个点就足够了，怎么办？16 个点可能太少或者太多。

图 5.2 的前两列显示了通过将点以递增的顺序进行排序来试图将 S 包含在一个序列中。问题是若 i=8，那么会损失一半的 C。而我们希望对于任何 i，对 C 都有一个好的覆盖。Van der Corput 的思路是当序列用二进制数表示时，颠倒位的顺序。原始的顺序是最高位只转换一次，最低位在每一步都发生转换。颠倒顺序后，最高位在每一步都发生转换，这意味着序列在 C 的下半部和上半部交替采样。图 5.2 的第三列和第四列显示原始和颠倒顺序后的二进制表示。所得到的序列以很好的方式在[0,1]/～上跳跃，就像图 5.2 最后两列所示。令 $v(i)$表示 Van der Corput 序列的第 i 点。

与原始序列不同，每个 $v(i)$远离 $v(i+1)$。而且序列的前 i 点实现了对 C 的均匀覆盖。当 i 是 2 的幂时，点与点之间的间隔非常均匀。对其他 i，在长度为 l 的任意区间中出现的点数大致为 il，从这个意义上来说覆盖依然很好。例如，当 i =10，长度为 1/2 的每个区间大致包含 5 个点。

原始序列的长度 16 实际上并不重要。如果改为 8，同样可以得到 $v(1),\cdots, v(8)$。从图 5.2 颠倒的二进制中观察到，这等于从每个二进制表示中去掉最后一个 0，而不改变它们

1 参见第 9.1.2 节关于概率论的回顾。

i	原序列	二进制表示	颠倒顺序的二进制表示	Van der Corput序列	[0,1]/~中的点
1	0	.0000	.0000	0	
2	1/16	.0001	.1000	1/2	
3	1/8	.0010	.0100	1/4	
4	3/16	.0011	.1100	3/4	
5	1/4	.0100	.0010	1/8	
6	5/16	.0101	.1010	5/8	
7	3/8	.0110	.0110	3/8	
8	7/16	.0111	.1110	7/8	
9	1/2	.1000	.0001	1/16	
10	9/16	.1001	.1001	9/16	
11	5/8	.1010	.0101	5/16	
12	11/16	.1011	.1101	13/16	
13	3/4	.1100	.0011	3/16	
14	13/16	.1101	.1011	11/16	
15	7/8	.1110	.0111	7/16	
16	15/16	.1111	.1111	15/16	

图 5.2　Van der Corput 序列是通过颠倒原始序列的二进制表示中的比特位来得到的

的值。如果序列长度改为 32，那么前 16 个 $v(1),\cdots, v(16)$是相同的，并且序列从 $v(17)$继续到 $v(32)$。为了得到从 $v(33)$到 $v(64)$的 Van der Corput 序列，需要颠倒 6 比特的序列（对应于序列有 64 个点的情况）。过程不断重复进行就可以产生一个不需要提前指定固定点数的无穷序列。除了对每个 i 所应该具有的均匀特性外，无穷 Van der Corput 序列在[0,1]/～中也是稠密的。这意味着每个开子集必须至少包含一个采样点。

已经介绍了在单位区间中随机性地和确定性地生成采样点的方法。第 5.2.2 节～第 5.2.4 节将阐述如何在运动规划中出现的复杂空间中生成具有良好特性的稠密采样点。

5.2.2　随机采样

现在想象在[0,1]以外运动，并且为任何有界 C-空间 $C\subseteq\mathbb{R}^m$ 生成一个稠密采样序列。本部分的目的是产生均匀随机采样点。这意味着 C 上的概率密度函数 $p(q)$是均匀的。也意味着概率密度与 Haar 测度是一致的。我们不允许因为选择一个糟糕的参数表示而引入任何人为的差异。例如采用均匀随机欧拉角并不产生 $SO(3)$上的均匀随机采样点。但是采用均匀随机单位四元数却能够如此，因为四元数使用与 Haar 测度相同的参数表示；两者都在 $\mathbb{S}^3$ 上选点。

随机采样是 C 空间上所有采样方法中最容易的。一个重要原因是 C 空间形成于笛卡儿积，并且独立随机采样容易扩展到这些积。如果 $X=X_1\times X_2$，并且均匀随机采样点 x_1 和 x_2 分别从 X_1 和 X_2 中选取，那么(x_1, x_2)是 X 上的均匀随机采样点。这实现起来非常方便。例如假设运动规划问题涉及 15 个机器人，它们都平移任意$(x_i, y_i)\in[0,1]^2$；这产生 $C=[0,1]^{30}$。在这种情况下，从[0,1]可以随机均匀地选择 30 个点，并将这些点组合成一个 30 维向量。这样进行的采样点是 C 上的均匀随机分布。对非随机方法（确定性的）在笛卡儿积上将这些采样点进行组合是非常困难的，这些非随机方法将在第 5.2.3 节和第 5.2.4 节介绍。

生成 $SO(3)$的一个随机元素　在旋转空间上进行均匀采样必须特别仔细。概率密度必

须与 Haar 测度一致，这意味着随机旋转应该通过在 $\mathbb{S}^3$ 上随机选取一个点并形成单位四元数来得到。文献[46]给出了一个非常巧妙的方法来均匀随机采样 $SO(3)$，这里采用这种方法。随机均匀选择 3 个点 $u_1, u_2, u_3 \in [0,1]$。一个均匀、随机四元数由如下表达式给出：

$$h = \left(\sqrt{1-u_1}\sin 2\pi u_2, \sqrt{1-u_1}\cos 2\pi u_2, \sqrt{u_1}\sin 2\pi u_3, \sqrt{u_1}\cos 2\pi u_3\right) \tag{5.15}$$

文献[46]给出了对该方法的完整解释，这里给出简要直观说明。首先降低一维，选取 $u_1, u_2 \in [0,1]$ 生成 $\mathbb{S}^2$ 上的点。令 u_1 表示第三坐标的值 $(0,0,u_1) \in \mathbb{R}^3$。$\mathbb{S}^2$ 上 u_1 固定的点产生 $\mathbb{S}^2$ 上的一个圆，这个圆对应于 $\mathbb{S}^2$ 上某纬度线。第二个参数选择经度 $2\pi u_2$。遗憾的是，点在 $\mathbb{S}^2$ 上并不是均匀分布的。为什么？将 $\mathbb{S}^2$ 想象成一个球形面包的切片的皮。切片沿着平行于赤道的方向被切下来，它们是等厚度的。每个切片的皮的面积不同；因此点不是均匀分布的。但是对于 $\mathbb{S}^3$，三维外皮恰好面积相同；这可以通过计算面积分来解释。这意味着可以随机均匀地选择一个（无穷小）厚度为 u_1 的切片，其外皮上的点可以由 u_2 和 u_3 随机均匀地选择。对于 $\mathbb{S}^4$ 及以上维数的情况，外皮的测量会发生变化，即仅对于 $\mathbb{S}^3$ 才有这些结果。为了遵守旋转的对径点等同原则，任何在较低维半球中发现的四元数 h（例如 $a<0$）都可以取反得到 $-h$。这并不使采样点的均匀随机分布发生扭曲。

生成随机方向 一些基于采样的算法要求随机选择运动方向[1]。从一个位形 q，可能的运动方向可以想象成分布在球的周围。一个 $(n+1)$ 维 C 空间中，这对应于在 $\mathbb{S}^n$ 上进行采样。例如，在 $\mathbb{R}^2$ 中选择一个方向就等于选取 $\mathbb{S}^1$ 的一个元素；这可以参数化为 $\theta \in [0, 2\pi]/\sim$。如果 $n=4$，那么可以采用 $SO(3)$ 的方法。如果 $n=3$ 或者 $n>4$，那么采样点可以用稍微复杂一点的方法来生成，该方法利用了多维高斯密度函数的球对称性[343]。在 $\mathbb{R}^{n+1}$ 情况下对该方法进行解释；对其他空间必须考虑边界和等同。对 $n+1$ 维坐标，为每个坐标从具有相同方差的零均值高斯分布生成一个采样点 u_i。根据中心极限定理，通过在 $[-1,1]$ 上随机生成 k 个采样点并将它们相加可以近似得到 u_i（实际上 $k \geq 12$ 就足够了）。向量 $u_1, u_2, \cdots, u_{n+1}$ 给出了 $\mathbb{R}^{n+1}$ 中的一个随机方向，因为每个 u_i 都是独立取得的，并且所产生的概率密度函数的水平集是球形。我们不能使用 u_i 的均匀随机采样点，因为这将使方向向方体的角落偏移；而高斯分布却能产生球对称。最后一步是通过 $u_i/\|u\|$ 将向量归一化。

产生伪随机数 尽管均匀随机采样存在许多优点，但是它也有不足之处。这激发了对确定性方法的思考。因为存在折中，理解如何在运动规划中使用这两种方法是非常重要的。首要的问题是计算机生成的数不是随机的[2]。通常会用到一个伪随机数产生器，它是模拟随机行为的一种确定性方法。因为采样点不是真正随机的，那么也就无法将采样点扩展到笛卡儿积上。有时候问题是由不可预见的确定依赖关系所引起的。避免这种麻烦的最好的伪随机数产生器之一是 Mersenne twister[687]，在 Internet 上可以找到它的实现方法。

为了清楚看到一般的困难，文献[622,741]简要解释了经典的线性全等伪随机数产生器。该方法使用 3 个整数参数 M、a 和 c，这些参数由使用者进行选择。前两个必须是互质的，即 $\gcd(M, a)=1$，第三个参数必须满足 $0 \leq c \leq M$。使用求模，可以生成一个序列：

$$y_{i+1} = ay_i + c \mod M \tag{5.16}$$

---

1 当介绍平滑流形以后，在第 8.3.2 节将描述方向。在那种情况下，方向对应于有单位数量的可能速度的集合。目前，只给出非正式的方向概念。

2 有许多例外，它们将物理现象用作随机源[811]。

从任意种子 $1 \leqslant y_0 \leqslant M$ 开始。那么[0,1]之间的伪随机数可以由如下序列生成：

$$x_i = y_i / M \tag{5.17}$$

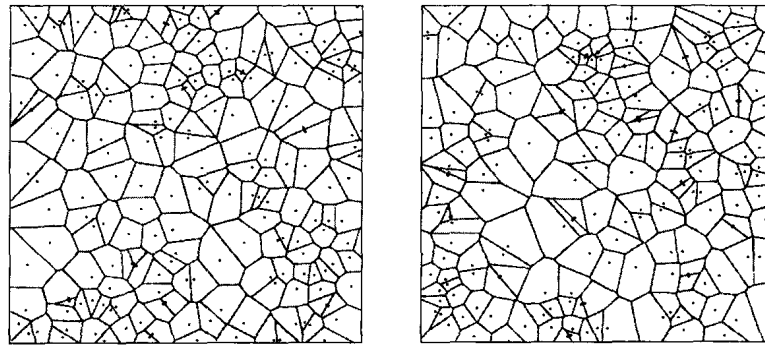
序列是周期性的；因此 M 通常非常大（例如 $M=2^{31}-1$）。由于周期性，采样点中存在出现规则性的问题，特别是当应用笛卡儿积来生成 $\mathbb{R}^n$ 中的点的时候。参数必须选择特定的值，用统计检验来从理论上或者实验上对采样点进行计算[741]。

检验随机性　认识到即使“随机”采样点也是确定性的这一点非常重要。对它们进行设计以优化统计检验的性能。用到一些均匀随机的复杂统计检验。这里描述的卡方检验就是最简单的一种。这种检验确定统计量的计算值距离它们的期望值有多远。作为一个简单的例子，假设 $C=[0,1]^2$，它被分割成 10×10 的 100 个正方形盒子。如果随机选取 k 个采样点的一个集合 P，那么直觉地，每个盒子应该接收大约 $k/100$ 个采样点。可以定义一个误差函数来测量这种直觉离真实情况有多远：

$$e(P) = \sum_{i=1}^{100} \left(b_i - k/100\right)^2 \tag{5.18}$$

其中 b_i 是落入第 i 个盒子的采样点的个数。可以看出 $e(P)$服从卡方分布[524]。一个惊人的事实是目标并不是最小化 $e(P)$。如果误差太小，我们就可以宣布采样点由于太均匀而不是随机的了。假设 $k=1\,000\,000$，这 100 个盒子中每个都有 10 000 个采样点。这产生 $e(P)=0$，但是发生这种情况的可能性有多大呢？考虑到由于随机性而引起的不规则性，误差通常必须比较大。

根据 Voronoi 图可以观察到这种不规则性，如图 5.3 所示。Voronoi 图将 $\mathbb{R}^2$ 分割成基于采样点的区域。每个采样点 x 有一个相应的 Voronoi 区域 $Vor(x)$。对于任意点 $y \in Vor(x)$，x 就是与 y 的欧氏距离最近的采样点。这些区域不同的大小和形状表明随机采样的不规则性。基于采样的运动规划可能并不希望有这种不规则性，这可以通过第 5.2.3 节和第 5.2.4 节的确定性采样方法加以修补（当然这些方法也有缺陷）。



(a) 196 点伪随机采样点　　(b) 196 点伪随机采样点

图 5.3　用 Voronoi 图可以很好地观察到（伪）随机采样点集的不规则性

5.2.3　低离散度采样

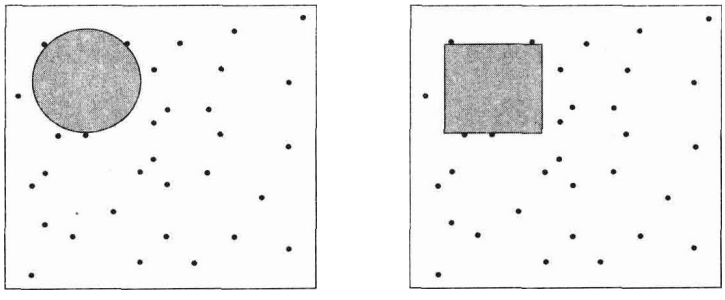
本节描述另外一种随机采样。目的是优化一个称为离散度的性能指标[741]。从直觉上看，放置采样点的方法是使最大的未覆盖区域尽可能得小。这是对栅格分辨率的推广。对

于一个栅格，通过定义每个轴的步长来选择分辨率。当步长降低时，分辨率增加。如果基于栅格的运动规划算法能够任意增加分辨率，那么它是分辨率完备的。使用本节的概念，它可以减小离散度来得到一个分辨率完备算法。因此离散度可以认为是对“分辨率”的一种有力的推广。

离散度 度量空间(X,ρ)中采样点的一个有限集合 P 的离散度[1]是[2]：

$$\delta(P)=\sup_{x\in X}\left\{\min_{p\in P}\{\rho(x,p)\}\right\} \tag{5.19}$$

图 5.4 阐述了两个不同度量的定义。考虑离散度的另外一种方法是采用最大空球的半径（对于 L_∞ 度量，球实际上是方体）。注意到在 X 的边界上（如果边界存在），空球被截断，因为它不能超过边界。根据 Voronoi 图可以很好地阐述这个问题。可以用图 5.3 来帮助解释 $\mathbb{R}^2$ 中的 L_2 离散度。Voronoi 顶点是这样一些点，在这些点上有三个或者三个以上 Voronoi 区域相交。这些点是 C 中的点，对于它们来说最近的采样点也很远。一个开的、空圆盘可以放置在任何 Voronoi 顶点上，它的半径等于与三个或者三个以上最近采样点的距离。放置在所有 Voronoi 顶点上的圆盘中最大圆盘的半径就是离散度。这种阐述也可以扩展到更高维数上。



(a) L_2 离散度　　(b) L_∞ 离散度

图 5.4 减小离散度意味着减小最大空球的半径

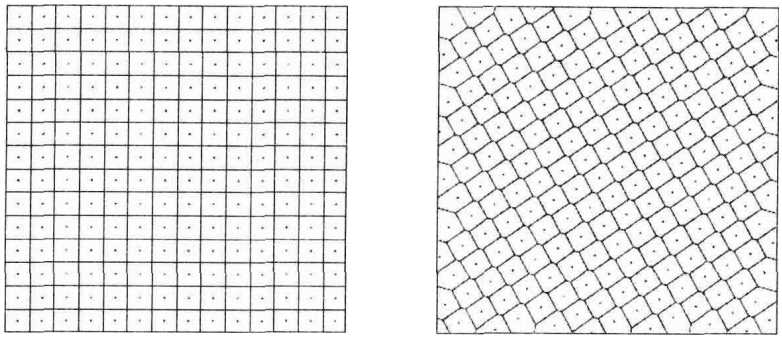
创建栅格 对离散度的优化迫使采样点在 C 上更均匀地分布。这使它们的统计检验失效，但是这些点的分布对运动规划来说却很好。如果 ρ 是 L_∞ 度量并且 $X=[0,1]^n$，那么考虑减小离散度的最好的方法。假设给定采样点数量 k。将区域[0,1]划分为方体的格，并将一个采样点放置在每个方体的中心，图 5.5(a)显示了 $n=2$ 和 $k=196$ 的情况，这样可以得到最优的离散度。每个轴上方体的个数必须为$[k^{\frac{1}{n}}]$，其中[•]表示取整。如果 $k^{\frac{1}{n}}$ 不是整数，那么剩余的点可以放置在任意地方而不影响离散度。注意到，对于 n 维情况下 k 个采样点的栅格，$k^{\frac{1}{n}}$ 仅仅给出每个轴上点的数量。所产生的栅格被称为 Sukharev 栅格[923]。

最有可能得到的是通过 Sukharev 栅格所得到的离散度。因此可以为 k 个采样点的任意集合 P 给出一个非常有用的下界[923]：

$$\delta(P)\geqslant\frac{1}{2\left[k^{\frac{1}{n}}\right]} \tag{5.20}$$

1 这个定义是从直觉产生而来的。低离散度意味着点散布得非常好。因此更大的离散度不好，这违背直觉。

2 Sup 表示上确界，它是最小上界。如果 X 是闭的，那么 sup 是一个最大值。详细内容请参见第 9.1.1 节。



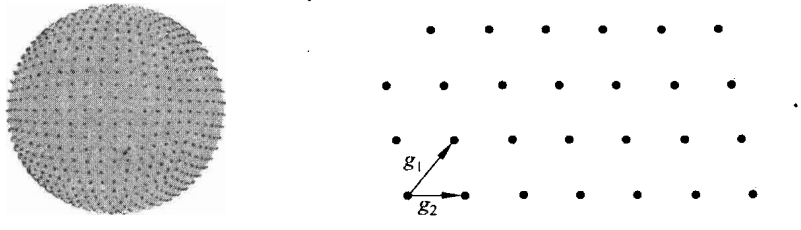
(a) 196 点 Sukharev 栅格　　(b) 196 个网格点

图 5.5　Sukharev 栅格和一个网格

这意味着如果要保持固定离散度，那么需要与维数 d 呈指数关系的点数。

现在读者可能会问为什么用 L_∞ 代替 L_2。这是因为 L_2 很难进行优化（除 $\mathbb{R}^2$ 以外，在 $\mathbb{R}^2$ 中可以构造等边三角形片，使得每个点都在三角形的中心）。对多数 k 值来说，确定一个最好的方法来在 $[0,1]^3$ 中分布固定数量的点这种简单问题都没有得到解决。对该问题的更进一步的处理方法见文献[243]。

假设现在考虑用其他拓扑来代替 $[0,1]^n$。令 $X=[0,1]/\sim$，其中等同产生了一个圆环。这不同于以前的情况，因为 X 不再有边界。Sukharev 栅格依然产生最优离散度，但是它也可以移动而不增加离散度。在这种情况下，也可以采用标准栅格，这种栅格和 Sukharev 栅格有相同数量的点，只是被平移到原点。因此第一个栅格点是(0,0)，该点通过等同与 2^n-1 个其他点相同。如果 X 表示一个圆柱，并且给定点的数量 k，那么最好只使用 Sukharev 栅格。但是有可能移动那些行为像 $\mathbb{S}^1$ 的每个坐标。如果 X 是矩形，而不是正方形，那么仍然可以通过用方体覆盖空间来构造一个好的栅格。在某些情况下也产生最优离散度。对诸如 $SO(3)$ 等复杂空间，这种意义下的栅格不存在。但是有可能在内接理想几何体的表面上生成栅格[253]，然后将采样点在较小扭曲情况下提升到 $\mathbb{S}^n$ [987]。例如在 $\mathbb{S}^2$ 上采样，Sukharev 栅格可以放置在一个方体的表面。然后提升它们得到如图 5.6(a)所示的弯曲栅格。



(a)　　(b)

图 5.6　(a) 通过将栅格放在一个内接方体的面上并将它们提升至曲面，这样就可以将一个变形的栅格放在球体和 $SO(3)$上；(b) 一个网格可以被认为是生成元不正交的栅格

例 5.15（Sukharev 栅格） 假设 $n=2$ 和 $k=9$。如果 $X=[0,1]^2$，那么 Sukharev 栅格可以为 9 种情况产生点，其中坐标可能是 1/6、1/2 或者 5/6。L_∞ 的离散度为 1/6。沿着每个轴，点与点之间的间隔为 1/3，这个值是离散度的两倍。如果 $X=[0,1]^2/\sim$，这表示一个圆环，那

么移动这 9 个点可以产生标准栅格。在这种情况下，每个坐标可能是 0，1/3 或者 2/3。离散度和点与点之间的间隔保持不变。■

栅格的一个良好特性是它们有格形结构。这意味着通过向量的加减可以很容易地得到相邻的点。令 g_j 为一个 n 维向量，它称为一个生成元。对于 n 个独立的生成元，格上的一个点可以表示为：

$$x=\sum_{j=1}^{n}k_j g_j \tag{5.21}$$

如图 5.6(b)所示。在二维栅格中，生成元表示“上”和“右”。如果 $X=[0,100]^2$，并使用间隔为整数的标准栅格，那么点(50,50)加(0,1)、(0, −1)、(−1,0)或者(1,0)就可以得到邻近点。在一般的网格中，生成元不需要正交，如图 5.5(b)所示。在第 5.4.2 节中，格形结构对定义搜索图来说非常重要和方便。

无穷栅格序列 现在假设采样点数 k 没有给定。我们的任务是定义一个具有良好 Van der Corput 序列特性、适合任意维数的无穷序列。这将变成多分辨率栅格。分辨率可以迭代加倍。对于 $\mathbb{R}^n$ 中一个多分辨率的标准栅格，序列首先将一个点放置在原点上。放置 2^n 个点后，栅格的每个轴有两个点。4^n 个点以后，每轴有 4 个点。因此 2^{ni} 个点以后（i 为任意正整数），栅格的每轴有 2^i 个点。如果只允许完整栅格，那么就非常清楚为什么它们不适于高维问题了。例如，如果 n=10，那么整个栅格在 1、2^{10}、2^{20}、2^{30}、…个采样点后才出现。分辨率加倍使点数增加 2^n 倍。因此为了在高维情况下使用栅格，那么必须接受部分栅格的概念，并且定义一个以良好方式放置采样点的无穷序列。

很容易扩展 Van der Corput 序列。假设 $X=\mathbb{T}^2=[0,1]^2/\sim$。原始的 Van der Corput 序列从二进制计数开始。最低位用来选择采样[0,1]的哪一半。在目前的情况下，两个最低位用来选择$[0,1]^2$ 的象限。紧接着的两位用来选择象限内的象限。递归进行这个过程在 $k=2^{2i}$ 个点后就得到一个完整栅格，i 为任意正整数。但是对任意 k，仅存在一个部分栅格。点以最优 L_∞离散度进行分布。通过使用二进制序列中的 n 个比特来选择 n 维象限，相同的思路就可以应用于 n 维情况。一些其他排序产生最优 L_∞离散度。选择优化诸如差异或者 L_2 离散度等其他指标的排序，见文献[642,647]。非常遗憾，很难构造多分辨率 Sukharev 栅格。如果指数的底由 2 变成 3，那么每 3^{ni} 个点后得到一个完整栅格。例如一维情况下，第一个点出现在 1/2 处。下两个点在 1/6 和 5/6 处出现。9 个点后出现下一个完整一维栅格。

离散度的界 因为采样序列是无穷的，考虑离散度的渐近的界是非常有趣的。我们知道对于 $X=[0,1]^n$ 和任何 L_p 度量，对 k 点 n 维情况，最可能的渐近离散度是 $O(k^{-1/n})$ [741]。在这种表达中，k 是极限中的变量，n 被当作一个常数。因此 n 的任何函数都作为一个常数出现（例如对于任何正 $f(n)$，$O(f(n)k^{-1/n})=O(k^{-1/n})$）。在渐近分析中一个重要的实际问题是 $f(n)$的大小。例如对于第 5.2.1 节中的 Van der Corput 序列，离散度的界是 $1/k$，即 $f(n)=1$。这似乎并不好，因为对于作为 2 的幂的 k 值来说，离散度是 $1/2k$。使用多分辨率 Sukharev 栅格，常数变为 3/2，因为在得到整个栅格之前需要花费大量的时间。存在 $f(n)=1/\ln 4$ 的非栅格化、低离散度的无穷序列[741]；但它们不是均匀分布的，这令人不可思议。

5.2.4　低差异采样

在一些应用中，不希望选择那些沿着坐标轴排列的点。因此采样理论得以进一步发展来确定当均匀分布采样点时避免排列成直线的方法。在基于采样的运动规划中，栅格有时会产生意想不到的行为，因为一行采样点可能会沿着 C_{free} 中的一个狭长区域进行排列。在某些情况下，用非常少的采样点就可以得到结果，但是在其他情况下，则需要太多的采样点。这些排成直线的问题在计算中通常产生更大的方差，因为很难预测它的影响。这促进了试图减小这种敏感性的采样技术的发展。

误差理论及其相应的采样方法进一步的发展来避免数值积分中的这些问题[741]。令 X 为诸如$[0,1]^n$这样的一个测度空间。令$\mathcal{R}$为被称为范围空间（range space）的 X 的一个子集族。在多数情况下，选择$\mathcal{R}$作为所有轴向排列的矩形子集的集合；因此从此都做这种假设。关于一个特定点集 P 和范围空间$\mathcal{R}$，k 点采样的差异[965]定义为（见图 5.7）：

$$D(P,\mathcal{R})=\sup_{R\in\mathcal{R}}\left\{\left|\frac{|P\cap R|}{k}-\frac{\mu(R)}{\mu(X)}\right|\right\} \tag{5.22}$$

其中$|P\cap R|$表示$P\cap R$中点的数量。上确界中的每一项都考虑了用 P 来估计 R 的体积的效果。例如，如果 $\mu(R)=1/5$，那么我们希望 P 中大约 1/5 的点落入 R。差异测量最大的体积估计误差，该误差可以在$\mathcal{R}$中所有集合上得到。

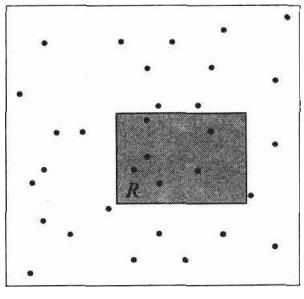


图 5.7　差异能够测量是否有相应数量的点落入盒子中；它与 χ^2 检验有关，只是在所有可能的盒子上进行了最优化

渐近界　根据范围空间和测度空间的不同，差异有许多不同的渐近界[685]。广泛使用的界是基于$[0,1]^n$中轴向矩形盒的标准范围空间。根据点数 k 是否给定，那么有两种不同的界。单个序列的最可能的渐近差异是 $O(k^{-1}\log^n k)$。这意味着不指定 k。但是如果对于每个 k，都选择一个新的点集，那么最可能的渐近差异是 $O(k^{-1}\log^{n-1}k)$。这个界更小，因为它考虑了点集序列所能够完成的最好的结果，其中每个点集可能是完全不同的。在单个序列中，下一个集合必须从目前集合通过加入一个采样点扩展而来。

差异和离散度的关系　因为球有正体积，所以基于测度的差异和基于度量的离散度之间有非常密切的关系。例如对于任何 $P\subset[0,1]^n$，

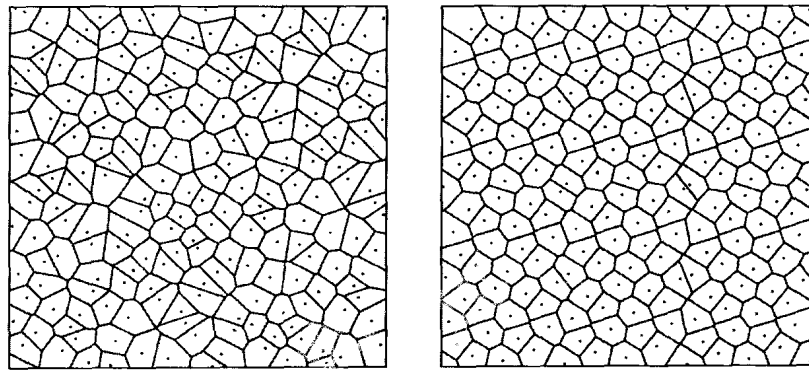
$$\delta(P,L_\infty)\leq D(P,\mathcal{R})^{1/d} \tag{5.23}$$

即低差异就意味着低离散度。但是相反的命题不成立。一个轴向排列的栅格因为沿着$\mathcal{R}$中集合的边界排成一行而产生高差异，但是离散度却非常低。即使低差异意味着低离散度，

但是通常在得到低离散度时并没有考虑差异（从而减少一个约束）。因此在应用中必须仔细考虑其中的折中权衡。

低差异采样方法 由于数值积分的重要性以及差异和离散度之间复杂的联系，大多数采样方面的书籍都能够产生低差异序列和点集[741,894,937]。尽管运动规划与积分大不相同，但是对这些经过仔细构造和分析的采样点进行评估却是很有意义的。它们在运动规划中的潜在应用与使用伪随机序列一样具有合理性，只是它们的目的不同（满足随机性的统计检验）。

低差异采样方法分为 3 种类型：（1）Halton/Hammersley 采样；（2）(t, s)-序列和(t, m, s)网；（3）网格。第一种类型代表了一种最早的方法，它是基于对 Van der Corput 序列的扩展。Halton 序列是对 Van der Corput 序列的 n 维扩展，但是没有用二进制表示，而是每个坐标使用不同的基[433]。在许多应用中，这种情况是对随机采样的一种合理的确定性替代。产生的差异（和离散度）比随机采样的低（以高的概率）。图 5.8(a)展示了 $\mathbb{R}^2$ 中前 196 个 Halton 点。



(a) 196个 Halton点　　(b) 196个 Hammersley点

图 5.8 很容易构造 Halton 点和 Hammersley 点，它们提供了随机采样以外的一种非常好的方法。比较图 5.3 中的 Voronoi 区域。需要注意的是尽管这些序列产生了渐近最优差异，但是在高维情况下它们的性能会大幅下降（例如超过十维）

选择 n 个互质的整数 $p_1, p_2, \cdots, p_n$（通常选择前 n 个质数 $p_1=2, p_2=3, \cdots$）。为了构造第 i 个采样点，考虑 i 采用以 p 为底的形式 $i=a_0+pa_1+p^2a_2+p^3a_3+\cdots$。[0,1]中后继点通过翻转位的顺序并移动小数点来得到（见图 5.2）：

$$r(i,p)=\frac{a_0}{p}+\frac{a_1}{p^2}+\frac{a_2}{p^3}+\frac{a_3}{p^4}+\cdots \tag{5.24}$$

对于 $p=2$，这产生 Van der Corput 序列中的第 i 个点。从 $i=0$ 开始，Halton 序列中第 i 个采样点是：

$$(r(i, p_1), r(i, p_2), \cdots, r(i, p_n)) \tag{5.25}$$

假设给定所需要的点数 k。在这种情况下，可以得到采样点的一个更好的分布。Hammersley 点集[434]是对 Halton 序列的改进。仅用 d–1 个质数并从 $i=0$ 开始，有 k 个元素的 Hammersley 点集中第 i 个采样点是：

$$(i/k, r(i, p_1), \cdots, r(i, p_{n-1})) \tag{5.26}$$

图 5.8(b)显示了 $n=2$ 和 $k=196$ 的 Hammersley 集。

Halton/Hammersley 采样点的构造非常简单和有效，所以应用非常广泛。它们都实现渐近最优差异；但是在它们的渐近分析中，常数增加得比维数的指数还要多[741]。离散度的常数呈指数增加，这比第 5.2.3 节中的情况更差。

分别使用(t, s)-序列和(t, m, s)网可以使常数得到提高改进[741]。关键的思路是迫使在$\mathcal{R}$的被称为规范矩形（canonical rectangles）的特殊子集上的差异为零，并且在$\mathcal{R}$中其他地方的一定范围的变动对差异的贡献小。得到广泛应用的(t, s)-序列是 Sobol 和 Faure 序列（见文献[741]）。Niederreiter-Xing(t, s)-序列有最著名的渐近常数$(a/n)^n$，其中 a 是一个小的正常数[742]。

第三种类型是网格，它可以认为是对栅格的推广[685,894,959]，它允许非正交轴。作为一个例子，考虑图 5.5(b)所显示的由如下技术生成的 196 个网格点。令 α 是一个正的无理数。对于一个固定的 k，根据$(i/k,\{i\alpha\})$生成第 i 个点，其中$\{\cdot\}$表示真值的小数部分（模 1 算法）。在图 5.5(b)中，$\alpha=(\sqrt{5}+1)/2$，黄金分割。通过选取 $n-1$ 个不同的无理数，以上过程可以推广到 n 维情况。通过不可约多项式的根来选择 α 参数的方法见文献[685]。网格中的第 i 个采样点是：

$$\left(\frac{i}{k},\{i\alpha_1\},\cdots,\{i\alpha_{n-1}\}\right) \tag{5.27}$$

最近的分析表明一些网格的集合能够实现的渐近差异非常接近于著名的非网格采样集合所实现的渐近差异[448,938]。因此将点限制网格上似乎对性能的影响很小或者没有影响，但是却增加了对路径规划非常有益的规则相邻结构。从历史上看，网格要求提前指定 k；但更多的研究致力于可扩展网格，这样的序列是无穷序列[449,938]。

5.3　碰 撞 检 测

一旦确定了采样点放置的位置，接下来的问题是确定位形是否存在碰撞冲突。因此碰撞检测是基于采样的规划中非常关键的组成部分。即使经常将它作为黑箱来对待，但是通过研究它的内部机理来理解它所提供的信息及其相应的计算代价是非常重要的。在多数运动规划应用中，绝大多数的计算时间花费在碰撞检测上。

有各种各样的碰撞检测方法，包括从具有良好计算复杂性的理论算法到在特殊应用中能够进行性能调整的启发式的实际算法。通过定义一个使用 C_{obs} 的几何模型的逻辑谓词，第 4.3 节介绍的技术可以用于碰撞检测算法。对于有凸机器人和障碍物的二维情况，这产生了一个线性时间碰撞检测算法。一般情况下，避免构造整个 C_{obs} 就可以有效地确定位形是否存在碰撞。

5.3.1　基本概念

如第 3.1.1 节所述，碰撞检测可以看作一个逻辑谓词。在目前情况下，它以这种形式出现：$\phi: C \rightarrow \{\text{TRUE, FALSE}\}$，其中定义域用 C 代替了 $\mathcal{W}$。若 $q \in C_{obs}$，则 $\phi(q)=\text{TRUE}$；否

则 $\phi(q)$ = FALSE。

两个集合的距离 对于布尔值函数 ϕ，没有关于机器人距离障碍物有多远的信息。这种信息在规划算法中非常重要。而距离函数能够提供这种信息，它定义为 d: $C \rightarrow [0, \infty]$，函数值表示在从 $\mathcal{A}(q)$ 到 $\mathcal{O}$ 的所有点对中最近点对之间的距离。一般情况下，对于 $\mathbb{R}^n$ 中两个有界闭子集 E 和 F 来说，距离定义为：

$$\rho(E,F) = \min_{e \in E} \left\{ \min_{f \in F} \{ \| e - f \| \} \right\} \tag{5.28}$$

其中 $\| \cdot \|$ 是欧氏范数。如果 $E \cap F \neq \phi$，那么 $\rho(E, F)$=0。这里的方法可以用来计算距离或者只用来确定是否 $q \in C_{obs}$。在后一种情况下，计算速度非常块，因为需要计算的信息少。

两阶段碰撞检测 假设机器人是 m 个连杆 $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_m$ 的族，$\mathcal{O}$ 有 k 个连通分支。对这种复杂的情形，碰撞检测可以分两个阶段来进行。

（1）宽阶段：该阶段的任务是对相距遥远的物体之间避免大量计算。每个物体外都配置一个简单的包围盒，进行简单测试以避免复杂的碰撞检测，除非包围盒相互重叠。在某些情况下可以用散列法来大量减少不得不进行的重叠检验的包围盒对的数量[706]。对于由多个物体组成的机器人，必须提前指定需要考虑碰撞检测的物体对，如参见第 4.3.1 节。

（2）窄阶段：在该阶段，仔细检验每个物体对。该阶段的方法将在第 5.3.2 节和第 5.3.3 节描述。

5.3.2 层次法

本节假设对两个复杂的非凸物体 E 和 F 进行碰撞检测。每个物体可以是机器人或者障碍区域的一部分。它们是用三角形等类型的几何本原定义 $\mathbb{R}^2$ 或者 $\mathbb{R}^3$ 的子集。层次方法通常将每个物体剖分成一个树。树中的每个顶点表示包含物体某子集的一个包围区。根顶点的包围区包含整个物体。

通常有两个相反的准则来选择包围区的类型：

（1）区域应该尽可能紧密地匹配物体上的点。

（2）应该尽可能有效地对两个区域进行交集检验。

对 L 形状的物体的几种常见的选择如图 5.9 所示。

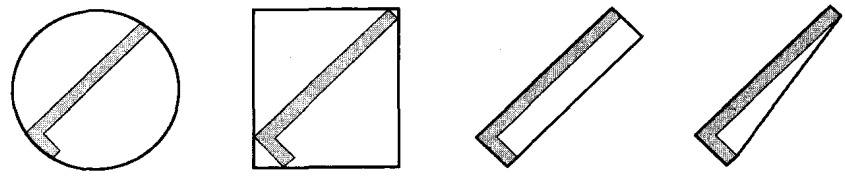


图 5.9 四种不同类型的包围区：(a) 球形；(b) 轴向包围盒（AABB）；(c) 定向包围盒（OBB）；(d) 凸壳。它们通常比前一种情况更加紧凑，但是检验起来开销更大，因为有相互重叠的点对

为物体 E（或者 F）递归地构造一个树。对每个顶点，考虑 E 中包含在包围区内的所有点的集合 X。通过定义两个更小的包围区来构造两个孩子顶点，两个区域的并集覆盖 X。

分割的原则是使每个孩子顶点所覆盖的部分有相似的大小。如果几何模型由三角形等本原组成，那么可以将三角形分割成两个包含大致相同数量的更小三角形的集合。计算每个孩子的包围区。图 5.10 显示了一个 L 形物体的分割的例子。孩子是通过分割而递归地生成，直到得到非常简单的集合。例如空间中有多个三角形，进行分割直到每个顶点代表一个三角形。在这种情况下，很容易检验两个三角形是否相交。

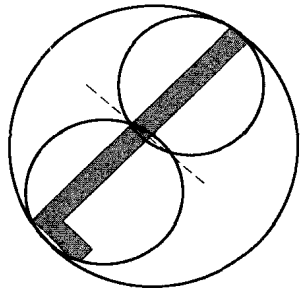


图 5.10　大圆是覆盖 L-形物体的一个顶点的包围区；沿着点线进行分割以后，两个较小的圆分别覆盖半个物体；每个圆对应于一个孩子顶点

考虑确定物体 E 和 F 是否碰撞的问题。假设已经分别为 E 和 F 构造了树 T_e 和 T_f。如果 T_e 和 T_f 的根顶点的包围区没有相交，那么不用额外的计算就可以知道它们没有发生碰撞。如果包围区相交，那么将 T_e 的孩子的包围区与 T_f 的包围区进行比较。如果相交，那么用 T_f 的孩子的包围区代替 T_f 的包围区，这个过程递归进行。只要包围区重叠，那么需要遍历树的更低级别的区域，直到最后到达树的叶子。如果几何模型采样三角形本原，那么在叶子的级别上算法对单个三角形的碰撞情况进行检验，而不是在整个包围区上。当遍历树的时候，如果 T_e 的一个顶点 v_1 的包围区与 T_f 的顶点 v_2 的包围区不相交，那么不需要将 v_1 的孩子与 v_2 的孩子进行比较。与朴素的检验所有三角形对的方法比较而言，层次方法通常可以显著减少比较的次数。

碰撞检测的层次方法可以扩展到进行距离的计算。当前所得到的最近的点对可以作为上界来排除掉一些尚未检测的点对。如果包围区之间的距离比所计算的最小距离大，那么就不必再检测它们的子包围区[641]。在这种情况下，还必须加入一个要求，每个包围区必须是它的父包围区的子集[810]。如果不需要距离信息，那么可以去掉这个要求。

5.3.3　增量法

本节介绍一种被称为增量距离计算的独特方法，它假设碰撞检测算法的相继两次调用之间物体运动的距离非常小。在这种假设条件下，对于凸多面体物体，算法实现的是“几乎固定的时刻”的性能[639,705]。非凸物体可以剖分成多个凸分支。

这些碰撞检测算法似乎为我们提供了性能良好的解决方案，但是这是有代价的。模型必须是相连接的，即所有的本原必须良好匹配。例如，如果一个二维模型使用线段，那么所有线段必须构成多边形。它们不是相互孤立的线段或者线段链。在三维情况下，要求多面体模型的面必须匹配形成三维形状的边界。模型不能是任意三维三角形族。

在作为 $\mathbb{R}^2$ 的凸子集的二维凸多边形情况下对该方法进行解释。凸多边形的 Voronoi 区

域用其要素来定义。要素集（feature set）是指凸多边形的所有顶点和边的集合。因此一个有 n 条边的多边形有 $2n$ 个要素。按照欧氏距离，相对于多边形外的任何一个点，都有一个最近的要素。对于一个给定的要素 F，在 $\mathbb{R}^2$ 中将 F 作为最近要素的所有点的集合称为 F 的 Voronoi 区域，记为 $Vor(F)$。图 5.11 显示了五边形的所有 10 个 Voronoi 区域。每个要素都可以认为是一个下面将要讨论的点集。

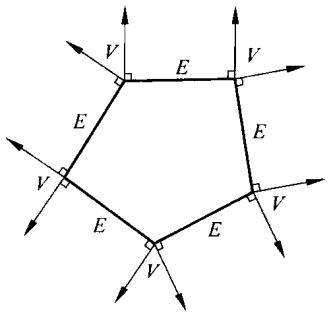


图 5.11 Voronoi 区域在基于边和基于顶点之间交替进行：顶点的 Voronoi 区域用“V”标注，边的 Voronoi 区域用“E”标注

对于任何两个不相交的凸多边形，它们之间最近的点由一个点对确定，该点对的点在其中一个多边形上（点是唯一的，除非有相互平行的边）。考虑最近对中每个点的要素。有 3 种可能的组合：

顶点-顶点 最近点对的每个点都是多边形的一个顶点。

边-顶点 最近点对的一个点在一条边上，另外一个点在一个顶点上。

边-边 最近点对的每个点都在一条边上。在这种情况下，边必须是平行的。

令 P_1 和 P_2 是两个凸多边形，令 F_1 和 F_2 表示任意要素对，一个要素来自于其中一个多边形。令 $(x_1, y_1) \in F_1$ 和 $(x_2, y_2) \in F_2$ 表示 F_1 和 F_2 的所有点对中最近的点对。如下条件意味着 (x_1, y_1) 和 (x_2, y_2) 之间的距离就是 P_1 和 P_2 之间的距离：

$$(x_1, y_1) \in Vol(F_2) \quad 且 \quad (x_2, y_2) \in Vol(F_1) \tag{5.29}$$

如果给定的要素对满足式（5.29），那么 P_1 和 P_2 之间的距离等于 F_1 和 F_2 之间的距离。这意味着 P_1 和 P_2 之间的距离可以在固定时间内得到。依然需要 P_1 相对于 P_2 移动很小距离这个假设来增加最近要素对保持不变的可能性。这就是为什么用“几乎固定的时刻”这个短语来描述算法的性能。当然最近要素对可能发生变化。在这种情况下，用上面介绍的条件对邻近要素进行检验，直到发现新的最近要素对。在这种最坏的情况下，搜索过程可能非常耗时，这与相继两次碰撞检测算法的调用之间物体运动的距离非常小这个假设相违背。

将二维的思路扩展到三维凸多面体[249,639,705]。根本的区别在于考虑的要素有三种：面、边和顶点。虽然情况变得更加复杂，但是思路是相同的。互为 Voronoi 区域这个条件依然成立，并且所产生的增量碰撞检测算法实现的是“几乎固定的时刻”的性能。

5.3.4 检验路径片段

碰撞检测算法用于确定一个位形是否在 C_{free} 中，但是运动规划算法要求整个路径都在 C_{free} 中。规划器与碰撞检测之间的接合点通常涉及确认路径段。不能逐点地进行检验，因

为那要不可数无限次地调用碰撞检测算法。

假设需要检验路径 $\tau:[0,1]\to\mathcal{C}$是否 $\tau([0,1])\subset\mathcal{C}_{free}$。一个常见的方法是对区间[0,1]进行采样，对采样点进行碰撞检验。需要什么样的采样分辨率呢？怎样保证没有采样到的地方是无碰撞的呢？对这些问题，既有实际的解答，又有理论的解答。实际上，通常选择一个固定值 $\Delta q>0$ 作为 $\mathcal{C}$-空间的步长。选择距离足够近的两个点 $t_1, t_2\in[0,1]$来保证 $\rho(\tau(t_1),\tau(t_2))\leqslant\Delta q$，其中 ρ 是$\mathcal{C}$上的度量。Δq 的值通常用实验方法来确定。如果 Δq 太小，那么碰撞检验需要花费大量时间。如果 Δq 太大，那么机器人可能会跳过一个小的障碍物。

凭经验设置的 Δq 可能不太会令人满意。幸运的是，有许多算法来验证一条路径是无碰撞的。在某些应用中并不采用这些方法，因为它们实现起来需要一定的技巧，并且性能往往不佳。因此这里将同时介绍这两种方法，读者根据具体情况和个人喜好自行决定使用哪一个。

确保 $\tau([0,1])\subset\mathcal{C}_{free}$，这要求使用距离信息以及在 $\mathbb{R}$ 中$\mathcal{A}$上的点所能够运动的距离的界限。界限可以通过使用例 5.6 的机器人位移度量来得到。在表达一般情况之前，我们首先来解释在 $\mathcal{W}=\mathbb{R}^2$ 平移和旋转的刚体机器人。令 $x_t, y_t\in\mathbb{R}^2$ 和 $\theta\in[0,2\pi]/\sim$。假设碰撞检测算法指出$\mathcal{A}(q)$距离$\mathcal{W}$中障碍物至少 d 个单位。该信息可以用来确定$\mathcal{C}_{free}$中包含 q 的一个区域。假设沿τ被检验的下一个候选位形是 q'。如果沿着τ从 q 到 q'，$\mathcal{A}$上的点的运动距离都没有超过 d，那么 q'以及 q 和 q' 之间的所有位形都一定是无碰撞的。这假设在从 q 到 q'的路径上，对于第 i 个坐标和从 1 到 n 的任意 i，每个访问过的位形必须在 q_i和 q_i' 之间。如果机器人采取 q 和 q'之间的任意路径，那么无法做出这样的保证。

当$\mathcal{A}$进行平移时，所有点运动相同的距离。但是对于旋转，点所运动的距离依赖于该点离旋转中心(0,0)有多远。令 $a_r=(x_r, y_r)$表示$\mathcal{A}$上的一个有最大值 $r=\sqrt{x_r^2+y_r^2}$ 的点。图 5.12 显示了一个例子。一个变换点 $a\in\mathcal{A}$可以用 $a(x_t, y_t, \theta)$ 表示。如果机器人从 θ 旋转到 θ'，那么对于任何 $a\in\mathcal{A}$，可以得到如下的界：

$$\|a(x_t,y_t,\theta)-a(x_t,y_t,\theta')\|\leqslant\|a_r(x_t,y_t,\theta)-a_r(x_t,y_t,\theta')\|\leqslant r|\theta-\theta'| \tag{5.30}$$

其中假设在 θ 和 θ' 之间进行插值得到 C 中的路径（使用$\mathbb{S}^1$中 θ 和 θ' 之间的最短路径）。因此如果$\mathcal{A}(q)$距离障碍物至少 d，那么方向的改变不会引起碰撞，只要 $r|\theta-\theta'|<d$。注意到这是一个宽松的上界，因为 a_r沿着圆弧运动，它的位移不会超过 $2\pi r$。

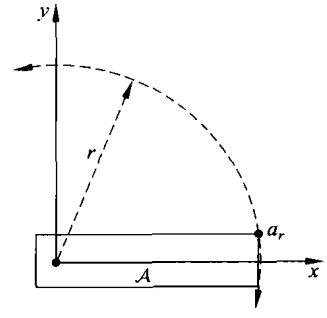


图 5.12　当旋转$\mathcal{A}$时，$\mathcal{A}$上离原点最远的点运动最快；如果 x_t和 y_t固定，那么它最多只能位移 $2\pi r$

相似地，x_t和 y_t都可能单独变化到 d，产生$|x_t-x_t'|<d$ 和$|y_t-y_t'|<d$。如果所有 3 个参

数同时改变，那么可以在 C_{free} 中定义一个区域：

$$\{(x_t', y_t', \theta') \in C \| x_t - x_t'| + |y_t - y_t'| + r|\theta - \theta'| < d\} \tag{5.31}$$

这个界通常可以用来设置碰撞检验的步长 Δq，该步长保证中间点在 C_{free} 中。根据 d 和路径的方向[1]，用到的特定步长值可能会发生改变。

对于 $SO(3)$，$\mathcal{A}$ 上有最大 r 的点的位移是有界的。在这种情况下，最好用四元数差分 $\|h-h'\|$ 来表示这个界。也可以采用欧拉角来对式（5.31）进行一般化，式（5.31）有六项，其中三个平移项、三个旋转项。对三个旋转项，必须选择垂直于旋转轴的平面内有最大 r 的点。

如果有多个连杆，确定步长就变得更加复杂。每个点 $a \in \mathcal{A}_i$ 被第 3.3 节和第 3.4 节中基于运动学表达式的非线性函数所变换。令 a: $C \to \mathcal{W}$ 表示这种变换。在某些情况下，可能导出一个如下形式的 *Lipschitz* 条件：

$$\|a(q) - a(q')\| < c\|q - q'\| \tag{5.32}$$

其中 $c \in (0, \infty)$ 是一个固定常数，a 是 $\mathcal{A}_i$ 上的任意一点，并且该表达式对于任何 $q, q' \in C$ 都成立。目的是使 *Lipschitz* 常数 c 尽可能小；这使 q 的变化范围更大。

更好的方法是单独地界限连杆关于每个参数的位移的范围：

$$\|a(q_1, \cdots, q_{i-1}, q_i, q_{i+1}, \cdots, q_n) - a(q_1, \cdots, q_{i-1}, q_i', q_{i+1}, \cdots, q_n)\| < c_i |q_i - q_i'| \tag{5.33}$$

来得到 *Lipschitz* 常数 $c_1, \cdots, c_n$。机器人位移的界限变为：

$$\|a(q) - a(q')\| < \sum_{i=1}^{n} c_i |q_i - q_i'| \tag{5.34}$$

通过一个长链我们可以看到使用单个参数界限的好处。考虑 $\mathbb{R}^2$ 中 50 个连杆的链，并且每个连杆长度为 10。C-空间是 $\mathbb{T}^{50}$，它可以参数化为 $[0,2\pi]^{50}/\sim$。假设链处在直线位形下（对于所有 $1 \leqslant i \leqslant 50$，$\theta_i = 0$），这意味着最后一个点在 $(500,0) \in \mathcal{W}$。第一个连杆的方向 θ_1 的改变引起 $\mathcal{A}_{50}$ 的巨大移动。但是 θ_{50} 的改变引起 $\mathcal{A}_{50}$ 的移动却很小。因此对于每个 $1 \leqslant i \leqslant 50$，选取不同的 Δq_i 是有益的。在这个例子中，与 $\Delta\theta_{50}$ 比较而言，$\Delta\theta_1$ 应该用一个较小的值。

非常遗憾，还有更多的复杂性。假设这个链处在折叠状态（对于每个 $1 \leqslant i \leqslant 50$，$\theta_i = \pi$）。在这种情况下，当 θ_1 受到扰动时，$\mathcal{A}_{50}$ 的移动没有直线位形那样快。这种位形下相对于 C 的其他部分而言，θ_1 可以使用一个较大的步长。这意味着尽管 *Lipschitz* 常数在所有 C 上都成立，但是仍然希望在 $q \in C$ 周围的局部区域中确定一个更好的界。通过分析诸如式（3.53）和式（3.57）这样的变换的 *Jacobian* 矩阵，可以得到一个线性方法。

当检验一条路径时，另一个需要关注的问题是检验采样点的顺序。为简单起见，假设 Δq 是常数，并假设路径用定常速度表示的。碰撞检验应该从 0 到 1 进行吗？实验显示最好使用递归的二叉策略[381]。如果路径无碰撞，这没有什么区别，但是如果路径有碰撞时，它能够节省时间。这是一种在[0,1]上采样的问题，Van der Corput 序列 v 讨论的就是这个问题。图 5.2 中最后一列精确指出沿着路径的每一步在哪里进行检验。开始时检验 $\tau(1)$。然后按这种顺序检验 Van der Corput 序列中的点：$\tau(0)$, $\tau(1/2)$, $\tau(1/4)$, $\tau(3/4)$, $\tau(1/8)$, $\cdots$。如果发现碰撞或者离散度低于 Δq，那么过程终止。如果 Δq 不是常数，那么在 q 中变化范围允许较大的区域中可能会跳过 v 的一些点。

1 为了正式的讨论方向，最好在C上定义一个可微结构。这将在第 8.3 节进行，在那里这似乎是不可避免的。

5.4　递增采样与搜索

5.4.1　一般架构

第 5.4 节和第 5.5 节的算法遵循的是单疑问模型，即一旦每个机器人和障碍物设定，那么(q_I, q_G)就给定了。这意味着预计算并没有什么优势，并且基于采样的运动规划问题可以被认为是一种搜索过程。多疑问模型将在第 5.6 节介绍。

本节提出的基于采样的规划算法与第 2.24 节概述的搜索算法族极其相似。主要的区别在于下面的第（3）步，其中应用行动 u 用生成一个路径片 τ_s 所代替。另外的区别在于搜索图 $\mathcal{G}$ 是无方向的，其中的边代表路径，而方向图中的边代表行动。为由路径的族组成的运动规划定义一个行动空间可能使得这些看起来相似，但是这里尽量避免如此。在有微分约束的运动规划中，实际上要求这样，参见第 14 章。

多数单疑问、基于采样的运动规划算法采用如下模板：

（1）初始化：令 $\mathcal{G}(V, E)$ 表示一个无方向的搜索图，对该图来说，V 至少包含一个顶点，E 不包含边。典型地，V 包含 q_I、q_G 或者两者都有。一般情况下，可能还包括 C_{free} 中的其他点。

（2）顶点选择方法（VSM）：为进行扩展而选择一个顶点 $q_{cur} \in V$。

（3）局部规划方法（LPM）：对某 $q_{new} \in C_{free}$，它可能用 V 中的顶点表示，也可能不用，它试图构造一条路径 τ_s:[0,1]→C_{free} 以使 $\tau(0)=q_{cur}$ 和 $\tau(1)=q_{new}$。采用第 5.3.4 节的方法，必须检验 τ_s 以确保它不引起碰撞。如果这一步没有产生无碰撞路径片，那么回到步骤（2）。

（4）将边插入图：将 τ_s 插入 E，作为从 q_{cur} 到 q_{new} 的边。如果 q_{new} 不在 V 中，那么将它插入。

（5）对解进行检验：确定 $\mathcal{G}$ 是否对一条路径解进行了编码。就像离散情况，如果只有一棵搜索树，那么这很容易；否则，检验过程非常复杂，开销是昂贵的。

（6）返回步骤（2）：迭代直到发现一个解或者某终止条件得到满足，如果过程终止，那么算法报告搜索失败。

目前，$\mathcal{G}$ 是一个拓扑图，如例 4.6 所定义。每个顶点是一个位形，每条边是一条连接两个位形的路径。本章中，在不引起混淆的情况下，它将被简单地当作一个图。一些作者称之为路线图（roadmap）；对于包含足够多路径以使运动规划问题容易解决的一个图，我们保留路线图这个术语。这种情况将在第 5.6 节和第 6 章中进行讨论。

变化步骤（2）和步骤（3）就可以描述一族基于采样的规划算法。其他步骤也可以改变，但是这不太重要，并会在相应的地方被描述。为方便起见，步骤（2）被称为顶点选择方法（VSM），步骤（3）被称为局部规划方法（LPM）。VSM 的作用与第 2.2.1 节中的优先级队列 Q 很相似。LPM 的作用是计算可以加入到图中的无碰撞路径片段。它之所以称为局部的，是因为路径片段通常比较简单（例如最短路径），并且运行的距离短。它不是全局的，原因在于 LPM 不试图解决整个规划问题；LPM 可能经常得不到想要的路径片段。

稍后我们将会表述它，但是现在暂时假设通过用高分辨率的栅格近似 C，第 2.2 节中

的任何搜索算法都可以被应用到运动规划中。那么所产生的问题看起来像对例 2.1 的多维扩展（C_{obs} 形成 labyrinth 墙）。对于高维空间中的高分辨率的栅格来说，大多数经典离散搜索算法都有陷入局部最小值的问题。可能有大量的位形落入 C_{obs} 的凹形区域，必须逃离这些区域才能解决规划问题，如图 5.13(a)所示。假设 C 空间障碍物是一个巨大的“碗”，位形可能陷入其中。这个图形用二维绘制，但是假设 C 有许多维，例如 $SE(3)$的六维或者一个连杆组可能有几十维数。如果将第 2.2 节中的离散规划算法应用于对 C 的高分辨率栅格近似，那么它们在逃离凹形区域到达 q_G 以前的大部分时间里都在碗中徘徊。对于 n 维 C 空间，碗中栅格点数的数量级通常是 100^n。因此基于采样的运动规划算法以克服这种困难的方式将采样方法和搜索方法结合起来。

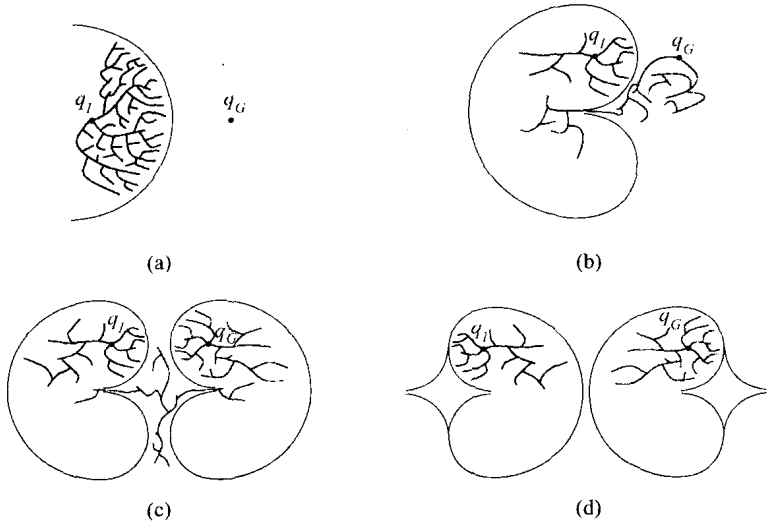


图 5.13 这些图都刻画的是 C-空间中的高维障碍区域；(a) 搜索一定涉及某种类型的多分辨率问题，否则算法可能探测太多的点；(b) 有时问题就像一个陷阱，在这种情况下，双向搜索能够提供帮助；(c) 对于两个陷阱，可能需要多向搜索；(d) 这个例子甚至用多向搜索也很难求解

就像离散搜索算法的情形一样，基于搜索树的数量多少，有几种类型的算法。

单向（单树）方法：在这种情况下，规划与图 2.4 中的离散前向搜索方法非常相似。主要的区别在于如何实现 VSM 和 LPM。图 5.13(b)显示了一个“bug 陷阱”[1]的例子，用前向搜索方法解决这个例子有很大困难；但是如果规划器融合一些贪恋的、最好优先的行为，那么用后向搜索方法解决这个问题可能并不困难。这个例子的高维情况可以认为是“bug 陷阱”的一种。为了跳出陷阱，必须发现一条从 q_I 到狭窄出口的路径。假设一只苍蝇在高维陷阱中飞行。在二维情况下发现出口并不困难，但是如果它关于每个位形参数的变化范围非常小，那么发现出口几乎是不可能的。“火山”顶端与 bug 陷阱的其他部分比较而言非常小。这个例子促使我们研究双向算法。一个从 q_G 开始的搜索树可能更容易到达陷阱。

双向（双树）方法：因为不知道 q_I 或者 q_G 是否处在陷阱（或者另外一个具有挑战性的区域）中，双向搜索方法通常是可取的。从直觉上讲，与以 q_I 为中心、必须到达 q_G 的单个

1 这个原理在实际生活中用来捕获飞行中的昆虫。这种类似是由 James O'Brien 在与 James Kuffer 的讨论中提出的。

波前比较而言，分别以 q_I 和 q_G 为中心的两个传播波前在覆盖较小区域以后就能够相遇。当选择顶点时，双向搜索通过定义 VSM 来交替进行前向和后向搜索，LPM 有时会生成探索 C_{free} 的新的部分的路径，有时则试图生成一条连接两个树的路径。

多向（多树）方法：如果问题存在两个陷阱，如图 5.13(c)所示，那么从其他地方生长搜索树是很有意义的，期望从其他方向上有更好的机会进入陷阱。但是这也使树与树之间的连接更加复杂。在可能的连接中，应该选择哪两个树？相同的树对应该多长时间选择一次？应该选择哪个顶点对？在实际情况下，一些启发式的参数可以回答这些问题。

当然也可以构造图 5.13(d)的例子使所有基于采样的规划算法都无效。还可以产生更难的情况，其中必须定位和遍历一系列狭窄的通道。我们必须接受这样一个事实，那就是有些问题用基于采样的规划方法是根本无法解决的，除非可以利用问题的一些特定结构。

5.4.2　改进离散搜索算法

产生基于采样的规划算法最方便的方法是在 C 上定义一个栅格，并用第 2.2 节中的算法引导离散搜索。所产生的规划问题看起来与例 2.1 非常相似。每条边对应于 C_{free} 中的一条路径。一些边可能因为碰撞而不存在，但是这些边不得不在搜索过程中逐步显现出来，因为显式表示 C_{obs} 的开销太大而无法构造。

假设 n 维 C 空间用单位方体 $C=[0,1]^n/\sim$ 表示，其中～表明：方体的侧面的等同是用来反映 C 空间拓扑的。将 C 表示成一个单位方体通常需要再次参数化。例如将一个角度 $\theta\in[0, 2\pi)$ 用 $\theta/2\pi$ 代替，使其范围限制在[0,1]。如果 $SO(3)$ 采用四元数，那么 $\mathbb{S}^3$ 的上半部分必须变形为 $[0,1]^3/\sim$。

离散化　假设 C 用分辨率 $k_1, k_2, \cdots, k_n$ 进行离散化，其中 k_i 是正整数。对于每个 C 空间坐标来说，这允许不同的分辨率。可以用标准栅格或者 Sukharev 栅格。令：

$$\Delta q_i=[0 \quad \cdots \quad 0 \quad 1/k_i \quad 0 \quad \cdots \quad 0] \tag{5.35}$$

其中前 i–1个分量和最后 n–i 个分量是 0。一个栅格点就是一个位形 $q\in C$，它可以用如下形式表示[1]：

$$\sum_{i=1}^{n} j_i \Delta q_i \tag{5.36}$$

其中每个 $j_i\in\{0,1,\cdots, k_i\}$。可将整数 $j_1, \cdots, j_n$ 想象成栅格的阵列索引。$j_i=0$ 或 $j_i=k_i$ 的栅格点称为边界栅格点。因为等同，用式（5.36）表示的边界栅格点可能有多个。

邻域　对每一个栅格点 q，需要定义邻近栅格点的集合，对于这些点来说可以构造一条边。必须给出特殊情况来定义边界栅格点的邻域来确保不违背等同和 C 空间边界（如果存在）。如果 q 不是边界栅格点，那么 1-邻域定义为：

$$N_1(q)=\{q+\Delta q_1,\cdots,q+\Delta q_n,q-\Delta q_1,\cdots,q-\Delta q_n\} \tag{5.37}$$

对于 n 维 C 空间，至多有 $2n$ 个 1-邻近点。在二维情况下，这至多产生 4 个 1-邻近点，它们可以作为“上”、“下”、“左”、“右”。至多产生 4 个 1-邻近点是因为一些方向可能被障碍区域阻挡。

1　也可以用式（5.21）中的一般网格定义。

2-邻域定义为：

$$N_2(q)=\left\{q\pm\Delta q_i\pm\Delta q_j \,\middle|\, 1\leqslant i,j\leqslant n, i\neq j\right\}\cup N_1(q) \tag{5.38}$$

相似地，对任何正整数 $k\leqslant n$，可以定义 k-邻域。对应一个 n-邻域，至多存在 3^n-1 个邻近点；也可能因为边界或者碰撞的缘故邻近点会更少。这些定义很容易扩展到处理边界点。

离散规划问题 一旦定义了栅格和邻域，那么就可以得到离散规划问题。图 5.14 描述了构造一个问题的过程，该问题在$[0,1]^2$中有 9 个 Sukharev 栅格点。采用 1-邻域，搜索图 $\mathcal{G}(V,E)$中潜在的边如图 5.14(a)所示。$\mathcal{G}$是一个拓扑图，就像例 4.6 中定义的，因为每个顶点是一个位形，每条边是一条路径。如果 q_I 和 q_G 与栅格点不一致，那么需要将它们连接到一些邻近的栅格点，如图 5.14(b)所示。q_I 和 q_G 应该连接到哪些栅格点？一般原则是，如果采用 k-邻域，那么应该将 q_I 和 q_G 连接到这样的栅格点：它们至少与一个典型栅格点的最远 k-邻近点一样近。

如图 5.14(b)所示的所有顶点和边通常不出现在$\mathcal{G}$中，因为有些与$\mathcal{C}_{obs}$ 相交。图 5.14(c)显示了一种更典型的情况，其中一些潜在顶点和边因为冲突而被删除。这种表示可以通过检验所有潜在顶点和边是否冲突来提前进行计算。这产生了一个适用于多疑问问题的路线图，这将在第 5.6 节介绍。本节假设$\mathcal{G}$是在搜索过程中逐步展现出来的。这与第 2.2 节介绍的离散规划方法情形相同。目前，$\mathcal{G}$的潜在的边可在搜索过程中得到验证。通过定义 k-邻域来给出候选的边。在搜索过程中，任何已经进行碰撞检测的边或者顶点将显式出现在数据结构中，以后就不需要再次检测了。搜索过程结束时就发现了一条路径，如图 5.14(d)所示。

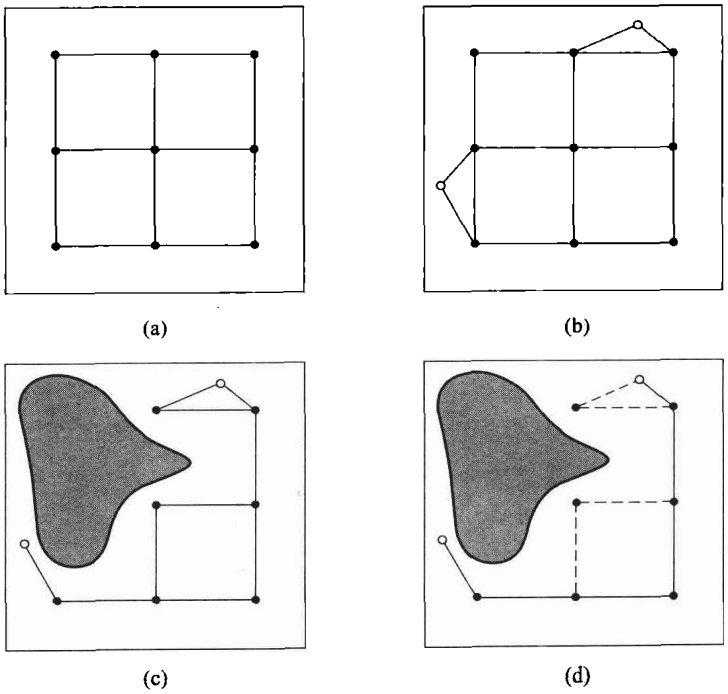


图 5.14 拓扑图可以在搜索中进行构造，这样用较少的采样点就可以解决运动规划问题

栅格分辨率的问题　如果分辨率合适，到目前为止所介绍的方法就能够发现问题的解。如果每个轴的点数太多，那么搜索速度就可能太慢。如果在每个轴上选择较少的点，那么解可能丢失。这种折中是基于采样的运动规划的基础。一般情况下，如果采用其他形式的采样和邻域，那么不得不生成足够多的点来产生足够小的离散度。

有两种一般方法可以避免选择分辨率（或者离散度）：

（1）不断改进分辨率直到发现问题的解。在这种情况下，采样和搜索是交替进行的。一个重要的变量是两个过程以什么频度交替进行。这将在稍后予以介绍。

（2）另外一个方法是放弃对离散搜索算法的改进，直接对连续问题的算法进行研究。这形成了第 5.4.3 节、第 5.4.4 节和第 5.5 节的基础。

最简单的方法是进行迭代来提高栅格的分辨率。假设初始时采用诸如最好优先或者 A^* 等离散搜索算法搜索总共有 2^n 个点、每个轴两个点的标准栅格。如果搜索失败，那么应该怎么办？一种可能是将分辨率加倍，产生有 4^n 个点的栅格。一些边可以重复使用；但是在高维情况下，这种节省将很快消失。一旦分辨率加倍，搜索过程就会再次进行。如果又一次失败，那么分辨率再次加倍产生 8^n 个点。对于每个 i，整个栅格一般有 2^{ni} 个点。问题是如果 n 很大，那么点数增长速度太快。例如 n=10，那么初始时有 1024 个点，如果失败，下一次的搜索点数将多达一百多万个。如果再次失败，那么将要花费非常长的时间到达有 2^{30} 个点的下一级分辨率。

希望有一个类似于第 2.2.2 节迭代深化的方法。放弃以前的对分辨率所做的努力，为每次迭代 i 产生每个轴有 i^n 个点的栅格。这产生的栅格的大小为 2^n、3^n、4^n、…。搜索一个更大的栅格所花费的时间并不比搜索更小分辨率的栅格所用时间多多少。因此放弃以前的工作是无害的。

一个更好的解决方法是不要求在进行搜索之前存在完整栅格。例如在再次搜索之前，为每个轴增加分辨率。更好的情况可能是搜索过程和采样过程紧凑交替进行。例如，采样点是无穷的、稠密的序列 α，每增加 100 个点后进行图的搜索，假设即使栅格仅有部分完成，也能够定义或者构造邻域。如果搜索过程进行得太频繁，那么搜索过程将决定运行时间。一个简单而且更加有效率的方法是应用 union-find 算法[245,826]来迭代跟踪 $\mathcal{G}$ 中的连通分支，而不是进行显式搜索。如果 q_I 和 q_G 变成相同连通分支的一部分，那么就发现了一个路径解。每次序列 α 中加入一个新的点，union-find 算法的“搜索”过程几乎[1]在固定时间内完成。这紧凑地交替进行搜索过程和采样过程，所产生的基于采样的算法并不需要分辨率参数。最好是选择一个包含一些网格结构的序列 α，这有利于在每次迭代中确定邻域。

如果在开始时我们就断言分辨率非常非常高，那么怎么办？假设栅格中有 100^n 个点。那么搜索算法的负担太重。如果搜索算法自身善于避免局部最小，并且有内建多分辨率性质，那么它可能不需要采样的迭代改进。第 5.4.3 节的方法就是基于这种思路在高分辨率栅格上实现最好优先搜索，并且组合了随机游走方法以避免局部最小。第 5.5 节的算法向前更进一步，它用多分辨率的方法进行搜索而不要求显式确定分辨率和邻域。这可以认为是当每个轴上的点数趋近无穷时的极限情况。

1　它不是常数，因为运行时间与 Ackerman 函数的反函数成比例，时间增长得非常慢。从实际应用角度，可以认为算法是在固定时间内进行的，参见第 6.5.2 节。

尽管本节着重于栅格，当然也可以采用第 5.2 节中其他形式的采样。这要求用对本节的 k-邻域进行一般化的方法来定义邻域。在每种情况下，必须定义一个无穷的、稠密的采样序列来得到分辨率完备性，这可以通过在极限情况下将离散度减少到 0 来实现。得到不规则采样点集的邻域的方法已经在基于采样的路线图环境中得到发展，参见第 5.6 节。提高分辨率的概念被推广到加入改善离散度（甚至差异）的采样点。

5.4.3 随机势场

如果问题可以用少量的点加以解决，那么对第 2.2 节离散算法所进行的适应性修改就成功了。每个轴上点的数量必须少或者维数必须低，以确保在每个轴有 k 个点、维数为 n 的情况下的点数 k^n 足够小以使能够在合理的时间内到达$\mathcal{G}$中每个顶点，例如，如果问题要求每个轴有 50 个点和维数是 10，那么搜索所有 50^{10} 点是不可能的。利用最好优先进行启发式的规划器可能不需要搜索大多数点就能够找到答案；但是对于诸如图 5.13(a)所示的简单问题，规划器也会花费太长时间来探索碗中的顶点[1]。

当最好优先搜索过程陷入困境时，随机势场方法[71,73,591]采用随机游走试图跳出局部最小点。这是最早的基于采样的规划器之一，它利用经典离散搜索以外的专门技术来试图更好地解决运动规划问题。在许多情况下已经取得了显著的效果。这种方法可以解决自由度多达 31 的问题，这在以前是不可能的。但是它的不足之处是涉及一些必须根据每个问题的不同而进行调整的启发式参数。这也促使人们研究更好的方法，这将在第 5.4.4 节、第 5.5 节和第 5.6 节进行介绍。尽管如此，在早期基于采样的方法中研究巧妙的启发式方法仍然是非常有价值的，因为它们阐述了许多有趣的问题，并且这种方法在其他基于采样的规划算法的研究中也很有影响[2]。

算法的最复杂的部分是定义势场，势场可以认为是一个伪度量，它尝试计算任何位形到目标的距离。在大多数势场表述中都有一个吸引项和一个排斥项。吸引项是$\mathcal{C}$上的一个度量，它产生与目标之间距离；排斥项对离障碍物太近的位形进行惩罚。构造势函数涉及许多启发式方法，详见文献[591]。一种最有效的方法涉及在$\mathcal{W}$上构造尚需代价函数并将它们提升到$\mathcal{C}$[72]。本节假设已经定义了一些势函数 $g(q)$，它们与第 2.2.2 节中的尚需代价函数相同。但是在这种情况下并不要求 $g(q)$是最优的，甚至也不要求 $g(q)$是真实尚需代价的低估值。

当搜索过程陷入困境并且需要随机游走时，它被迭代执行许多次。采用第 5.4.2 节的离散步骤，开始时定义一个高分辨率的栅格（例如每个轴有 50 个点）。每次迭代中，用如下方法对当前的位形进行修改。基于投掷硬币的结果，每个坐标 q_i 增加或者减少 Δq_i（栅格步长）。当然也必须遵守拓扑等同原则。每次迭代后，对新的位形进行碰撞检测，或者检查是否超出$\mathcal{C}$的边界（如果有边界）。如果发生碰撞或者超出边界，那么放弃该位形，并对以前的位形进行其他尝试。失败不断重复直到在$\mathcal{C}_{free}$中得到新的位形。

---

1 当然，那种问题不需要每轴上有太多的点；（如果算法能够适应性修改采样分辨率和离散度），那么该问题只需要用较少的点。

2 多年前，由该方法所得到的结果曾激发我致力于运动规划的研究。

所产生的规划器可以用图 5.15 所示的三状态机进行描述。每个状态称为一种模式，以避免与状态空间的概念相混淆。VSM 和 LPM 用模式来定义。初始时，规划器处在最好优先 BEST-FIRST 模式下，并使用 q_I 开始进行梯度下降。在 BEST-FIRST 模式下，VSM 选择最新的顶点 $v \in V$。在第一次迭代中，它是 q_I。LPM 在 v 的邻域内沿着最小化 g 的方向创造一个新顶点 v_n。方向采样可以用随机选取的或者确定性的采样点来完成。如果采用随机采样点，那么可以采用第 5.2.2 节的球形采样方法。进行一些尝试后，如果 LPM 不能减小 g，那么模式切换到随机游走 RANDOM WALK 模式，因为最好优先搜索陷入 g 的局部最小值。

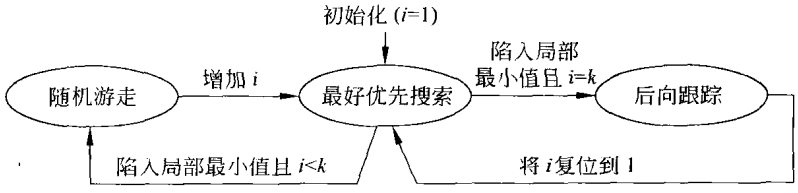


图 5.15　随机化势场方法可以建模为一个三状态机

在 RANDOM WALK 模式下，从最新的顶点执行随机游走。如果 g 降低或者到达迭代的指定极限值，那么随机游走结束。极限值从预先确定的随机变量采样而来（该随机变量包含一些必须选择的参数）。当 RANDOM WALK 模式结束时，模式又切换到 BEST-FIRST 模式。一个计数器跟踪尝试随机游走的次数。参数 K 指定试图进行随机游走的最大次数（一个合理的值是 K=20[72]）。如果 K 次随机游走后，BEST-FIRST 失败，那么进入 BACKTRACK 后向跟踪模式。该模式从随机游走过程中得到的顶点中随机选取一个顶点。然后计数器复位，模式又切换到 BEST-FIRST。

由于随机游走，所产生的路径通常因过于复杂而无法应用。幸运的是，所得到的路径很容易变换成一个更加简单且无碰撞的路径。通常的方法是沿着路径区域随机获取点对，试图用直线（一般情况下，C 中的最短路径）代替所得到的路径片段。例如假设随机选取 $t_1, t_2 \in [0,1]$，$\tau:[0,1] \to C_{free}$ 是计算得到的路径解。这条路径被变换成一条新的路径：

$$\tau'(t) = \begin{cases} \tau(t) & \text{如果 } 0 \leqslant t \leqslant t_1 \\ a\tau(t_1) + (1-a)\tau(t_2) & \text{如果 } t_1 \leqslant t \leqslant t_2 \\ \tau(t) & \text{如果 } t_2 \leqslant t \leqslant 1 \end{cases} \tag{5.39}$$

其中 $a \in [0,1]$表示从 t_1 到 t_2 路径的片段。即 $a = (t_2 - t)/(t_2 - t_1)$，新的路径必须进行碰撞检测。如果它能够通过检测，那么用它代替原来的路径；否则，放弃它，并选择一对新的 t_1 和 t_2。

随机势场方法能够跳出高维局部最小值，从而为一些具有挑战性的高维问题寻找到有趣的解。遗憾的是，大量的参数调整使多数人在目前宁可放弃这种方法，而去寻找新的方法。

5.4.4　其他方法

这里给出几种很有影响的基于采样的方法。它们具有随机势场方法所不具备的优势。它们都用到随机选择，这可能是受到势场方法的启发。

Ariadne 线索算法 该方法生长的搜索树在每次迭代中偏向于尽可能多地探索新的区域[690,691]。它有两种模式：搜索和探索，它们在相继的迭代中交替进行。在探索模式下，VSM随机选择顶点 v_e；LPM发现一个可以连接到 v_e 且离 $\mathcal{G}$ 中其他顶点尽可能远的新的位形。全局最优函数汇集了与其他顶点的距离的总和，它用遗传算法进行优化。在搜索模式下，试图将在探索模式下所增加的顶点向目标位形延伸。该方法影响下面将要介绍的方法以及第 5.5 节中的方法，其关键思路是必须花费一些时间来探索空间，而不是只关注于寻找问题的解。随机势场的贪婪行为有一定的效率，但是对某些问题它也会失败，因为它们都基于跳出关于目标的局部最小值，而不是将时间花费在全局探索上。Ariadne 线索算法的不足之处在于它很难解决探索模式下放置一个新的顶点的优化问题。文献[690]采用遗传算法，而实际上在运动规划中一般情况下不使用遗传算法，因为它需要针对特定问题进行参数调整。

可扩张空间规划器 该方法用试图探索空间的新的部分的方法生成采样点。在这种意义下，这与 Ariadne 线索算法的探索模式很相似，并且通过借鉴离散搜索算法中的双向搜索的思路，规划器会更加有效，就像第 2.23 节所介绍的那样。VSM 以一定概率从 $\mathcal{G}$ 选取一个顶点 v_e，概率与落在 v_e 预定邻域的其他顶点的数量成反比。因此，“孤立”顶点更可能被选中。LPM 在 v_e 的预定邻域内随机生成新的顶点 v_n。它以一定概率将 v_n 插入 $\mathcal{G}$，概率与落在 v_n 的预定邻域的其他顶点的数量成反比。对于固定次数的迭代，VSM 重复选择相同顶点直到移动到其他顶点。用非常简单的放置点的准则，所产生的规划器能够解决一些有趣的问题。主要的不足之处在于规划器要求大量的针对特定问题的参数调整（或者至少针对相似的问题族），并且如果问题要求系统地搜索一个长的迷宫，那么它的性能会下降。选择预定邻域的半径就等于确定相应的分辨率。

随机游走规划器 一个非常简单而有效的算法可以完全从随机游走产生[181]。为了避免参数调整，算法在每次迭代中调整它的方向和大小的分布，这种分布是基于过去 k 次成功迭代（k 可能是仅有的参数）。在每次迭代中，VSM 仅选择最近加入到 $\mathcal{G}$ 的顶点。LPM 通过从多变量高斯分布生成采样点来生成一个方向和大小，高斯分布的协方差参数是自适应调整的。其不足之处与以前的方法相似。它们都很难通过长的曲折的通道。可以将有适应能力的随机游走与诸如势场规划器等其他搜索算法组合起来[180]。

5.5 快速探索稠密树

本节介绍一种递增采样和搜索方法，该方法在实际应用中具有良好的性能，且不需要任何参数调整[1]。思路是递增地构造一个搜索树，搜索树逐步提高分辨率，但不需要明确设置分辨率参数。在极限情况下，树稠密地覆盖整个空间。因此它的特性与充满曲线的空间相似[845]，树是由许多短的路径而不是一条长的路径所组成。在递增地构造树的过程中，一个稠密的采样序列用来作为引导。如果该序列是随机的，那么所产生的树称为快速探索随机树（rapidly exploring random tree，RRT）。一般情况下，不管序列是随机的还是确定性

1 原始的 RRT[601]引入一个步长参数，但是目前去掉该参数。为了实现该算法，某些人可能仍然希望用这种旧的方法来表述算法，因为实现起来更容易。这将在稍后进行讨论。

的，这种树称为快速探索稠密树（rapidly exploring dense trees，RDT），用它来表明得到了空间的一个稠密覆盖。这种方法最初是为微分约束情况下的运动规划而产生的[611,614]；这将在第 14.4.3 节介绍。

5.5.1　探索算法

在说明如何使用这些树来求解规划问题之前，假设从初始位形开始，目标是到达距离每个位形尽可能近的地方。方法针对任何稠密序列。再次令α表示C中的一个无穷、稠密的采样序列。第 i 个采样点用$\alpha(i)$表示。这可能包括了均匀随机序列，该序列仅以概率 1 稠密。产生一个非均匀偏差的随机序列也是可以接受的，只要它们以概率 1 稠密。

一个 RDT 是一个拓扑图$\mathcal{G}(V, E)$。令 $S \subset C_{free}$ 表示$\mathcal{G}$所能到达的所有点的集合。因为每个 $e \in E$ 是一条路径，这可以表示为图的刈幅（swath）S，定义为：

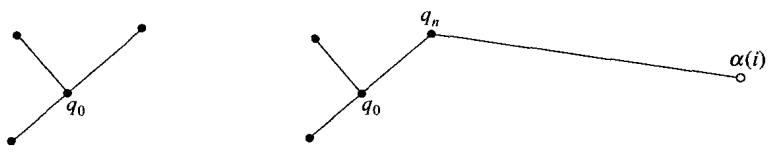
$$S = \bigcup_{e \in E} e([0,1]) \tag{5.40}$$

在式（5.40）中，$e([0,1]) \subseteq C_{free}$ 是路径 e 的像。

探索算法首先由图 5.16 来解释，这里没有任何障碍物或者边界阻挡物。假设C是度量空间。初始情况下，在 q_0 产生一个顶点。对 k 次迭代，将$\alpha(i)$连接到刈幅 S 中离它最近的点，可以迭代地生长一个树。连接通常沿着可能的最短路径进行。在每次迭代中，$\alpha(i)$变成一个顶点。因此产生的树是稠密的。图 5.17 和图 5.18 用图形的方式描述了一个迭代过程。假设树有 3 条边、4 个顶点，如图 5.17(a)所示。如果距离$\alpha(i)$最近的点 $q_n \in S$ 是一个顶点，如图 5.17(b)所示，那么从 q_n 到$\alpha(i)$形成一条边。但是如果最近的点落在一条边的内部，如图 5.18 所示，那么这条边被分割以使 q_n 作为新的顶点而出现，并且产生一条从 q_n 到$\alpha(i)$的边。如果需要，假设边的分割在第 4 行用增加边的方法进行处理。这样在每次迭代中边的总数会增加一条或者两条。

```
SIMPLE_RDT(q0)
1   G.init(q0);
2   for i=1 to k do
3       G.add_vertex(α(i));
4       qn ← NEAREST(S(G),α(i));
5       G.add_edge(qn, α(i));
```

图 5.16　没有障碍物时构造 RDT（它将 RRT 作为一种特殊情况）的基本算法；它要求一个稠密序列α并迭代地将α(i)连接到在$\mathcal{G}$到达的所有点中最近的点



(a)　　(b)

图 5.17　(a) 假设到目前为止已经用图 5.16 中的算法构造了这个树；(b) 加入一条新的边，它将采样点$\alpha(i)$连接到 S 中的最近点，即顶点 q_n

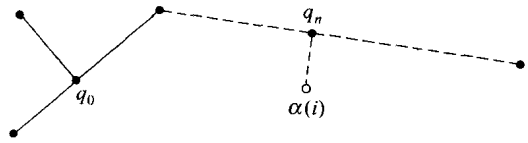
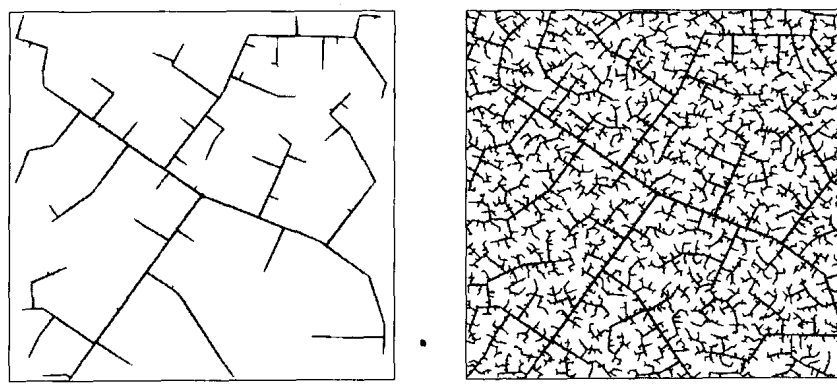


图 5.18 如果 S 中的最近点在一条边上，那么边被一分为二，并且新的顶点被插入到$\mathcal{G}$

这里描述的方法与第 5.4.1 节的一般架构并不完全相符；但是在经过用第 5.5.2 节提出的建议修改后就相符了。在 RDT 表述中，NEAREST 函数实现的是 VSM 的目的，但是点可以从图的刈幅中的任何地方进行选择。VSM 可以推广为刈幅-点选择方法（SSM）。这种推广将在第 14.3.4 节得以应用。LPM 试图沿着$\mathcal{C}$中可能的最短路径将$\alpha(i)$到 q_n。

图 5.19 显示了图 5.16 所示算法的一个执行过程，其中$\mathcal{C}=[0,1]^2$ 和 $q_0=(1/2, 1/2)$。它展示了一种分形行为[1]。它首先构造几个主要的分支快速到达空间的远的角落。逐渐地，更多的面积被更小的分支所填充。从该图可以清楚地看到，在极限状况下，树稠密地充满整个空间。因此当迭代过程继续进行时，树逐步提高分辨率（或者离散度）。这种行为对基于采样的运动规划来说是非常理想的。



45 次迭代　　2345 次迭代

图 5.19 在迭代的初期，RRT 快速地到达没有探索到的地方。尽管如此，RRT 在极限情况下是稠密的（以概率 1），这意味着它能够到达空间中的任意点

回忆到在基于采样的运动规划中，障碍区域 $\mathcal{C}_{obs}$ 并没有明确表示。因此在树的构造过程中必须加以考虑。图 5.20 说明如何修改图 5.16 的算法以使碰撞检测全被考虑到。修改的算法将出现在图 5.21 中。步骤 STOPPING-CONFIGURATION 沿着$\alpha(i)$的方向产生距离 $\mathcal{C}_{free}$

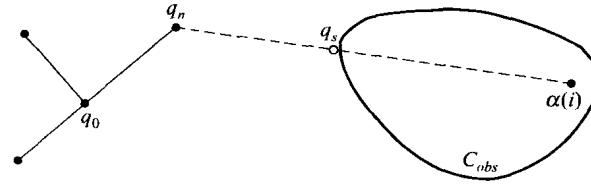


图 5.20 如果有一个障碍物，那么边发展到障碍物的边界，与边界之间的距离为碰撞检测算法所允许的值

1 如果 α 是均匀的、随机的，那么得到随机分形[589]。确定的分形可以用有适当对称性的序列来构造。

```
RDT(q0)
1   G.init(q0);
2   for i=1 to k do
3       qn ←NEAREST(S,α(i));
4       qs ←STOPPING-CONFIGURATION(qn, α(i));
5       if qs ≠ qn then
6           G.add_vertex(qs);
7           G.add_edge(qn,qs);
```

图 5.21　有障碍物的 RDT

的边界最近的位形。最近点 $q_n \in S$ 被定义成相同的（忽略障碍物）；但是新的边可能没有到达 $\alpha(i)$。在这种情况下，从 q_n 到在碰到障碍物之前的最后的点 q_n 之间构造一条边。边能够到达离障碍物的边界多近的地方？这依赖于所使用的碰撞检测方法，如第 5.3.4 节所描述。有时可能 q_n 在 $\alpha(i)$ 的方向上距离 C_{free} 的边界可能已经足够近了。在这种情况下，那次迭代不增加新的边和顶点。

5.5.2　有效地发现最近点

有几种有趣的方法可实现图 5.16 算法的第三行 NEAREST 函数。一般有两个方法族：精确的或者近似的。首先考虑精确情况。

精确解　假设对于某维数 $m \geq n$，$\mathcal{G}$中的所有边都是 $\mathbb{R}^m$ 中的直线段。在构造过程中生成的一条边在以后的迭代中将被多次分割。对于寻找 S 中的最近点来说，最好将它们作为单个线段来处理。例如图 5.19 中从根部扩展的三个大的分支。当点的数量增加时，使线段凝聚成团的好处就逐渐显现。这些凝聚后的线段称为超线段。为了实现 NEAREST，需要一个本原来计算一个点与一条线段之间距离。这可以用简单的向量积分来完成。使用该本原，NEAREST 是这样实现的：在所有超线段上进行迭代，选择距离最短的点。建立可以消除大的超线段集合的层次数据结构有可能提高算法的性能，但是这依然需要用实验的方法才能得到。

在某些情况下，$\mathcal{G}$的边可能不是线段。例如 $SO(3)$中两个点之间的最短路径实际上是沿着 $\mathbb{S}^3$ 的圆弧。一种可能的解决方法是用 C 的一个单独参数。例如通过进行相应的等同来得到 $\mathbb{RP}^3$，那么 $SO(3)$可以表示为 $[0,1]^3/\sim$。这样就可以使用线段了。问题是所产生的度量与 Haar 测度不一致，这意味着将产生一个偶然的差异。另外一种方法是将 $\mathbb{S}^3$ 紧密地封在一个四维方体中。$\mathbb{S}^3$ 上的每个点都可以向外映射到方体的面上。由于对径点等同，方体的八个面中只需要用其中四个来得到所有旋转集合和方体面之间的一个双射。沿着方体的面可以进行线性插值，只要这两个点在同一个面上。如果点在不同面上，那么可以用将最短路径沿两个面之间的角进行弯曲所得到的两条线段。这种安排比将 $SO(3)$映射到 $[0,1]^3/\sim$所产生的变形小；但是仍然存在变形。

另外一种方法是用可以计算点与曲线之间距离的本原，这可以完全避免变形。在 $SO(3)$情况下，需要一个能够发现 $\mathbb{R}^m$ 中的圆弧和 $\mathbb{R}^m$ 中的点之间距离的本原。这可能不太困难，但是如果曲线太复杂，那么 NEAREST 函数的进行可能花费太多计算时间。

近似解　与精确解相比较而言，近似解更容易构造，但是需要引入分辨率参数。每个

路径片段都可以通过沿着长线段插入中间顶点来近似，如图 5.22 所示。每当一个新的采样点 $\alpha(i)$插入$\mathcal{G}$时，都应该增加中间顶点。定义一个参数 Δq，插入的中间采样点应该确保$\mathcal{G}$中任何两个相邻顶点不超过 Δq。使用中间顶点，当寻找 S 中的最近点时忽略$\mathcal{G}$中边的内部。NEAREST 函数的近似计算是通过发现$\mathcal{G}$中距离 $\alpha(i)$最近的顶点来实现的。到目前为止，该方法是最简单的。它与第 5.4.1 节的递增采样和搜索是完全相符的。

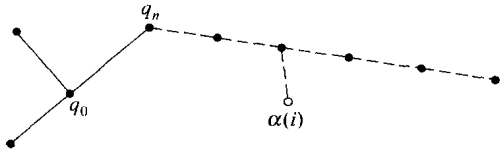


图 5.22 从实现的角度，插入中间顶点以避免对沿着线段最近的点进行检验；所付出的代价是顶点的数量大幅增加

当使用中间顶点时，很显然进行了折中。NEAREST 的计算时间与顶点数呈线性关系。增加顶点的数量可以提高近似的质量，但是也会增加计算时间。补偿这种损失的一种方法是将顶点插入一个为搜索最近邻近点而建立的数据结构。目前应用最广泛的一种数据结构是 Kd-树[266,367,761]。图 5.23 显示了 14 个点的 Kd-树。Kd-树可以认为是二叉树的多维推广。P 个点的 Kd-树构造过程如下。初始情况下，将点关于 x 坐标进行排序。那么中值点 $p \in P$ 将 P 分为两个集合，这取决于点处在通过点 p 的垂线的哪一侧。对其中的每一侧来说，再根据 y 坐标进行分类，选取中值点。通过该中值点的水平线将点再次分割为上下两个部分。在下一轮的递归中，再次使用垂线、水平线，如此重复。通过循环进行 n 维坐标分割，相同的思路可以应用于 $\mathbb{R}^n$。文献[53]将 Kd 树扩展到运动规划的拓扑空间，在 RRT 和基于采样的路线图中产生了良好的性能。k 点 Kd-树可以在 $O(nk \lg k)$时间内构造完毕。对树进行遍历时必须仔细考虑拓扑等同。为了发现树中距离给定点最近的点，算法访问一个叶子顶点，与该顶点有关的区域包含疑问点，算法发现从这片叶子中的数据点到疑问点的所有距离，选取最近的点。接下来，递归地访问那些周围的、比所发现的最近点稍微远一点的叶子顶点[48,53]。最近点可以在与 k 呈对数关系的时间内被发现。

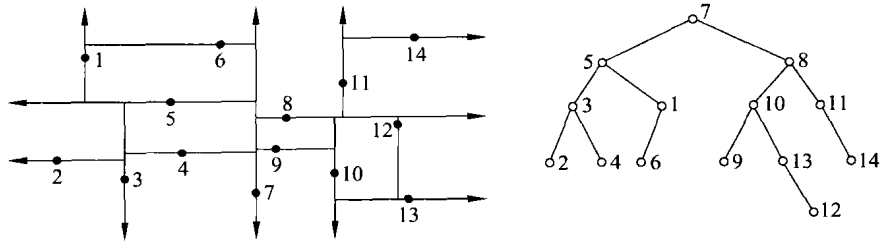


图 5.23 在有效的最邻近计算中用到 Kd-树

遗憾的是，这些限制隐藏了一个随维数 n 呈指数增长的常数。实际上，Kd-树在维数大约为 20 的问题的运动规划中很有用。如果维数再增加，性能通常会很快降低。一个经验原则是，如果有多于 2^n 个点，那么 Kd-树比原始的最邻近方法更有效。一般情况下，在特定的应用中必须仔细考虑折中来确定精确解、原始的最邻近方法计算的近似解或者 Kd-树法得到的近似解究竟哪个更有效。由于存在实现起来复杂性的问题，这可能使人们宁可采用原始的最邻近方法进行近似。

5.5.3　在规划中使用树

到目前为止的讨论都集中在探索C_{free}，但是这本身并不能解决规划问题。RRT 和 RDT 都可以以多种方式应用到规划算法中。例如将它们用在第 5.4.3 节的随机势场规划器中来跳出局部最小点。

单树的搜索　直接使用图 5.21 的算法从q_I开始生长树，并且周期性地检查是否可能将 RDT 连接到q_G，这样就可以产生一个有效的规划器。完成该思路的一个简单的方法是从一个稠密序列 α 开始，周期性地以等间隔插入q_G。例如每第 100 个采样点可以是q_G。每到这个采样点，尝试从 RDT 中最近的顶点到达q_G。如果采样序列是随机的，这样生成的是 RRT，那么可以对它进行如下修改。在每次迭代中，抛掷一个有偏性的硬币，它出现正面的概率是 99/100，出现反面的概率是 1/100。如果抛掷的结果是正面，那么将$\alpha(i)$设置为伪随机序列的下一个元素；否则，$\alpha(i)=q_G$。这使 RRT 偶尔尝试连接目标q_G。当然 1/100 是随意而定的，但是它在正常的范围内。如果偏得太多，那么 RRT 会变得像随机势场一样贪婪。如果偏得太少，那么就无法将树连接到q_G。也可考虑C中其他稠密的，但不一定是非均匀分布的序列。例如在随机采样情况下，概率密度函数可以含一个偏向目标的偏值。选择偏值是一个非常困难的启发式问题；因此使用该技术必须谨慎（或者说应该避免使用）。

平衡双向搜索　通过生长两个 RDT，一个从q_I开始，另一个从q_G开始，这样可以得到更好的性能。对于跳出陷阱来说，这特别有价值，如第 5.4.1 节所示。对于栅格搜索，很容易实现能够保证两个树相遇的双向搜索。对于 RDT，则需要特别的考虑来保证两个树相遇，同时保持它们的“快速探索”的性质。还需要确保双向搜索是平衡的[563]，这样才能保证树的大小相同。

图 5.24 给出了算法的概略描述。图$\mathcal{G}$被分解成两个树，记为T_a和T_b。初始情况下，它们分别从q_I和q_G开始。经过多次迭代以后，T_a和T_b被交换；因此记住T_a并不总是包含q_I的树。在每次迭代中，T_a的生长方式与图 5.16 中算法一次迭代的方式相同。如果新的顶点q_s加入T_a，那么在第 10 行～第 12 行试图扩展T_b。扩展T_b用的是T_a的新顶点q_s，而不是$\alpha(i)$。这使得T_b向T_a生长。如果它们连接起来，那么就得到问题的解，由第 13 行检验它们是否连接上。

第 14 行是平衡搜索过程的一个重要步骤。这对于诸如图 5.13(b)所示的陷阱、图 1.2 所示的难题等问题来说是非常重要的。如果其中有一个树在探索中遇到麻烦，那么给它更多的能量是有意义的。因此对于较小的树来说，新的探索总能完成。那么“较小”是如何定义的呢？一个简单的指标是所用顶点的数量。另一个合理的指标是用树中所有线段的长度。

通过在每次迭代中将两个树进行交换，可以产生一个不平衡的双向搜索。一旦交换了树，那么角色也颠倒过来。例如第一次交换后，T_b扩展的方式与图 5.16 相同，并且如果加入新的顶点q_s，那么尝试将T_a连接到q_s。

如果α是确定性的，那么还存在一个重要的问题。即使α是稠密的，当采样点分配到每个树中时，每个树接收到的可能不是一个稠密集合。如果每个树使用它自己的确定性序列，那么可以避免这个问题。在产生双向 RRT 规划器的情况下，每个树可以使用相同的（伪）随机序列而不会遇到麻烦。

```
RDT_BALANCED_BIDIRECTIONAL(q_I, q_G)
1   T_a.init(q_I); T_b.init (q_G);
2   for i=1 to K do
3       q_n ←NEAREST(S_a, α(i));
4       q_s ←STOPPING-CONFIGURATION(q_n, α(i));
5       if q_s ≠ q_n then
6           T_a.add_vertex(q_s);
7           T_b.add_vertex(q_n, q_s);
8           q'_n ← NEAREST(s_b, q_s);
9           q'_s ← STOPPING-CONFIGURATION(q'_n, q_s);
10          if q'_s ≠ q'_n then
11              T_b.add_vertex(q'_s);
12              T_b.add_edge(q'_n, q'_s);
13          if q'_s =q_s then return SOLUTION;
14      if |T_b|>|T_a| then SWAP(T_a, T_b);
15  return FAILURE
```

图 5.24 基于 RDT 的一个双向规划器

多树搜索 如果双树方法具有单树方法所没有的优越性，那么自然就会提出这样的问题：生成 3 个或者 3 个以上的 RDT 是否会更好。这对于像图 5.13(c)所示的有两个陷阱的问题来说非常有帮助。新的树可以从$\mathcal{C}$的难以到达的部分进行生长。控制树的数量以及确定什么时候来进行连接是非常困难的。沿着这个研究方向近来有一些成果[84,919,920]。

这些额外的树可以从任意（可能是随机的）位形开始。当考虑更多的树时，就会产生一个复杂的决策问题。计算时间必须在探索空间和连接树之间进行分配。还不清楚应该进行哪种连接。在这种规划器以及其他基于 RRT 的规划器的发展中还有许多问题值得研究。极限情况是从每个采样点$\alpha(i)$开始一个新的树，并在可能的时候连接附近的树。这种方法产生了一个不依赖于疑问而覆盖整个空间的图。这是下一节的主要内容。

5.6 多疑问问题的路线图方法

以前假设对规划算法给定单个初始-目标对。现在假设对算法给定多个初始-目标对，同时保持机器人模型和障碍物不变。这就是多疑问运动规划问题。在这种情况下投入大量时间对模型进行预处理以有效地解答将来的疑问，这样做是有意义的。目标是构造一个称为路线图的拓扑图，它能够有效地解决多初始-目标疑问问题。从直观上看，路线图上的路径应该容易从q_I到达q_G，并且能够快速对图进行搜索。这里给出的一般架构主要在文献[519]中介绍，那里称之为随机路线图（probabilistic roadmap，PRM）。但是随机性对该方法并不重要。因此我们称该方法族为基于采样的路线图法。这用来区别在第 6 章将要介绍的组合路线图。

5.6.1 基本方法

令$\mathcal{G}(V, E)$表示一个拓扑图，其中 V 是顶点的集合，E 是$\mathcal{C}_{free}$中的路径的集合。多疑问

问题的基本原理是将运动规划分为两个计算阶段。

预处理阶段：在预处理阶段，大量的工作在于以有利于快速解答将来疑问的方式建立$\mathcal{G}$。因此它称为路线图，在某种意义上，从$\mathcal{C}_{free}$的任何位置都应该能够访问路线图。

疑问阶段：在疑问阶段，给定一个q_I、q_G对。每个位形必须很容易用局部规划器连接到$\mathcal{G}$。接着用第 2.2 节中的任何算法进行离散搜索来得到边的序列，这些边构成了从q_I到q_G的路径。

基本预处理阶段　图 5.25 概略描述了基本预处理阶段过程，图 5.26 对算法进行了阐述。可以看到，算法利用了一个均匀、稠密序列α。在每次迭代中，算法必须检查是否$q \in C_{free}$。如果$q \in C_{obs}$，那么必须继续迭代直到得到无碰撞采样点。一旦$\alpha(i) \in C_{free}$，那么在第 4 行将该采样点插入作为$\mathcal{G}$的顶点。下一步是尝试将$\alpha(i)$连接到$\mathcal{G}$中的某些邻近顶点 q。连接是用 CONNECT 函数进行的，该函数是第 5.4.1 节中介绍的一种典型的局部规划方法 LPM（local planning method）。在多数实现过程中，这只是简单地检验了$\alpha(i)$与 q 之间的最短路径。根据经验，使用第 5.3.4 节结尾所描述的多分辨率 Van der Corput 方法似乎更加有效。如果不采用最短路径，那么也可以采用更加复杂的连接方式，如图 5.24 中的双向算法。如果路径是无碰撞的，那么 CONNECT 函数返回 TRUE。

```
BUILD_ROADMAP
1   G.init(); i←0;
2   while i<N
3       if α(i)∈C_free then
4           G.add_vertex(α(i)); i←i+1;
5           for each q∈NEIGHBORHOOD(α(i),G)
6               if ((not G.same_component (α(i),q)) and CONNECT (α(i),q)) then
7                   G.add_edge(α(i),q));
```

图 5.25　基于采样的路线图的基本构造算法；如果$\alpha(i)$发生碰撞，那么 i 不增加；这使 i 能够准确地记录路线图中顶点的数量

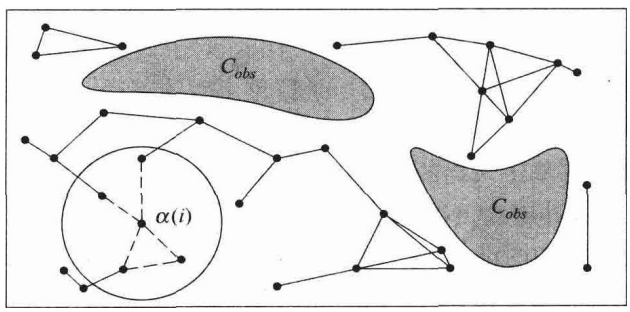


图 5.26　通过将每个新的采样点$\alpha(i)$连接到路线图中附近顶点上，这样就可以递增地构造基于采样的路线图

在进行碰撞检测之前，第 6 行的 same_component 条件检查确认$\alpha(i)$和 q 是$\mathcal{G}$中不同的分支。这就保证了每次进行连接的时候，$\mathcal{G}$中被连接的分支的数量是减少的。这可以通过使用以前提到的 union-find 算法[245,826]来实现。在一些实现中可以忽略这一步，特别是如果需要生成多个可选的解时。例如从不同同伦类中生成路径解。在这种情况下，对于某固

定的 K，条件 not $\mathcal{G}$.same_component($\alpha(i)$, q)被$\mathcal{G}$.vertex_degree(q)<K 所代替（例如 K=15）。

选择邻近采样点 第 5 行有几种可能的实现。在这些实现中，最好对顶点进行排序，这些顶点以增加与$\alpha(i)$的距离的顺序进行连接。这是有意义的，因为对较短路径进行的碰撞检测所花费的开销通常较小，并且它们无碰撞的可能性更大。如果进行了连接，这就避免了对开销大的、通向属于相同连通分支的位形的、更长路径进行的碰撞检测。

几种有用的 NEIGHBORHOOD 的实现是：

（1）**最近的 K**：考虑离$\alpha(i)$最近的 K 个点。这要求设置参数 K（典型值为 15）。如果不能确定使用哪个实现方法，那么可以试一试该方法。

（2）**分支 K**：从$\mathcal{G}$中每个连通分支尝试得到 K 个最近的采样点。一个合理的 K 值是 K = 1；否则，就需要尝试太多的连接。

（3）**半径**：选取以$\alpha(i)$为中心、以 r 为半径的球中所有的点。设定 K 的上限以防尝试太多的连接。典型的是 K=20。点数的增加可以相应缩小球的半径。这种缩小可以基于离散度或者差异，如果它们对α来说都是可用的。如果采样点很有规律（例如栅格），那么选择最近 K 点与选取球中的点本质上是等价的。如果点集很不规律，例如随机采样情况，那么选择最近 K 点似乎更可取。

（4）**可视性**：在第 5.6.2 节将会描述一个变化形式来确定将α尝试连接到$\mathcal{G}$中所有顶点是否值得。

上述方法都要求$\mathcal{C}$是一个度量空间，并且确保相对$\alpha(i)$的 NEIGHBORHOOD 点的方向是均匀分布的，这一点以前并未予以太多注意。例如，如果 20 个最近的点都簇拥在一个方向上，那么试图连接到另外一个点上是更可取的，因为该点在相反方向上。

疑问阶段 在疑问阶段，假设$\mathcal{G}$对解答一些疑问来说是足够完整的，每个疑问都给出一个初始位形 q_I 和目标位形 q_G。首先，疑问阶段假装 q_I 和 q_G 好像是为连接到$\mathcal{G}$而从α中选择的。这要求多运行两次图 5.25 中算法的迭代。如果 q_I 和 q_G 成功连接到$\mathcal{G}$中的其他顶点，那么就完成了对连接顶点 q_I 和顶点 q_G 的路径的搜索。图中的路径直接对应于$\mathcal{C}_{free}$ 中的路径，这就是疑问的解。遗憾的是，如果该方法失败，那么就不能确定解是否存在。如果知道采样序列α的离散度，那么至少可能得到这样的结论：对于规划器的该分辨率来说，解不存在。也就是说，如果解存在，那么它要求一条穿过走廊的路径，这个走廊的宽度不会比最大空球的半径更宽[603]。

一些分析 有许多对基于采样的路线图性能的分析[70]。我们简要介绍其中一种。考虑如图 5.27 所描述问题，其中 CONNECT 方法在细管道中很可能失败，即使存在一种连接。更高维数的这些问题则更加困难。一些规划问题涉及可移动机器人通过一个空隙很小的区域。在$\mathcal{C}_{free}$ 中这通常形成的是一条狭窄的通道，这导致了对于基于采样的路线图算法来说非常具有挑战性的规划问题。从陷阱中逃离出来也非常难，但是对于路线图方法来说，即使从单个走廊穿过也非常不容易（除非采用更加复杂的 LPM[482]）。

令 $V(q)$表示用 CONNECT 方法能够连接到 q 的所有位形的集合。直觉地，这可以认为是视线能够“看见”的所有位形的集合，如图 5.28(a)所示。

$\mathcal{C}_{free}$ 的 ϵ-优度定义为：

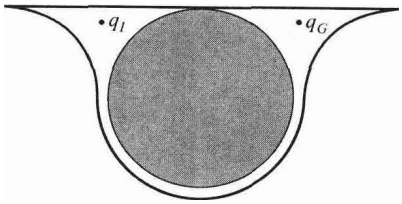
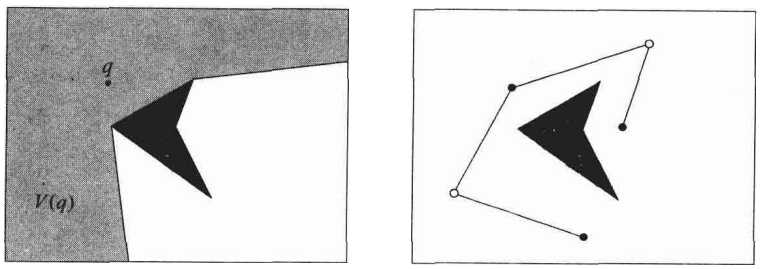


图 5.27　对于基于采样的路线图（在高维 C-空间中）来说是非常困难的一个例子，因为某些采样点必须沿着弯曲管道中的一些点；但是其他方法可能很容易解决它



(a) 可视性的定义　　(b) 可视路线图

图 5.28　(a) $V(q)$是用 LPM 方法从 q 所能到达的点的集合；(b) 一个可视路线图有两种类型的顶点：guard，用黑色显示；connector，用白色显示；不允许 guard 看见其他 guard；connector 必须看见至少两个 guard

$$\epsilon\left(C_{free}\right)=\min_{q\in C_{free}}\left\{\frac{\mu\left(V\left(q\right)\right)}{\mu\left(C_{free}\right)}\right\} \tag{5.41}$$

其中 μ 表示测度。直觉地，ϵ (C_{free})表示从任何点都可以看见的C_{free} 中的一片区域。根据 ϵ 和 $\mathcal{G}$中顶点的数量，可以建立发现解的概率的界[70]。主要的困难在于 ϵ -优度的概念非常保守（它在所有位形上使用最坏情况），并且它是根据C_{free} 的结构来进行定义的，所以不能有效地进行计算。这个结果和其他相关结果有助于加深对基于采样的规划的理解，但是这个界很难应用于确定算法的性能是否良好这种特殊问题。

5.6.2　可视路线图

最有用的一种基于采样的路线图方法是可视路线图法[886]。该方法力求确保路线图很好地覆盖C_{free}。它的运行时间通常比图 5.25 的基本算法长，但是对于多疑问问题，额外的时间消耗通常是值得的。

思路是在$\mathcal{G}$中定义两个不同类型的顶点。

- Guard：为了成为一个 guard，一个顶点 q 一定不能看见其他 guard。因此可视区域 $V(q)$必须没有其他 guard。
- Connector：为了成为一个 Connector，一个顶点 q 必须看见至少两个 guard。因此存在两个 guard q_1 和 q_2 使得 $q\in V(q_1)\cap V(q_2)$。

路线图构造阶段的进行与图 5.25 的算法相似。NEIGHBORHOOD 函数返回$\mathcal{G}$中的所有顶点。因此对于每个新的采样点$\alpha(i)$，都试图将该采样点连接到$\mathcal{G}$中其他每个顶点。

可视路线图的新颖之处在于它使用一个强的指标来确定是否保留$\alpha(i)$及其$\mathcal{G}$中相关的

边。对于每个$\alpha(i)$，有三种可能的情况：

（1）新的$\alpha(i)$不能连接到任何 guard。在这种情况下，$\alpha(i)$成为 guard，并被插入到$\mathcal{G}$中。

（2）新的$\alpha(i)$能够连接到来自于至少两个不同连通分支的 guard。在这种情况下，该采样点变成一个 Connector，它和它的相关边一同插入到$\mathcal{G}$中，这些相关的边将它连接到来自于不同分支的 guard。

（3）前面两个条件都不满足。这意味着该采样点仅能够连接到来自于同一个连通分支的 guard。在这种情况下，应放弃该采样点$\alpha(i)$。

最后一个条件能够大量减少路线图顶点的数量。

这种方法的一个问题是：为了往后可能出现的更好的 guard，它不允许删除当前的 guard。guard 的位置取决于α中采样点出现的顺序。如果 guard 在序列中不能很好定位，该方法的效果可能不好。最好能够进行自适应策略，在后来的迭代中当有更好的位置时对 guard 进行重新分配。如何有效地实现这种策略还是一个问题。注意到如果采用随机采样，算法在概率意义下依然是完备的；如果α是稠密的，那么即使一些采样点被拒绝，算法仍然是分辨率完备的。

5.6.3 改进路线图的启发式方法

在寻求通过采样来设计一个好的路线图的过程中，已经产生了许多进行采样和连接路线图的启发式方法。它们大多数都利用了 C-空间的形状和/或机器人和障碍物的几何与运动学等特征。重点是寻找一个大量减少所需采样点数的方法。这里简要描述几种。

顶点加强[519] 启发式策略着力于在图 5.25 中的路线图构造算法中那些难于连接到其他顶点的顶点。给顶点 $v\in V$ 定义一个概率分布 $P(v)$。然后按照 $P(v)$在 V 中对顶点进行多次采样，然后从 v 进行一些随机运动以尝试到达新的位形。这些新的位形作为顶点而加入，并将它们连接到其他顶点，就像图 5.25 算法的正常迭代中 NEIGHBORHOOD 函数所进行的选择。一种值得推荐的定义 $P(v)$的启发式方法是为每个 v 定义一个统计量 $n_f/(n_t+1)$[519]，其中 n_t 是为 v 所进行的连接的总数，n_f 则是连接失败的次数。概率 $P(v)$是 $n_f/(n_t+1)m$，其中 m 是在所有 $v\in V$ 上统计量的总和（这对统计量进行归一化来得到有效的概率分布）。

在C_{free}的边界上进行采样[22,26] 这个策略是基于这种直觉：沿着边界∂C_{free}进行采样有时可能更好，而不是将采样点浪费在很可能没有障碍物的C_{free}的大范围内。图 5.29(a)显示了一种可以实现的方法。对每一个落入C_{obs}的采样点$\alpha(i)$，在C中随机选择许多方向；这些方向能够用第 5.2.2 节的$\mathbb{S}^n$采样方法进行采样。对每个方向，进行二叉树搜索来得到C_{free}中距离C_{obs}尽可能近的采样点。对二叉树搜索中点的计算顺序显示在图 5.29(a)中。令$\tau:[0,1]$表示从$\tau(0)\in C_{obs}$到$\tau(1)\in C_{free}$的路径。第一步检验中间点$\tau(1/2)$。如果$\tau(1/2)\in C_{free}$，这意味着∂C_{free}处在$\tau(0)$和$\tau(1/2)$之间；否则它处于$\tau(1/2)$和$\tau(1)$之间。接下来的迭代过程选择包含∂C_{free}的路径片段的中间点。这将是$\tau(1/4)$或者$\tau(3/4)$。该过程递归继续直到得到满意的解为止。

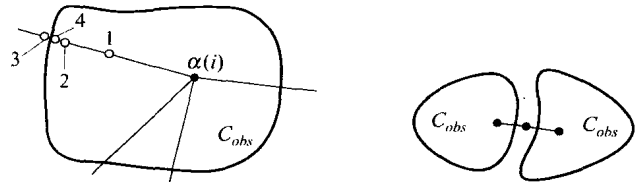


图 5.29　(a) 为了得到沿着边界的采样点，从C_{obs}中的一个采样点沿着随机方向进行二叉树搜索；(b) 通过检查沿着一条直线的 3 个邻近采样点，bridge-test 寻找狭窄走廊

高斯采样[134]　高斯采样策略的动机与在边界上进行采样是相同的。在这种情况下，目的是通过使用高斯分布得到∂C_{free}附近的点，高斯分布的采样点距离∂C_{free}更近且偏差更小，这种差异由高斯分布的方差来确定。采样点由如下方法生成。随机均匀地生成一个采样点 $q_1 \in C$。接着按照均值为 q_1 的高斯分布生成另外一个采样点 $q_2 \in C$；分布必须适应性修改到任何拓扑等同和/或C的边界。如果 q_1 或者 q_2 其中一个落在C_{free}，而另一个落在C_{obs}，那么落在C_{free} 的那个采样点继续作为路线图的顶点而保留下来。对某些例子，这样可以大量减少所需顶点的数量。

bridge-test **采样**[468]　部分地基于对一个点对的检验，高斯采样策略决定是否保留一个采样点。该思路往前进行一步就得到 bridge-test 采样，它使用沿着一个线段的 3 个采样点。如果采样点如图 5.29(b)那样排列，那么中间点就变成一个路线图顶点。这是基于狭窄的走廊至少在一个方向上是很细的这种直觉。bridge-test 能够表明一个点处在细走廊中，该点通常是放置顶点的重要位置。

中间轴采样[458,638,971]　如果不对接近边界的点进行采样，那么另外一种策略是迫使采样点距离边界尽可能远。令(X, ρ)是度量空间。令一个最大球 $B(x,r) \subseteq X$，这使得其他球都不是适当的子集。所有最大球的中心描绘出一个被称为中间轴的一维点集。$\mathbb{R}^2$ 的矩形子集的一个中间轴如图 5.30 所示。C_{free} 中的中间轴是基于可以内切 clC_{free} 的最大球。在中间轴上进行采样通常很困难，主要是因为C_{free} 的表示是隐含的。碰撞检测中的距离信息可以用于开始一个采样点$\alpha(i)$，迭代地对它进行扰动以增加与∂C_{free}之间的距离[638,971]。文献[458]提出在 $\mathcal{W}\backslash\mathcal{O}$ 的中间轴上进行采样。在这种情况下，$\mathcal{W}\backslash\mathcal{O}$ 中的中间轴更容易进行计算，并且可以用来启发性地引导放置C_{free}中好的路线图顶点。

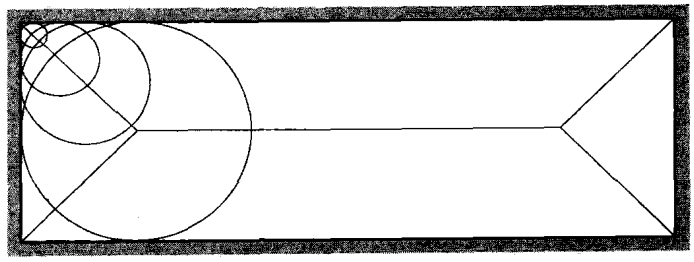


图 5.30　最大内切球的中心描绘出中间轴；矩形内的 5 条线段对应于中轴

进一步阅读

与前两章不同，本章是最新研究成果的集中体现。基于采样的运动规划的许多方面还在进一步发展中。早期的方法包括文献[71, 146, 195, 282, 284, 331, 332, 661, 763]。Gilbert-Johnson- Keerthi 算法[390]是早期的碰撞检测方法，它促进了基于采样的运动规划的发展；一些文献参考见文献[475]和文献[591]。在许多以前的工作中，广为采用的主要是随机化的方法；但是近年来人们已经认识到确定性采样在得到分辨率完备性的同时至少也有同样的效果。关于基于采样的运动规划的最新成果，参见文献[643]。

第 5.1 节是基于基本数学书中的内容。对基本定理和度量空间的大量实例的综述，参见文献[699]。更多内容出现了基本的点集拓扑书籍（例如文献[454,499]）和分析的书籍中（例如文献[348]）。文献[21,612]中讨论了基于采样的运动规划中的关于度量的内容。测度理论通常在关于实分析的内容中进行介绍[348,428,549,839,840]。更多关于 Haar 测度的内容，参见文献[428]。

第 5.2 节是基于数值积分和优化中的蒙特卡罗和准蒙特卡罗方法。一个非常有代表性的优秀文献是[740]。其他重要的文献包括[193,543,685,937,938]。文献[382,562,603,642,987]考虑了运动规划中的采样问题。文献[343,505]对蒙特卡罗算法进行了综合介绍。最早的关于蒙特卡罗方法的文献是[698]。关于计算 Voronoi 图的算法的综述，参见文献[55]。

更多关于碰撞检测的内容，参见文献[491,640,641,706]中的综述。文献[408,641,705]介绍了层次碰撞检测方法。第 5.3.3 节中的增量碰撞检测思路是受到算法[639]和 V-Clip [249, 705]的启发。文献[169,308,389,408,416,705,810]介绍了距离计算方法。适合检测连杆自碰撞的方法出现在文献[656]中。运动规划中的碰撞检测的组合方法出现在文献[857]中。大量的碰撞检测程序包都可以应用于运动规划研究。其中应用最广泛的是 PQP，因为它可以应用于任意三维三角形[948]。

递增采样和搜索架构是通过将许多规划方法的思想进行统一而得到的。其中某些方法包括基于栅格的搜索[72,551,623]和随机路线图（PRM）[519]。尽管 PRM 是为多疑问而设计的，但是文献[127]设计的单疑问版本能够有助于显现与以前规划方法之间的联系。这甚至导致了基于栅格的 PRM 的变形[125,603]。文献[413]提出了另外一种单疑问的变化形式。

RDT 主要是作为 RRT 而发展的，文献[601,613]对此进行了介绍。RRT 已经用在多个应用中，有文献[84, 140, 152, 202, 226, 246, 267, 364, 395, 498, 501, 502, 531, 634, 644, 645, 919, 920, 949, 979, 986] 中所描述的许多变化形式。初始情况下，它们是为微分约束下的规划而设计的，但是到目前为止它们的大多数应用都是普通运动规划。更多关于有效最邻近搜索的信息，参见文献[478,47,48,49,53,101]和文献[232,367,479,541,761,909,989]。

第 5.6 节主要是基于 PRM 架构[519]。“概率”部分对这种方法来说并不关键；因此它在这里称为基于采样的路线图。与 PRM 有关的早期工作参见文献[392,393]。PRM 在实际中得到广泛应用，有文献[1, 23, 62, 63, 127, 163, 183, 246, 482, 547, 603, 630, 631, 743, 787, 795, 886, 901, 950, 971, 979, 995]中所描述的许多变化形式。文献[382]对某些变化形式进行了经验性的比较。文献[70,470,576]中对 PRM 进行了某些分析。在某些工作中，PRM 这个术语已经应用于任意基于采样的规划算法中（例如文献[470]）；但是近些年来，它的运用

与它在文献[519]中的原始意义更加一致了。

还有许多其他方法和问题，本章未予讨论介绍。几个基于近似胞腔剖分[146,330,649,661]的有趣方法可以认为是一种形式的基于采样运动规划。设计全局势函数的采样方法出现在文献[126]中。其他基于采样的规划算法出现在文献[196,350,420,421,466]中。一般情况下，本章的算法不能确定一个运动规划问题的解不存在。但是可能在有限时间内用基于采样的技术来得到解不存在的结论[77]。在基于采样的运动规划中也研究了并行化的问题[84,179,185,259,798]。

习　题

1. 证明一个度量空间的笛卡儿积（采用式（5.4）中那样的一个线性组合）是一个度量空间。

2. 证明（或证明相反的内容）：如果 ρ 是一个度量，那么 ρ^2 也是一个度量。

3. 确定如下函数是否是任意拓扑空间上的一个度量：$X:\rho(x,x')=1$ 为 $x\neq x'$；否则 $\rho(x,x')=0$。

4. 论述和证明式（5.28）是否能够在 C=$SE(3)$上产生一个度量，假设两个集合是刚体。

5. 式（5.19）中给出的离散度的定义是基于最坏情况。考虑定义一个平均离散度：

$$\bar{\delta}(P)=\frac{1}{\mu(X)}\int_X \min_{p\in P}\{\rho(x,p)\}\,\mathrm{d}x \tag{5.42}$$

描述近似计算式（5.42）的蒙特卡罗（随机化的）方法。

6. 确定[0,1]/～上的 Van der Corput 序列（基 2）的平均离散度（作为 i 的函数）。

7. 显示使用 $\mathbb{S}^3$ 上的 Lebesgue 测度能够产生 $SO(3)$的 Haar 测度。

8. Haar 测度在一个合适的 C-空间度量的选择中是否有用？试进行解释。

9. 在[0,1]/～上基-p（图 5.2 显示的是基-2）Van der Corput 序列中，确定第 i 个采样（最坏情况）的离散度的一个表达式。

10. 确定[0,1]上如下序列的离散度。第一个点上 $\alpha(1)$=1。对于每个 $i>1$，令 c_i=ln(2i–3)ln4 和 $\alpha(i)=c_i-[c_i]$。这种序列实现最好的渐近离散度，即使按照前述的常数，并且点不是均匀分布。你能解释会发生这种情况吗？绘制序列中的点可能是有帮助的。

11. 证明式（5.20）成立。

12. 证明式（5.23）成立。

13. 说明对于 $[0,1]^n$ 中任意给定的点的集合，可以设计一个范围空间 $\mathcal{R}$ 以使差异接近于 1。

14. 假设 $\mathcal{A}$ 是 $\mathbb{R}^3$ 中的一个刚体，其方向由一个四元数 h 给定。假设 h 被一个小扰动作用得到另外一个四元数 h'（没有平移）。构造关于 $\mathcal{A}$ 上的点运动的距离的一个良好上界，用四元数的变化来表达。

15. 设计 $\mathcal{W}$ 中机器人和障碍物的组合，它使 C-空间障碍物类似于 bug 陷阱。

16. 在一个 n 维栅格中至少能有多少 k-邻域（$1\leqslant k\leqslant n$）？

17. 在一个高维栅格中，考虑所有 3^n–1 个 n-邻近点就变得过于复杂。仅考虑 2n 个 1- 邻

近点又可能不够。确定一个方案来选择空间分布良好而又不能太多的邻域。例如对于 $\mathbb{R}^{10}$ 中一个栅格，用什么方法选择 50 个邻近点？

18. 搜索一个隐含的、高分辨率栅格和不用栅格而直接在 C-空间上生长搜索树，解释说明这两种方法之间的区别。

19. 旋转多点描绘出一个圆，而不是一条直线，通过考虑这个事实来改进式（5.31）中的界限。

20.（待解决的问题）证明从 $\mathbb{R}^n$ 中一个“无穷大”的 n 维球的中心开始的一个 RRT 有 $n+1$ 个主分支。分支的方向对准以初始位形为中心的一个正则单形的顶点。

实现

21. 实现凸多边形的二维增量碰撞检查来得到“几乎固定的时刻”性能。

22. 实现基于采样的路线图方法。选择一个合适的运动规划问题族：二维刚体、二维物体链、三维刚体等。

（a）比较用基于可视采样所得到的路线图和用平常采样方法所得到的路线图。

（b）研究这种方法对特定 NIGHTBORHOOD 方法的灵敏性。

（c）比较随机采样方法和确定性采样方法。

（d）用 bridge-test 来产生更好的采样。

23. 实现平衡的双向 RRT 规划算法。

（a）改变沿着边所创建的中间顶点的数量，研究改变所产生的效果。

（b）尝试用更有力的下降函数来连接随机采样点。

（c） 研究用 Kd-树来选择最近邻近点所得到的性能。

24. 设计一个基于 RRT 的规划算法，它用两个以上的树。仔细考虑诸如树的最大数量、什么时候开始一棵树、什么时候进行树的连接等问题。

25. 实现可扩张空间规划器和 RRT，并指导进行关于规划问题的有关实验。对于问题的全部集合，保持算法参数固定。

26. 实现一个基于采样的算法，它计算一个刚体机器人的无碰撞路径，机器人能够在如图 4.5 所示的任意平直二维流形上进行平移和旋转。

第 6 章　组合运动规划

运动规划的组合方法不借助于近似来寻找连续位形空间的路径。因此它们也被称为精确算法。这不同于第 5 章的基于采样的运动规划算法。

6.1 简　　介

本章所有算法都是完备的，这意味着对于任何问题实例（在设计算法的问题空间上），算法要么发现问题的解，要么正确地报告问题无解。相比较而言，基于采样的规划算法只允许更弱的完备性：分辨率完备性和概率完备性。

表示方法非常重要　当研究组合运动规划算法时，需要仔细考虑输入的定义。用什么来表示机器人和障碍物？对机器人采用什么变换集？空间的维数是多少？机器人和障碍物是凸的吗？它们是分段线性的吗？可能的输入定义了问题实例的一个集合，算法就是在这些实例上进行的。如果实例有适当的特性（例如低维性、凸模型），那么组合算法可以提供一个非常好的、实用的解。如果实例的集合太广泛，那么同时要求解的完备性和实用性可能是不合理的。一般运动规划问题的一些普通表述是 PSPACE 复杂度的[1]；因此这样的期望是难以实现的。尽管如此，仍然存在一般的、完备的运动规划算法。组合运动规划算法关注于表示，这与基于采样的规划不同，基于采样的规划将这些问题隐藏在碰撞检测中。

研究组合方法的原因　研究运动规划的组合方法通常有两个原因：

（1）在许多应用中，可能只对特殊类型的规划问题感兴趣。例如空间是二维的，并且机器人只能进行平移。对于一些特殊类型，可以得到非常有效的算法。这些算法是完备的，它们不依赖于近似，并且性能比第 5 章基于采样的规划方法更好。

（2）如果知道对于一个非常广泛的运动规划问题来说存在完备的算法，那么这是非常有趣而且令人满意的。因此即使对所感兴趣的问题的类型没有某些特殊的限制假设，也仍然存在解决它的一般工具和算法。这些算法也提供关于解决运动规划问题所用时间的理论上界。

警告：一些方法是不切实际的　当研究本章的算法时必须小心以避免产生错误假设。一些算法是有效的，也很容易实现，但是另外一些却既不有效，也不容易实现。即使一个算法有一个令人惊奇的渐近运行时间，它也可能难以实现。例如，来自于计算几何的最著名的一种算法能够在 $O(n)$时间内将一个有 n 条边的简单[2]多边形剖分为多个三角形[192]。它非常令人惊奇，《纽约时报》都对此进行了报道，但是该算法过于复杂以至于它的可实现性受到了人们的置疑。有时使用一种比较耗时但容易理解和实现的算法是可取的。一般情

1　即 NP 复杂度。关于这种复杂性的概述将出现在第 6.5.1 节。

2　一个没有孔的多边形区域。

况下，理解这两种情况并进行权衡折中来确定采用哪种方法，这是非常有价值的。试图确定解决一个问题的效率，这本身就是一个有趣的智力追求，即使结果只有理论意义。这促使人们去寻找更简单，且与前面所述算法有相同或者相似的渐近运行时间的算法。

路线图 实际上，所有组合运动规划方法沿着解决疑问的方式构造一个路线图。路线图的概念在第 5.6 节已经介绍过，但是本章的路线图定义更加严格，因为构造的任何算法都需要是完备的。本章的某些算法首先构造$\mathcal{C}_{free}$的一个胞腔剖分，路线图就由此产生。其他方法则直接构造路线图而不需要考虑胞腔。

令$\mathcal{G}$是映射到$\mathcal{C}_{free}$的一个拓扑图（在例 4.6 中定义），并且令$S \subset \mathcal{C}_{free}$是刈幅，它是$\mathcal{G}$可以到达的所有点的集合，如式（5.40）所定义。图$\mathcal{G}$称为一个路线图，如果它满足两个重要条件：

（1）可达性：从任何$q \in \mathcal{C}_{free}$都很容易、有效地计算一条路径$\tau:[0,1] \rightarrow \mathcal{C}_{free}$，其中$\tau(0)=q$，$\tau(1)=s$，$s$是$S$中的任意一点。通常$s$是最靠近$q$的点，假设$\mathcal{C}$是一个度量空间。

（2）确保连通性：使用第一个条件，那么总可能将某q_I和q_G分别连接到S中的某s_1和s_2。第二个条件要求如果存在一条路径$\tau:[0,1] \rightarrow \mathcal{C}_{free}$满足$\tau(0)=q_I$和$\tau(1)=q_G$，那么也存在一条路径$\tau':[0,1] \rightarrow S$满足$\tau'(0)=s_1$和$\tau'(1)=s_2$。那么$\mathcal{G}$就不会丢失$\mathcal{C}_{free}$的连通性，从而也就不会漏掉问题的解。这确保得到的是完备算法。

满足这些性质，路线图就提供了连续运动规划问题的一个离散表示，而不损失解决问题所需要的任何原始的连通信息。对于一个疑问(q_I, q_G)，如果将每个疑问点连接到路线图，然后在$\mathcal{G}$上进行离散图的搜索，那么可以解决该疑问。为了保持完备性，第一个条件确保任何疑问都可以连接到$\mathcal{G}$，第二个条件确保如果解存在，搜索总能够成功。

6.2 多边形障碍区域

首先我们并不急于研究一般形式的组合运动规划，而是对一个比较容易形象化的情况阐述几种解决方法。对于$\mathcal{C}=\mathbb{R}^2$以及$\mathcal{C}_{obs}$是多边形的情形，存在几种性能良好而且简单的算法。这些方法大多数都不能直接扩展到高维；但是一些一般原则却是相同的。因此研究二维情况下组合运动规划方法是非常有启发意义的。在一些应用中，也可以直接采用这些算法。一个例子是对于可建模为一个点的可移动机器人，对它在建筑物内的运动进行规划，建筑物可以建模为二维多边形平面图，这个例子就可以直接采用上面所提到的算法。

第 6.2.1 节介绍了表示方法后，第 6.2.2 节～第 6.2.4 节提出解决同一个问题的 3 种不同算法。第 6.2.2 节的算法在建立路线图的过程中首先进行胞腔剖分，第 6.2.3 节和第 6.2.4 节中的另外两种算法直接产生路线图。其中第 6.2.3 节中的算法计算最大间隙路径，第 6.2.4 节中的算法则计算最短路径（该路径没有间隙）。

6.2.1 问题的表示

假设$\mathcal{W}=\mathbb{R}^2$；障碍物$\mathcal{O}$是多边形；机器人$\mathcal{A}$是一个仅能够进行平移的多边形物体。在这些假设条件下，$\mathcal{C}_{obs}$将是多边形的。对于$\mathcal{A}$是$\mathcal{W}$中的一个点这种特殊情形，$\mathcal{O}$直接映射到

C_{obs} 而不会有任何变形。因此这里考虑的问题也可以认为是对点机器人的规划。如果 $\mathcal{A}$ 不是点机器人，那么必须计算 $\mathcal{O}$ 和 $\mathcal{A}$ 的 Minkowski 差（见式（4.37））。对于 $\mathcal{A}$ 和 $\mathcal{O}$ 的每个分支都是凸的情形，可以采用第 4.3.2 节的算法来计算 C_{obs} 的每个分支。一般情况下，$\mathcal{A}$ 和 $\mathcal{O}$ 都是非凸的。它们甚至可能还包含有孔，这些孔产生了如图 6.1 所示的一个 C_{obs} 模型。在这种情况下，$\mathcal{A}$ 和 $\mathcal{O}$ 可以剖分为凸分支，并且可以计算每个分支对的 Minkowski 差。剖分为凸分支的过程可以通过修改第 6.2.2 节的胞腔剖分算法来实现。一旦计算了 Minkowski 差，那么将它们合并就可以得到按照图 6.1 中那样的简单多边形所进行的表示。文献[266]的第 2.4 节给出了一个有效地实现这种合并的算法。它也是基于一些与第 6.2.2 节中的规划算法相同的原理。

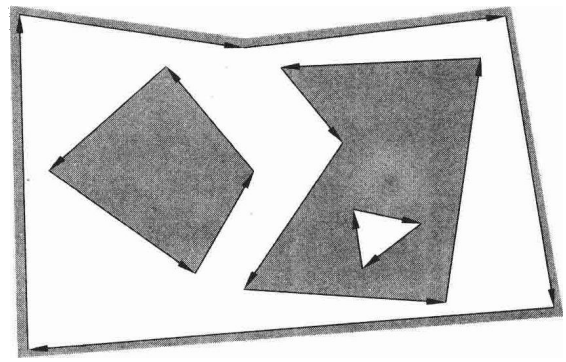


图 6.1　由四个有向简单多边形指定的一个多边形模型

为了实现这里所描述的算法，需要有一个便于访问信息的数据结构，信息是从图 6.1 那样的模型中得到的。如何表示外部边界？如何表示障碍物内部的孔？我们怎样知道哪个孔是哪个障碍物的内部？通过用第 3.1.3 节所描述的二重连接边的列表数据结构，这些问题都可以得到有效的回答。我们需要表示图 6.1 那样的模型以及在规划算法执行过程中需要保持的任何其他信息。有 3 种不同的记录：

- **顶点**：每个顶点 v 包含一个指向点 $(x, y) \in \mathcal{C} = \mathbb{R}^2$ 的指针和一个指向以 v 作为起始点的某半边的指针。
- **面**：每个面有一个指向包围这个面的边界上的半边的指针；如果面是最外面的边界，那么指针值为 NIL。为每个包含在面内的连通分支（例如孔），面也包含一个指针列表。列表中的每个指针指向分支边界的一个半边。
- **半边**：每个半边都是有向的，这使得障碍部分总在它的左边。它包含 5 种不同的指针。有一个指针指向它的起始顶点。有一个半边指针对，它们分别指向两个相反方向的半边（参见第 3.1.3 节）。如果半边与障碍接壤，那么该指针是 NIL。半边总是以环形链的形式排列来形成一个面的边界。这种链是定向的，这使得障碍部分（或一个“双胞胎”半边）总在其左边。每个半边保存了一个指向内面的指针。它还包含指向半边环形链中下一个和上一个半边的指针。

对于图 6.1 所示的例子，有 4 个半边环形链，每个链界限了一个不同的面。小的三角形孔的面指向包含孔的障碍物的面。每个障碍物包含一个指向用最外面的边界所表示的面的指针。通过一致地分配半边的指向，界限障碍物的环形链总是沿逆时针方向，界限孔的

链沿顺时针方向。不存在“双胞胎”半边，因为所有的半边都界限$\mathcal{C}_{obs}$。二重连接边的列表数据结构非常具有一般性，它允许插入切入$\mathcal{C}_{free}$的额外的边。这些边不在$\mathcal{C}_{obs}$的边界上，但是它们可以用“双胞胎”半边的指针进行合并。这对第 6.2.2 节中的算法非常有用。

6.2.2 垂直胞腔剖分

胞腔剖分将在第 6.3 节进行正式定义，但是这里我们采用其非正式的概念。组合方法必须构造一个能够精确编码规划问题的有限数据结构。胞腔剖分算法将$\mathcal{C}_{free}$分拆为区域的一个有限集合，这种区域被称为胞腔（或单元）。k-胞腔是指 k 维胞腔。胞腔剖分应该满足 3 个性质：

（1）从胞腔中的一个点到另外一个点的路径的计算必须很容易。例如，如果胞腔是凸的，那么胞腔中的任何点对都可以用一条线段来连接。

（2）容易提取胞腔的邻接信息来建立路线图。

（3）对于给定的q_I和q_G，应该能够非常有效地确定它们包含在哪个胞腔里。

如果胞腔剖分满足这些性质，那么运动规划问题就退化成图的搜索问题。可以采用第 2.2 节的算法；只是在目前情况下，通常提前知道整个图$\mathcal{G}$[1]。而对离散规划问题来说并不这样假设。

定义垂直剖分 下面提出构造垂直胞腔剖分的算法[191]，它将$\mathcal{C}_{free}$分拆为 1-胞腔和 2-胞腔的有限族。每个 2-胞腔是一个有垂边的梯形或者三角形（一个退化的梯形）。因为这个原因，该方法有时称为梯形剖分。剖分的定义如下。令 P 表示用来定义$\mathcal{C}_{obs}$的顶点的集合。在每个顶点$p \in P$，尝试将射线向上、向下延伸穿过$\mathcal{C}_{free}$直到碰到$\mathcal{C}_{obs}$。根据它是否可能向两个方向延伸，有四种可能的情况，如图 6.2 所示。如果将$\mathcal{C}_{free}$按照这些射线进行分拆，那么就产生了垂直剖分。为图 6.3(a)中的例子延伸这些射线就产生了$\mathcal{C}_{free}$的剖分，如图 6.3(b)所示。应注意到，对于$\mathcal{C}_{free}$中的 2-胞腔只能得到梯形和三角形。

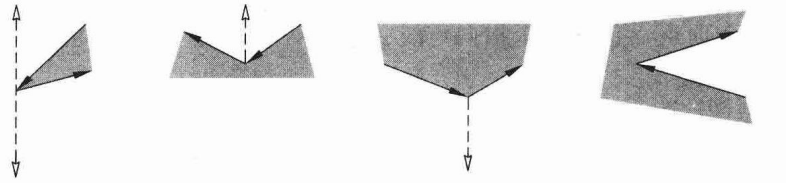
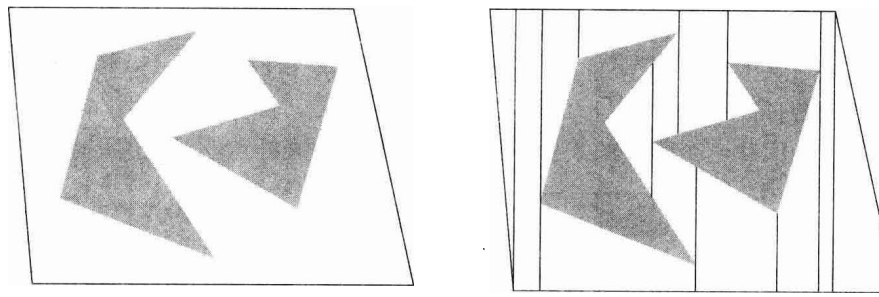


图 6.2 有四种一般情况：（1）能够向上和向下进行延伸；（2）只能向上；（3）只能向下；（4）不可能进行延伸

每个 1-胞腔是一个垂直线段，它是两个 2-胞腔之间的边界。我们必须确保能够正确地表示$\mathcal{C}_{free}$的拓扑。回顾到$\mathcal{C}_{free}$定义为一个开集合。每个 2-胞腔实际上是定义为$\mathbb{R}^2$上的一个开集合；因此它是一个梯形或者三角形的内部。1-胞腔是线段的内部。0-胞腔对应于线段端点，但是它们是不允许的，因为它们在$\mathcal{C}_{obs}$中。

最广位置问题 如果沿着$\mathcal{C}_{obs}$的两个点都在切入$\mathcal{C}_{free}$的一条垂线上，那么会怎样？当$\mathcal{C}_{obs}$的其中一条边是垂直的，那么会发生什么？到目前为止一直都忽略了这些特殊的情况。

1 例外情况是第 6.5.3 节所提及的某些算法，它们是通过只保持$\mathcal{C}_{obs}$的一个连通分支来提高效率的。



(a)　(b)

图 6.3　垂直胞腔剖分方法用胞腔来构造一个路线图，搜索路线图产生一个疑问的解

一些组合运动规划经常会忽略这些特殊情况，并且假设C_{obs}处在最广位置上。这通常意味着如果所有的数据点在某随机方向上有一个小的扰动，那么保持特殊情况的概率是 0。因为经过扰动后垂直边不再垂直，那么它就不处在最广位置上。进行最广位置假设是因为这样可以简化算法的表示（在某些情况下，它的渐近运行时间会更小）。实际情况下这种假设可能是无效的。大多数实现时间通常都用于正确处理这些特殊情况。进行随机扰动可以避免这个问题，但是它使求解过程更加复杂。对于垂直剖分，不借助于扰动，问题也并不太难处理；但是一般情况下，意识到这种困难是非常重要的，这些困难在多数情况下都难以排除。

定义路线图　为了处理运动规划疑问，从垂直胞腔剖分中构造一个路线图。对于每个胞腔 C_i，令 q_i表示一个采样点 $q_i \in C_i$。采样点可以作为胞腔的质心，但是并不需要进行特别的选择。令$\mathcal{G}(V, E)$表示如下定义的一个拓扑图。对于每个胞腔 C_i，定义一个顶点 $q_i \in V$。每个 1-胞腔和 2-胞腔都有一个顶点。对于每个 2-胞腔，从它的采样点到沿着它的边界的每个 1-胞腔的采样点定义一条边。每条边是胞腔的采样点间的一条线段路径。这样的图就是一个路线图，如图 6.4 所示。由于每个胞腔所具有的凸性，通过直线路径就可以到达每个采样点，所以满足可达性条件。连通性条件也是满足的，因为$\mathcal{G}$直接从胞腔剖分中产生，而这种剖分保持了$\mathcal{C}_{free}$的连通性。一旦构造了路线图，求解过程就不再需要胞腔信息了。

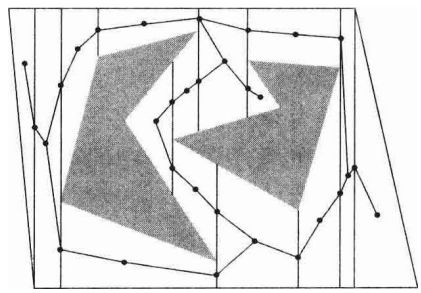


图 6.4　从垂直胞腔剖分产生的路线图

疑问的求解　一旦得到路线图，解决运动规划疑问(q_I, q_G)就容易了。令 C_0和 C_k分别表示包含 q_I和 q_G的胞腔。在图$\mathcal{G}$中，搜索一条连接 C_0的采样点和 C_k的采样点的路径。如果这样的路径不存在，那么规划算法报告无解。如果存在，那么令 $C_1, C_2, \cdots, C_{k-1}$表示沿着从 C_0到 C_k所计算出来的路径访问 1-胞腔和 2-胞腔的序列。

简单地“连接这些点”就可以形成一条路径解。令 $q_0, q_1, q_2, \cdots, q_{k-1}, q_k$ 表示$\mathcal{G}$中沿着路径的采样点。每个胞腔有一个采样点。通过设定$\tau(0)=q_I$和$\tau(1)=q_G$、沿着最短路径访问从 q_0 到 q_k 的序列中的每个点就可以形成路径解$\tau:[0,1]\rightarrow\mathcal{C}_{free}$。例如这产生图 6.5 所示的解。在选择采样点时，确保从一个胞腔的采样点到其邻近胞腔的采样点的路径线段都是无碰撞的[1]。

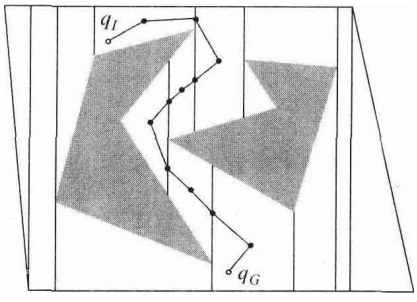


图 6.5 一条路径解

计算剖分 到现在还没有考虑有效地计算剖分的问题。如果不考虑计算的效率问题，那么问题就非常简单，所有的步骤都可以用蛮力计算。如果$\mathcal{C}_{obs}$有 n 个顶点，那么这种方法至少耗费 $O(n^2)$时间，因为在每条垂直射线与每个线段之间不得不进行交集检验。这忽略了涉及寻找包含疑问点的胞腔的数据结构问题以及建立保持连通信息的路线图的数据结构问题。仔细对计算进行组织，所有这些问题都可以得到处理，相应的计算时间仅为 $O(n \log n)$。

平面扫描原理 本算法基于计算几何中的平面扫描（或者线扫描）原理[129,266,304]，它是一些组合运动规划算法以及其他一些一般算法的基础。计算几何中的许多内容都可以看作是将排序问题推广到多维情况的数据结构和算法。也就是说，算法对几何信息进行仔细“排序”。

之所以用“扫描”这个词来描述这些算法是因为可以假设一条直线（平面）扫过整个空间，仅在信息的某临界变化发生时才停止。当然这只是一个直观假设，其中的扫描线并未明确给出。为了构造垂直剖分，假设有一条垂线从 $x=-\infty$到 $x=\infty$进行扫描，(x, y)表示 $\mathcal{C}=\mathbb{R}^2$ 中的一个点。

在第 6.2.1 节中，$\mathcal{C}_{obs}$的顶点的集合 P 是 $\mathbb{R}^2$ 中仅有的出现在问题输入中的数据。因此仅在这些点上发生有趣的事情似乎是有道理的。将这些点按 X 轴坐标进行排序。假设没有两个点的X轴坐标相同。以x的值递增的顺序访问P中的点。每次访问就是一个事件(event)。在每个事件之前、之后和之中，$\mathcal{C}_{obs}$ 的一些边的列表得以保持。在所有时间里都必须以边被垂直扫描线穿刺的顺序保持这个列表。保持的顺序是从低到高。

算法执行 图 6.6 和图 6.7 显示了算法是如何进行的。初始情况下，L 为空，二重连接边的列表用来表示$\mathcal{C}_{free}$。$\mathcal{C}_{free}$的每个连通分支在数据结构中产生一个面。我们可以归纳性地猜想到，数次事件发生后，L 得以正确保持。对于每次事件，发生图 6.2 中的四种情况之一。通过在平衡的二叉搜索树中保持 L[245]，可以在 $O(\lg n)$时间内确定 p 上方和下方的边。这

1 这是方法的定义为什么不同于文献[590]的第 1 章的原因。在那里，采样点没有放置在 2-胞腔的内部，并且对于某些疑问会产生碰撞。

比对每条线段都进行检查所花费的 $O(n)$时间好得多。根据图 6.2 中四种情形所发生的不同，对 L 的更新也不同。如果发生第一种情形，那么插入两种不同的边，面被垂直线段分拆两次，该面的 p 点在边界上。对于两条垂直线段的每条而言，增加两个半边，并且所有面和半边都必须正确更新（这种运算是局部性的，仅有邻近于发生变化的那些记录需要更新）。图 6.2 中的后面两种情形比较简单；仅有单个面的分拆。对于最后一种情况，则没有发生分拆。

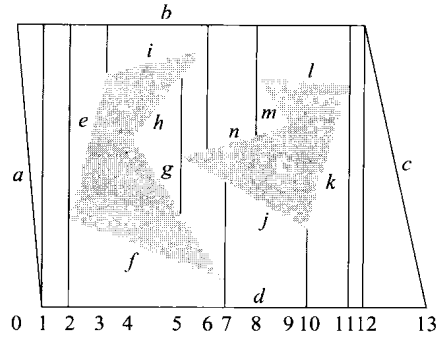


图 6.6　这个例子中有 14 个事件

事件	L 中以事件进行排序的边	事件	L 中以事件进行排序的边
0	$\{a, b\}$	7	$\{d, j, n, b\}$
1	$\{d, b\}$	8	$\{d, j, n, m, l, b\}$
2	$\{d, f, e, b\}$	9	$\{d, j, l, b\}$
3	$\{d, f, i, b\}$	10	$\{d, k, l, b\}$
4	$\{d, f, g, h, i, b\}$	11	$\{d, b\}$
5	$\{d, f, g, j, n, h, i, b\}$	12	$\{d, c\}$
6	$\{d, f, g, j, n, b\}$	13	$\{\}$

图 6.7　14 个事件发生以后就显示出 L 的状况。在第一个事件发生之前 L 是空的

一旦完成面的分拆，那么就需要更新 L。当扫描线穿过 p，有两条边受到影响。例如图 6.2 中的第一种和最后一种情形，两条边必须插入 L（这些情形的镜像引起两条边从 L 中被删除）。如果中间两种情形发生，那么一条边被 L 中的另外一条边所代替。这些插入和删除运算都能够在 $O(\lg n)$时间内完成。因为有 n 个事件，构造算法的运行时间是 $O(n \lg n)$。

路线图$\mathcal{G}$可以从二重连接边的列表的面指针来计算。一个更好的方法是在每次事件递增建立$\mathcal{G}$。实际上，如果需要，可以不用保持用来得到一个一致的二重连接边的列表所需要的所有指针，只要正确建立$\mathcal{G}$，并且每个胞腔都有采样点。我们甚至可以向前更进一步，不进行胞腔剖分而直接建立邻近胞腔的所有采样点之间的线段路径的一个拓扑图。

6.2.3　最大间隙路线图

最大间隙路线图（maximum-clearance roadmap）试图尽可能远离C_{obs}，如图 6.8 所示的走廊。在可移动机器人应用中有时更宁愿选择这种路径，因为很难测量和控制可移动机器人的精确位置。沿着最大间隙的路线图行进可以减少由于这些不确定性而引起碰撞的机会。

这种路线图的另外一个名字是广义 Voronoi 图和收缩法（retraction method）[752]。它可以看作是 Voronoi 图从点推广到多边形情形（回忆第 5.2.2 节）。路线图的每个点与C_{obs}边界上两个点之间是等距的。每个路线图顶点对应于两个或者两个以上路线图边的交集，因此与C_{obs}边界上三个或者三个以上的点的距离是相等的。

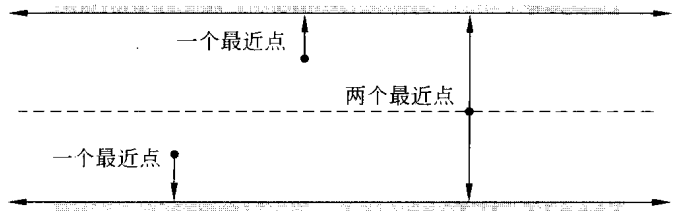


图 6.8 最大间隙路线图离C_{obs}尽可能远；这涉及沿着与C_{obs}的边界上两个或者多个点等距的那些点运动

收缩这个词来自于拓扑，它提供了对该方法的一个直观描述。一个子集 S 是一个拓扑空间 X 的变形收缩，如果如下连续同伦 $h: X\times[0,1]\rightarrow X$ 可以定义为[454]：

（1）对于所有 $x\in X$，$h(x, 0)= x$。

（2）$h(x, 1)$是一个将 X 的每个元素映射到 S 的某个元素的连续函数。

（3）对所有 $x\in[0,1]$，对于任意 $s\in S$，$h(x, t)=s$。

直观上，C_{free}在同伦过程中逐渐变薄，直到得到一个骨架 S。这种收缩过程的一个近似可以形象地描述为在整个C_{free}的边界周围削去薄薄的一层。如果重复迭代该收缩过程，那么留下来的仅剩最大间隙路线图（假设当得到薄的“切片”时切削过程总会停止）。

为了构造最大间隙路线图，再次用到第 5.3.3 节中的要素这个概念。令要素集指的是C_{obs}的所有边和顶点的集合。每对要素能够产生路线图的候选路径。这导致如下 $O(n^4)$时间算法。对于每个边-边要素对，生成如图 6.9(a)所示的一条直线。对于每个顶点-顶点对，生成如图 6.9(b)所示的一条直线。一个点和一条直线之间的最大间隙路径是一个抛物线。因此，对于每个边-点对，生成的一个抛物线如图 6.9(c)所示。落在最大间隙路线图上的那部分路径由曲线相交来确定。存在几种有良好渐近运行时间的算法[619,629]，但是它们更加难以实现。公认的算法运行时间是 $O(n \lg n)$，其中 n 是路线图曲线的数量[867]。

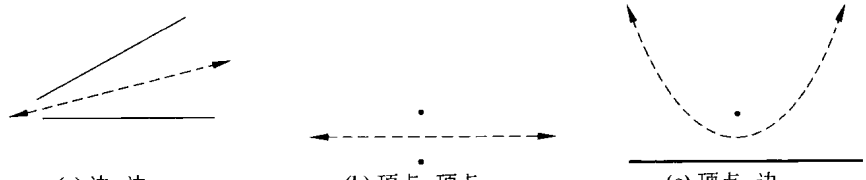


图 6.9 生成 Voronoi 路线图线段的三种可能的情况；第三种情况产生一条二次曲线

6.2.4 最短路径路线图

设想我们的目标是寻找最短路径，而不是生成最大化间隙的路径。这就产生最短路径路线图，在文献[591]中它也称为简化的可视图（reduced visibility graph）。该方法的思路首先由文献[745]提出，并且可能是运动规划算法的第一个例子。最短路径路线图与最大间隙

是相互矛盾的，因为最短路径趋向于掠过C_{obs}的角。实际上这里提出的问题存在缺陷，因为C_{free}是一个开集。对于任意路径τ:[0,1]→C_{free}，总有可能发现一条更短的路径。因此我们必须考虑在C_{free}的闭包 $cl(C_{free})$中确定最短路线图的问题。这意味着允许机器人“接触”或“掠过”障碍物，但是不允许刺入它们。为了将计算出来的路径作为运动规划问题的解，需要将它们细微调整到距离C_{obs}很近但不接触的程度。这稍微增加了路径的长度，但是当路径离C_{obs}任意近时，路径长度的增加可以任意小。

最短路径路线图$\mathcal{G}$是这样构造的。用反射顶点（reflex vertex）表示那些内角（在C_{free}中）大于 π 的多边形顶点。凸多边形的所有顶点（假设 3 个相继的顶点不共线）都是反射顶点。$\mathcal{G}$的顶点是反射顶点。$\mathcal{G}$的边有两种不同来源：

- **相继反射顶点**——如果两个反射顶点是C_{obs}的一条边的端点，那么在$\mathcal{G}$中这两个顶点之间构造一条边。
- **双切线的边**——如果通过一对反射顶点画出一条双切线（bitangent line），那么在$\mathcal{G}$中构造一条相应的边。如图 6.10 所示，双切线是一条直线，它入射到两个反射顶点但并不刺入C_{obs}的内部，并且这些顶点必须是相互可视的。

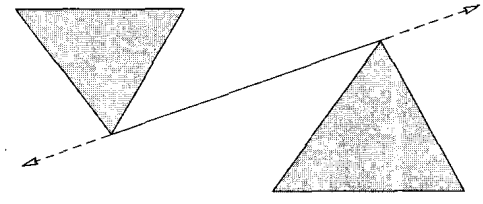


图 6.10　一个双切线边必须接触相互可见的两个反射顶点，线必须向外延伸经过顶点而不能刺入C_{obs}

产生的路线图的一个例子如图 6.11 所示。注意到路线图可能有孤立顶点，如图的顶部的那个顶点。为了解决疑问，将 q_I和 q_G连接到所有可见的路线图顶点，如图 6.12 所示。这为搜索解路径而产生了一个扩展路线图。如果采用 Dijkstra 算法，并且对每条边都给出对应于它的路径长度的一个代价，那么所产生的路径解就是 q_I 和 q_G 之间的最短路径。图 6.12 例子的最短路径如图 6.13 所示。

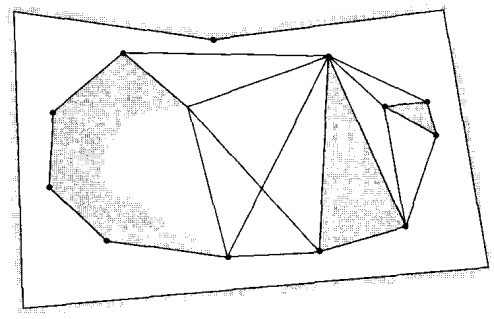


图 6.11　最短路径路线图包括C_{obs}上相邻反射顶点之间的边和双切线边

如果对双切线进行检验，那么产生的算法要求 $O(n^3)$时间，其中 n 是C_{obs}的顶点的数量。有 $O(n^2)$个反射顶点对需要进行检测，每次检测需求 $O(n)$时间来确保再没有其他边妨碍它

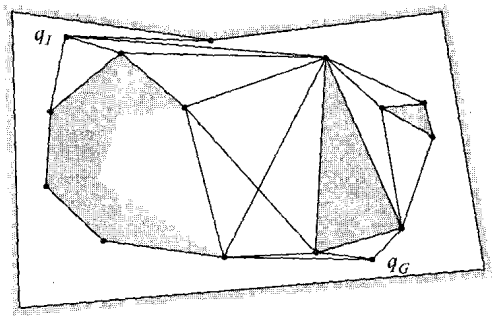


图 6.12 为了解决一个疑问，q_I和 q_G连接到所有可见路线图顶点上，并进行图的搜索

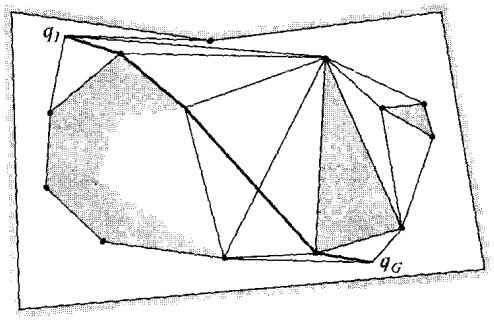


图 6.13 在扩展的路线图中最短路径就是 q_I和 q_G之间的最短路径

们的可视性。对第 6.2.2 节的平面扫描原理进行调整还可以得到更好的算法，算法所花费的时间仅 $O(n^2 \lg n)$。思路是从每个反射顶点 v 进行放射状扫描。射出的线从 $\theta=0$ 开始，当射线触到顶点时，事件发生。用这种方法可以在 $O(n \lg n)$时间内计算出经过 v 的双切线的集合。因为有 $O(n)$个反射顶点，总运行时间是 $O(n^2 \lg n)$。更详细内容见文献[266]的第 15 章。有一个算法可以在 $O(n \lg n+m)$内计算出最短路径路线图，其中 m 是路线图中边的总数[386]。如果障碍区域用简单多边形描述，那么时间复杂性可以减小到 $O(n)$；一些最短路径的变化形式和参考见文献[712]。

为了提高数值计算的鲁棒性，计算最短路径路线图可以不用三角函数。对于三点序列 p_1, p_2, p_3，定义左转谓词（left-turn predicate）$f_l:\mathbb{R}^2\times\mathbb{R}^2\times\mathbb{R}^2\to\{\text{TRUE, FALSE}\}$，当且仅当 p_3在从 p_1开始并刺入 p_2的射线的左边时 $f_l(p_1, p_2, p_3)$=TRUE。p_2点是一个反射顶点，当且仅当 $f_l(p_1, p_2, p_3)$=TRUE，其中 p_1和 p_3分别是沿着C_{obs}的边界 p_2点之前、之后的点。双切线检验通过分配如图 6.14 的点来实现。假设任何三点不共线且连接 p_2和 p_5的线段不冲突。顶点对 p_2和 p_5应该有一条双切线的边，如果如下命题是 FALSE：

$$\left(f_l(p_1,p_2,p_5)\oplus f_l(p_3,p_2,p_5)\right)\vee\left(f_l(p_4,p_5,p_2)\oplus f_l(p_6,p_5,p_2)\right) \tag{6.1}$$

其中⊕表示逻辑“异或”。f_l谓词的实现可以不用三角函数，而是通过定义：

$$M(p_1,p_2,p_3)=\begin{pmatrix}1 & x_1 & y_1\\ 1 & x_2 & y_2\\ 1 & x_3 & y_3\end{pmatrix} \tag{6.2}$$

其中 $p_i=(x_i, y_i)$。如果 $\det(M)>0$，那么 $f_l(p_1, p_2, p_3)$=TRUE；否则 $f_l(p_1, p_2, p_3)$=FALSE。

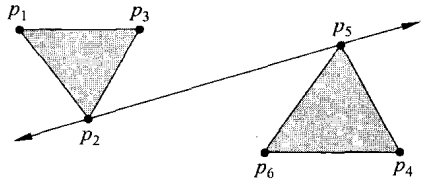


图 6.14　通过左转检验来识别潜在的双切线，这避免使用三角函数及与它们有关的数值问题

6.3　胞 腔 剖 分

第 6.2.2 节介绍当C_{obs}是多边形时，用垂直胞腔剖分来解决运动规划问题。但是应该理解到这仅仅是许多剖分方法中的一种。某些剖分方法对二维空间来说并不可取，但是它们可以推广到更高维数。因此这里介绍其他胞腔剖分方法，这些方法提供了从垂直胞腔剖分到第 6.4 节的圆柱代数剖分的平滑过渡，而圆柱代数剖分能够解决任意半代数模型、任意维数的运动规划问题。沿着该思路，为$\mathcal{W}=\mathbb{R}^2$中线段机器人这个特殊情形，第 6.3.4 节将介绍圆柱剖分方法。

6.3.1　一般定义

本节中的复形（complex）是指胞腔及其边界的一个族。复形可以分拆为胞腔，但是复形包含描述关于胞腔必须如何进行匹配的更多信息。胞腔剖分（cell decomposition）仍然指的是将空间分拆为胞腔，这些胞腔从复形中产生。

用一般的方式定义复形和胞腔剖分。假设C_{free}的任何分拆都可以叫做胞腔剖分。一个胞腔可能如此复杂以至于胞腔这个概念没有用处。甚至C_{free}自身就是一个大胞腔。因此从更简单的胞腔来建立剖分是很有用的，例如没有孔的胞腔。正式情况下，这要求每个 k 维胞腔与开 k 维单位球$B^k \subset \mathbb{R}^k$同胚。从运动规划的观点来看，这仍然产生非常复杂的胞腔，这需要特殊的胞腔剖分方法通过进一步的约束条件来产生一个完备规划算法。

这里介绍两种不同的复形。单纯复形（simplical complex）是最易理解的一种。尽管在许多应用中它非常有用，它却不能表示运动规划中出现的所有复形。因此我们也介绍奇异复形（singular complex）。尽管它的定义更加复杂，但是它却能够涵盖本书所感兴趣的所有胞腔复形。它也能够提供一个表示拓扑空间的方法。还有一种这里没有介绍的胞腔复形是 CW-复形[442]。

单纯复形　假设$X=\mathbb{R}^n$。令$p_1, p_2, \cdots, p_{k+1}$是$\mathbb{R}^n$中 k+1 个线性无关[1]的点。从下面这些点形成一个 k-单形$[p_1, \cdots, p_{k+1}]$：

$$[p_1, \cdots, p_{k+1}] = \left\{ \sum_{i=1}^{k+1} \alpha_i p_i \in \mathbb{R}^n \,\middle|\, \alpha_i \geq 0 \,,\ \sum_{i=1}^{k+1} \alpha_i = 1 \right\} \tag{6.3}$$

其中$\alpha_i p_i$是α_i与每个点坐标的标量乘。另外一种方法是将式（6.3）作为 k+1 点的凸壳（例

1　从其他 k 个点减去 p_1 后形成 k 个向量，正整数 $k \leq n$。将这些向量组成一个 $k \times n$ 矩阵。为了线性无关，至少必须有一个有非零行列式的 $k \times k$ 余子式。例如，如果 $k=2$，那么三个点不能共线。

如它们之间所有线性插值点)。如果 $k=2$，那么得到一个三角形区域。对于 $k=3$，则产生的是一个四面体。

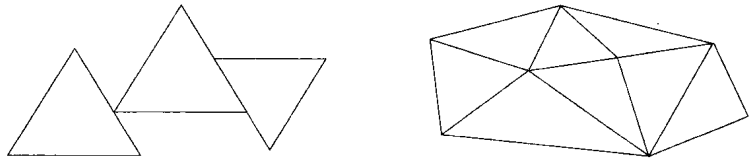
对于任何 k-单形（simplex）和任意 i，$1\leqslant i\leqslant k+1$，令 $\alpha_i=0$。这产生一个 $(k-1)$维单形，称为原始单形的一个面。一个 2-单形有三个面。每个面是一个称为边的 1-单形。每个 1-单形（或边）有两个面，它们是被称为顶点的 0-单形。

为了形成一个复形，单形之间必须相互匹配。这产生了一个叫做三角剖分（triangulation）的高维概念，在 $\mathbb{R}^2$ 中这就是一个由三角区域构成的瓦片覆盖。一个单纯复形 $\mathcal{K}$ 是满足如下条件的单形的有限集合：

（1）$\mathcal{K}$ 中的一个单形的任何面也在 $\mathcal{K}$ 中。

（2）$\mathcal{K}$ 中任何两个单形的交集是它们的公共面或者交集为空。

图 6.15 阐述了这些要求。对于 $k>0$，$\mathcal{K}$ 的 k-胞腔定义为任意 k-单形的内部 $\text{int}([p_1,\cdots,p_{k+1}])$。对于 $k=0$，每个 0-单形是一个 0-胞腔。这些胞腔的并集形成了 $\mathcal{K}$ 所覆盖的点集的一个分拆。与第 6.2.2 节一样，这也提供了一种胞腔剖分。



不是一个单复形　　一个单复形

图 6.15　为了成为一个单复形，单形面必须很好地匹配在一起

奇异复形　单纯复形在诸如几何建模和计算模型拓扑的计算机图形学等应用中很有用。由于运动规划中存在复杂的拓扑空间、隐含的非线性模型、剖分算法，对多数一般问题来说，单纯复形是远远不够的。奇异复形是对单纯复形的推广。奇异复形可以定义在任意流形 X 上（它甚至可以定义在任何 Hausdoff 拓扑空间上），而并不局限于 $\mathbb{R}^n$。主要的差别在于，对单纯复形来说，每个单形都是 $\mathbb{R}^n$ 的一个子集；但是对于奇异复形来说，每个奇异复形实际上是从 $\mathbb{R}^n$ 中的一个（单纯）复形到 X 的一个子集的一个同胚。

为了加深对该思路的理解，首先考虑一个 1 维奇异复形，它正好是一个拓扑图（如例 4.6 所介绍）。区间[0,1]是一个 1-单形，连续路径 $\tau:[0,1]\to X$ 是一个奇异 1-单形，因为它是[0,1]到 X 中 τ 的像的一个同胚。假设 $\mathcal{G}(V,E)$ 是一个拓扑图。胞腔是下面定义的 X 的子集。每个点 $v\in V$ 是 X 中的一个 0-胞腔。以这种形式，每个点都可以认为是函数 $f:\{0\}\to X$ 的像，该函数使它成为一个奇异 0-单形，因为{0}是一个 0-单形。对于每个路径 $\tau\in E$，相应的 1-胞腔是：

$$\{x\in X \mid \tau(s)=0 \quad \text{对一些} \quad s\in(0,1)\} \tag{6.4}$$

即它是路径 τ 的像 $\tau((0,1))$，除了端点被删除以外，因为它们已经被 0-胞腔所覆盖（胞腔必须形成一个分拆）。

这些原理将被推广到更高维数上。因为所有相同维数的球和单形是同胚的，所以在奇异单形的定义中可以用球来代替单形。令 $B^k\subset\mathbb{R}^k$ 表示一个闭的 k-维单位球：

$$D^k=\left\{x\in\mathbb{R}^n \mid \|x\|\leqslant 1\right\} \tag{6.5}$$

其中$\|\cdot\|$表示欧氏范数。一个奇异 k-单形是一个连续映射 $\sigma:D^k\rightarrow X$。令 $\text{int}(D^k)$表示 D^k 的内部。对于 $k\geqslant 1$，对应于一个奇异 k-单形 σ 的 k-胞腔 C 是像 $C=\sigma(\text{int}(D^k))\subset X$。0-胞腔作为 0 奇异单形的像而直接得到。每个奇异 0-单形映射到 X 中的 0-胞腔。如果 σ 限制在 $\text{int}(D^k)$，那么它实际上定义了 D^k 和 C 之间的一个同胚。如果 $k>0$，这两种情况都是空集。

一个单纯复形要求单形之间良好匹配。这里也如此，但是采用的是拓扑概念，因为这更具一般性。令$\mathcal{K}$表示可变维数的奇异单形的一个集合。对于所有 $i\leqslant k$，令 S_k 表示所有奇异 i-单形的像的并集。

映射到拓扑空间 X 的一个奇异单形族称为一个奇异复形，如果：

（1）对每个维数 k，集合 $S_k\subseteq X$ 必须是闭的。这意味着胞腔之间必须良好匹配。

（2）每个 k-胞腔是拓扑子空间 S_k 中的一个开集。0-胞腔在 S_0 中是开的，即使它们在 X 中通常是闭的。

例 6.1（垂直剖分） 第 6.2.2 节中垂直剖分是奇异复形的一个非常好的例子，该复形因为包含梯形而不是一个单纯复形。每个梯形和三角形的内部形成了一个 2-胞腔，它是一个开集。对每个邻近 2-胞腔对来说，在它们的公共边界上有一个 1-胞腔。没有 0-胞腔，因为顶点处在$\mathcal{C}_{obs}$ 中，而不是在$\mathcal{C}_{free}$ 中。所有 2-胞腔和 1-胞腔的并集产生子空间 $S_2=\mathcal{C}_{free}$。这满足闭包的要求，因为复形仅建立在$\mathcal{C}_{free}$ 中；因此拓扑空间是$\mathcal{C}_{free}$。集合 $S_2=\mathcal{C}_{free}$ 既是开的，也是闭的。集合 S_1 是所有 1-胞腔的并集。这也是闭的，因为 1-胞腔的端点全都落在$\mathcal{C}_{obs}$ 中。每个 1-胞腔也是一个开集。

避免从限制在$\mathcal{C}_{free}$ 的拓扑上产生某些奇怪结论的一种有效方法是在$\mathcal{C}_{free}$ 的闭包 $cl(\mathcal{C}_{free})$ 中建立垂直剖分。可以这样进行：从以前定义的垂直剖分开始，为$\mathcal{C}_{obs}$ 的每条边加入一个新的 1-胞腔，为$\mathcal{C}_{obs}$ 的每个顶点增加一个新的 0-胞腔。现在 $S_3=cl(\mathcal{C}_{free})$，它在 $\mathbb{R}^2$ 上是闭的。同样地，S_3、S_1 和 S_0 通常是闭的。单个 k-维胞腔在拓扑空间 S_k 中是开的。只有一种奇怪的情况是 0-胞腔被认为是开的，但是这仅在离散拓扑空间 S_0 中才如此。 ■

6.3.2　二维剖分

当$\mathcal{C}_{obs}$ 是多边形时，第 6.2.2 节中的垂直剖分只是解决运动规划问题的许多胞腔剖分方法中的一种。它在胞腔数、计算效率以及实现性之间进行了很好的平衡。通常有可能将$\mathcal{C}_{obs}$ 剖分成为数不多的几个凸胞腔。对多疑问问题来说这也是所希望的，因为路线图会更小。遗憾的是，很难优化胞腔的数量。用最小数量的凸胞腔来对有孔的多边形$\mathcal{C}_{obs}$ 进行剖分，复杂度是 NP 的[522,648]。因此我们宁愿忍受非最优的剖分。

三角剖分　除垂直剖分以外的另一种方法是三角剖分，它在$\mathcal{C}_{free}$ 上产生一个单纯复形。图 6.16 显示了一个例子。因为$\mathcal{C}_{free}$ 是一个开集，所以没有 0-胞腔。每个 2-单形（三角形）有一个、两个或者三个面，这取决于它与$\mathcal{C}_{obs}$ 分享的边界有多少。连接 1-胞腔和 2-胞腔的采样点就可以产生一个路线图，如图 6.17 所示。对于一个给定的问题，有许多方法对$\mathcal{C}_{free}$ 进行三角剖分。计算几何中相当多的注意力集中在寻找好的三角剖分[129,266,304]，例如试图避免细长的三角形。

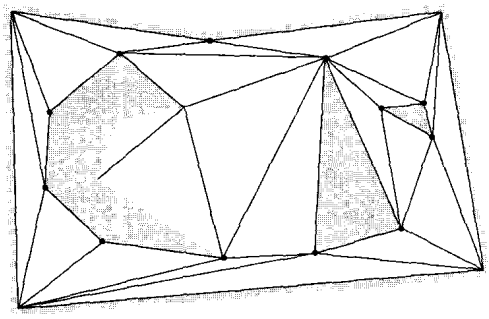


图 6.16 C_{free} 的一个三角剖分

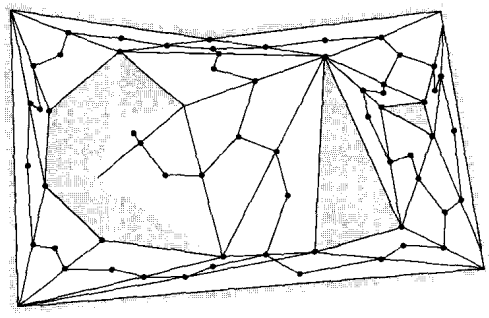


图 6.17 由三角剖分得到的一个路线图

如何计算三角剖分？这可以通过第 6.2.2 节的垂直剖分并将每个梯形剖分为两个三角形来进行。即使这产生了三角形胞腔，它也不会产生一个单纯复形（两个三角形可能在三角形边的同一侧）。一个自然的方法是通过用线段来连接一个面的两个顶点，从而逐步地对面进行划分。如果线段与其他线段不相交，那么划分就可以进行。在有多于三个顶点的面的所有顶点上迭代进行这种过程，直到最后得到三角剖分。遗憾的是，这导致了一个 $O(n^3)$ 时间算法，因为最坏情况下有 $O(n^2)$个对必须进行检验，每次检验需要 $O(n)$时间来确定是否与其他线段有交集。通过进行射线扫描，这很容易减小到 $O(n^2 \lg n)$。文献[266]的第 3 章提出的算法首先将C_{free} 分拆为单调多边形（monotone polygons），然后再对每个单调多边形进行三角剖分，其运行时间是 $O(n \lg n)$。如果C_{free} 是单连通的，那么三角剖分可以在线性时间内计算出来[192]。很遗憾，这种算法太复杂以至于不能在实际情况下使用（但是有更简单的算法，其复杂度接近于 $O(n)$；见文献[92]和文献[266]的第 3 章）。

圆柱剖分 圆柱剖分（cylindrical decomposition）与垂直剖分非常相似，除了当图 6.2 中的任何一种情况发生时有一条垂线从 $y=-\infty$ 到 $y=\infty$ 切开所有的面。结果可以认为是一个奇异复形，如图 6.18 所示。看上去这可能没有垂直剖分那样有效率；但是这里提出它是因为它可以推广到任意维数、任意 C 空间拓扑以及任意半代数模型。因此这里提出它是为了向更一般的剖分过渡。圆柱剖分的最重要的特性如图 6.19 所示。考虑两个事件之间的每个垂直带。当从 $y=-\infty$ 到 $y=\infty$ 经过这个带时，点在C_{obs} 和C_{free} 之间交替变化。例如在事件

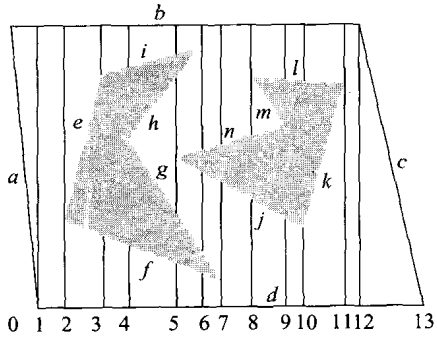


图 6.18 与图 6.6 进行比较；圆柱剖分与垂直剖分的不同之处在于射线继续向前，而不是停在最近的边

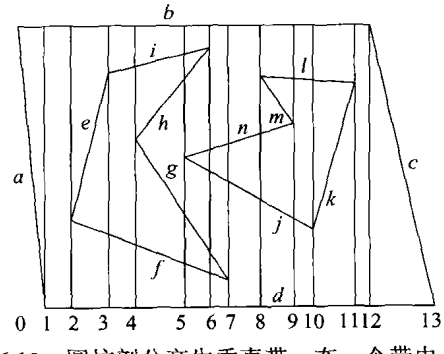


图 6.19 圆柱剖分产生垂直带；在一个带内，有被 C_{obs} 分割开的无碰撞胞腔的一个堆

4 和事件 5 之间，边 f 下方的点在 C_{free} 中，f 和 g 之间的点在 C_{obs} 中。g 和 h 之间的点在 C_{free} 中，如此重复。可以定义胞腔剖分使得在 C_{obs} 中也能够构造二维胞腔。令 $S(x, y)$表示第 3.1.1 节中的逻辑谓词（见式（3.6））。当经过一个带，$S(x, y)$的值也交替变化。这种行为是构造圆柱剖分的主要原因，这在第 6.4.2 节中会变得更加清晰。每个垂直带实际上可以看作是一个圆柱，因此这种剖分称为圆柱剖分（不一定有三维几何意义上的圆柱）。

6.3.3　三维垂直剖分

通过迭代地应用扫描的思路，第 6.2.2 节中的垂直剖分可以扩展到任何维数。但是要求 C_{obs} 是分段线性的。也就是说，C_{obs} 可以表示为半代数模型，该模型的所有本原都是线性的。很遗憾，由于旋转会产生非线性变换，因此大多数一般运动规划问题涉及非线性代数本原。回忆到第 4.3.3 节中构造的 C_{obs} 的复杂代数模型。为了处理一般代数模型，需要用到计算代数几何的内容。这将在第 6.4 节介绍。

如果 C_{obs} 是分段线性的，那么存在这样一个问题：能够在 $\mathbb{R}^3$ 中进行平移的多面体机器人和 $\mathcal{W}$ 中障碍物都是多面体。因为变换方程在这种情况下是线性的，所以 $C_{obs} \subset \mathbb{R}^3$ 是多面体。第 4.3.2 节所提及的 $\mathcal{A}$ 和 $\mathcal{O}$ 之间每种 FV 类、VF 类、EE 类接触形成几何本原，可以得到 C_{obs} 的多边形面。

图 6.20 阐述了构造三维垂直剖分的算法。将它与第 6.2.2 节的算法进行比较。令 (x, y, z) 表示 $C = \mathbb{R}^3$ 中的一个点。垂直剖分产生凸 3-胞腔、2-胞腔和 1-胞腔。不考虑退化情况，一个一般 3-胞腔以 6 个面为界。一个 3-胞腔对某固定 x 值的截面产生一个梯形或者三角形，这与二维情况相同，只是处在平行于 yz 平面的一个平面内。一个一般 3-胞腔的两侧平行于 yz 平面，其他两侧平行于 xz 平面。3-胞腔的上下以 C_{obs} 的两个多边形面为界。

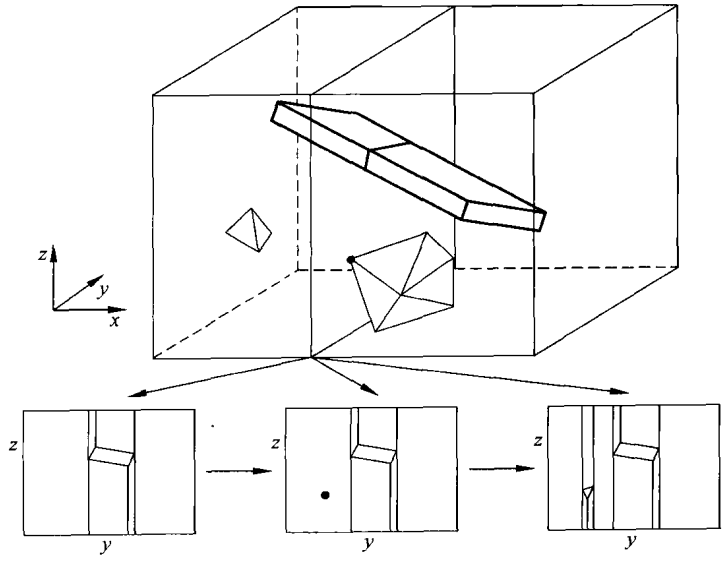


图 6.20　在高维情况下，可以递归地进行扫描

初始情况下，以 x 轴坐标对 C_{obs} 的顶点进行排序来得到事件。现在考虑扫描一个垂直于 x-轴的平面。一个 x 值固定的平面产生 C_{obs} 的一个二维多边形切片。图 6.20 的下部显示了三

个这种切片。每个切片平行于 yz 平面并且看起来像一个可以用二维垂直剖分方法来解决的问题。一个切片中的 2-胞腔实际是三维剖分中 3-胞腔的切片。这些 3-胞腔能够发生临界变化的地方只是当扫描平面停在某 x 值时。图 6.20 中的第二个切片对应的情形是遇到一个凸多面体的一个顶点，并且整个多面体都在扫描平面的右边(还没有遇到多面体的其他部分)。这对应于切片中临界变化必须发生的一个地方。它们是图 6.2 中情形的三维版本，这指出垂直剖分需要如何更新。算法首先在第一个 x 事件建立二维垂直剖分。在每个事件，二维垂直剖分必须更新以考虑临界变化。在这过程中，三维胞腔剖分和路线图可以逐步进行构造，就像二维情形一样。

通过将一个采样点放置在每个 3-胞腔和 2-胞腔的中心来构造路线图。顶点是采样点，对于一个 3-胞腔邻近于一个 2-胞腔的情形，通过连接采样点来将边加到路线图中。

该原理可以扩展到任何维数，但是在运动规划中的应用依然很有限，因为该方法要求线性模型（或者至少很难应用于非线性模型；在某些特殊情况下可以这样做）。文献[429]概括了对各种几何本原和维数所进行的垂直剖分的复杂度。

6.3.4 线段机器人的剖分

本节对非线性模型提出一种最简单的胞腔剖分方法，当然它也相当复杂。这有助于评估一般情况下组合运动规划的复杂度。考虑图 6.21 所示的规划问题。机器人$\mathcal{A}$是一个线段，它可以在 $\mathcal{W}=\mathbb{R}^2$ 中进行平移和旋转。$\mathcal{A}$的一个端点上用作原点，该点不是模型的一部分。C-空间 C 与 $\mathbb{R}^2\times\mathbb{S}^1$ 是同胚的。假设用 $\mathbb{R}^2\times[0,2\pi]/\sim$进行参数表示，其中 θ=0 和 θ=2π 是等同的。$\mathcal{C}$中的一个点表示为(x,y,θ)。

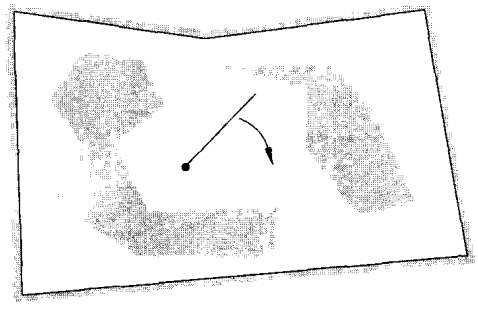


图 6.21 在二维世界中能够平移和旋转的一个线段的运动规划

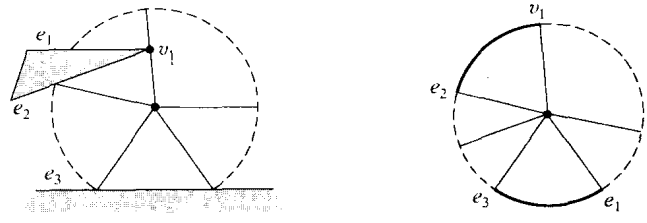
一个近似解 首先考虑对只能进行平移这种情况进行胞腔剖分。通过用 EV 类和 VE 类接触来处理机器人-障碍物相互作用，就可以使用第 4.3.2 节的方法来计算$\mathcal{C}_{obs}$。当$\mathcal{A}$的内部接触到一个障碍物顶点时，就得到 EV 类接触。$\mathcal{A}$的端点接触到一个物体内部时产生 VE 类接触。这两种情况都产生多边形$\mathcal{C}_{obs}$ 的一条边。一旦完成这些表示，那么就可以用垂直剖分来解决问题。这就产生了针对旋转情形的一种数值方法，该方法将 θ 离散化为 K 个值 $i\Delta\theta$，$0\leqslant i\leqslant K$，并且 $\Delta\theta=2\pi/K$[20]。在每种情况下障碍区域$\mathcal{C}_{obs}$都是多边形的，我们可以想象有 K 个多边形区域的一个堆。通过在切片内用通常的方法连接采样点，或者也可以通过连接相邻切片中相应胞腔之间的采样点，就可以形成一个路线图。如果 K 足够大，那么这

种策略非常好，但是这种方法并不完备，因为不能提前确定足够的 K 值。这种方法实际上是组合运动规划和基于采样的运动规划的一个有趣的混合体。也可以想象一个具有完备分辨率的版本。

在极限情况下，当 K 趋近无穷时，C_{obs} 的表面将沿着 θ 的方向弯曲。必须应用第 4.3.3 节的条件来生成实际的障碍区域。这是可能的，但是它产生的是一个采用隐含的多项式本原的一个半代数表示。而用可以在运动规划中使用的简单胞腔来确定一个显式表示并不是一个轻松的工作。不能使用第 6.3.3 节的方法，因为C_{obs}不是多面体的。因此需要特殊分析来产生一种胞腔剖分。

一般思路是在 $\mathbb{R}^2$ 中通过只考虑平移部分(x, y)来构造一个胞腔剖分。将 θ 作为 xy 平面“上方”的第三个轴，$\mathbb{R}^2$ 中的每个胞腔被提升到C中。一个圆柱剖分导致了 xy 平面中的每个胞腔为不同的 θ 值产生了胞腔的一个圆柱堆。回忆到图 6.18 和图 6.19 中的圆柱。垂直坐标轴对应于目前的 θ，水平坐标轴用 x 和 y 轴代替。

为了构造 $\mathbb{R}^2$ 中的剖分，考虑如图 6.22 所示的各种机器人-障碍物接触。在图 6.22(a)中，线段绕固定点(x, y)转动。这样会产生两种不同的接触。对某方位（θ 的值），线段接触到 v_1 后形成 VE 类接触。对于其他 3 个方位，线段接触到一条边，形成 VE 类接触。这里再次用到要素这个概念，在 4 个方位上线段接触到一个要素。每个要素可以是一个顶点或者一条边。在 e_2 和 e_3 之间，机器人是无碰撞的。这些位形落在C_{free}中。机器人处在 e_3（最右边的接触）和 v_1 之间的位形也在C_{free}中。所有其他方位产生C_{obs}中的位形。注意到线段不能从处在 e_2 和 e_3 之间的位形到达处在 e_3 和 v_1 之间的位形，除非(x, y)的位置发生变化。因此无论怎样进行剖分，这些位形都必须对应于不同的胞腔，这样做似乎是明智的。



(a)　(b)

图 6.22　固定(x, y)并旋转线段到所有角度$\theta \in [0,2\pi]$/～；(a) 注意被线段击中的顶点和边要素；(b) 记录机器人不发生碰撞的方位区间

雷达图　图 6.22(b)阐明哪个 θ 值产生碰撞。我们将这种表示称为雷达图（radar map）。4 种接触的方位由接触要素来指示。符号$[e_3, v_1]$和$[e_2, e_3]$确定$(x, y, \theta) \in C_{free}$的两个区间。现在假设稍微改变$(x, y)$得到$(x', y')$。那么雷达图如何改变？发生接触的精确角度将会改变，但是表示落在C_{free}中的位形的符号$[e_3, v_1]$和$[e_2, e_3]$保持不变。即使角度改变，在接触方面并没有令人感兴趣的变化；因此可以断言(x, y, θ)和(x, y, θ')处在C_{free}的相同胞腔中，因为 θ 和 θ'都将线段放置在相同两个接触之间。假设(x, y)周围一个小区域上方有两个 3-胞腔。一个 3-胞腔是为$[e_3, v_1]$中的方位，而另外一个是为$[e_2, e_3]$中的方位。这以C_{free}中的三维区域的形式出现，因为 x、y 和 θ 都可以在受到微小扰动的情况下不离开它的胞腔。

当然，如果(x, y)的变化足够大，那么最后我们预料在雷达图中发生一个显著的变化。例如想象 e_3 无限长，并且图 6.22(a)中 x 的值逐渐增加。图 6.22(b)中 v_1 和 e_2 之间的黑带的

长度会缩小。最后，当(x', y')到v_1的距离比$\mathcal{A}$的长度还长时，黑带就会消失，这种情形如图 6.23 所示。这种变化非常重要，因为那个区域消失以后，e_3和e_3之间的任何方位θ'绕着大圆弧转动产生一个位形$(x', y', \theta') \in \mathcal{C}_{free}$。这似乎非常重要，因为它告诉我们：机器人从$v_1$继续向前运动、旋转、然后向后运动，那么就可以在原始的两个胞腔之间运动。从图 6.23 所示位置向$+y$方向运动。剩下的黑带开始缩小，直至最后当与e_3的距离比机器人的长度还长时，黑带消失。这表示了另一种临界变化。

当有k个方位区间使得位形落在$\mathcal{C}_{free}$中时，那么雷达图可以通过环形排序来刻画：

$$([f_1, f_2], [f_3, f_4], [f_5, f_6], \cdots, [f_{2k-1}, f_{2k}]) \tag{6.6}$$

对于图 6.22(b)中的雷达图，这种表示产生$([e_3, v_1], [e_2, e_3])$。每个f_i就是一个要素，要素可以是一条边或者一个顶点。一些f_i可能是相同的；表示图 6.23(b)的是$([e_3, e_3])$。雷达图以逆时针的顺序指定区间。因为是环形的，无论指定谁是第一个区间都没有关系。这里有两种退化情形。如果对于所有$\theta \in [0, 2\pi)$，有$(x, y, \theta) \in \mathcal{C}_{free}$，那么顺序记为()。另一方面，如果对于所有$\theta \in [0, 2\pi)$，有$(x, y, \theta) \in \mathcal{C}_{obs}$，那么记为$\phi$。

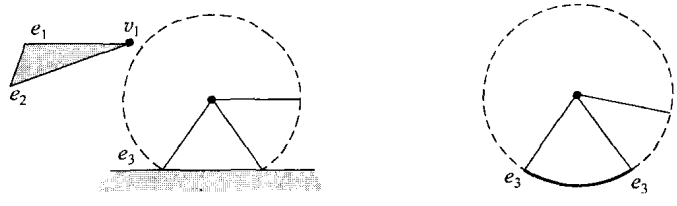


图 6.23 如果x增加到足够大，那么在雷达图中会出现一个临界变化，因为机器人不再可能到达v_1

胞腔中的临界变化 现在我们准备更加详细地解释胞腔剖分。假设沿着$\mathbb{R}^2$中的一条路径行进，并且产生图 6.22(b)中雷达图的一个动画。那么我们说每次当环形排序见（式(6.6)）变化时发生临界变化。当区间出现如下情况时发生临界变化：（1）出现；（2）消失；（3）分裂；（4）合并；（5）当区间的要素发生变化。首要任务是将$\mathbb{R}^2$分拆为不会发生临界变化的最大的 2-胞腔。这些 2-胞腔R都代表了$\mathcal{C}_{free}$中一个 3-胞腔带的投影。每个 3-胞腔的定义如下。令$\{R, [f_i, f_{i+1}]\}$表示$\mathcal{C}_{free}$中的三维区域，该区域中$(x, y) \in R$和θ将线段放置在接触点f_i和f_{i+1}之间。对于环形排序见（式（6.6））中的每个区间，$\{R, [f_i, f_{i+1}]\}$表示R上方的柱状胞腔。如果因为$\mathcal{A}$在R中从来没有接触到过障碍物而使得任何方位都是可能的，那么记为$\{R\}$。

在$\mathbb{R}^2$中的什么位置会引起临界变化发生？有 5 种不同的情况需要考虑，它们都在$\mathbb{R}^2$中产生临界曲线的一个集合。当越过这些曲线时就会发生临界变化。如果没有越过曲线，那么就不会发生临界变化。因此这些曲线精确地定义了在$\mathbb{R}^2$中想要的 2-胞腔的边界。令L表示机器人的长度（它是线段机器人）。

考虑上面所提到的五种情况是如何发生的。其中两种情况已经在图 6.22 和图 6.23 观察到。这些出现在图 6.24(a)和图 6.24(b)中，并且发生在(x, y)与一条边或一个顶点的距离小于L的时候。第三种和第四种情况分别如图 6.24(c)和图 6.24(d)所示。会发生第三种情况，因为曲线引起$\mathcal{A}$在能够接触e和能够接触v之间变化。该曲线从端点处的一条边延伸而来，该端点为反射顶点（内角大于π）。第四种情况实际上是图 6.10 的双切线情况，它产生短路

径图。如果顶点之间的距离小于 L，那么会生成一个线性临界曲线，因为当从右至左越过它时，$\mathcal{A}$不再能够接触到 v_2。双切线总是成对产生曲线；v_2 上方的曲线没有显示出来。图 6.25 所示的最后一种情况是最复杂的。它是一条 4 自由度代数曲线，称为 Nicomedes 螺旋线，它在$\mathcal{A}$同时接触 v 和 e 时出现。在曲线内，$\mathcal{A}$可以接触 e 但不能接触 v。仅在曲线的外面才可以接触到 v。如果 xy 坐标轴使得 v 在(0,0)，那么曲线方程为：

$$\left(x^2-y^2\right)\left(y+d\right)^2-y^2L^2=0 \tag{6.7}$$

其中 d 是从 v 到 e 的距离。

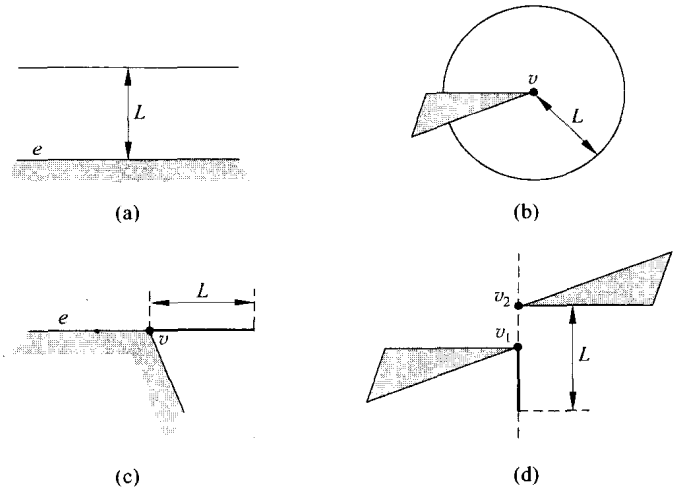


图 6.24　$\mathbb{R}^2$ 中五种产生临界曲线情况中的四种情况

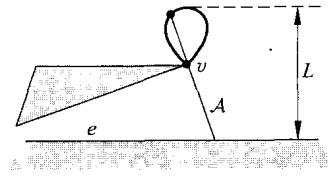


图 6.25　第五种情况是最复杂的；它产生一个被称为 Nicomede 螺旋线的四自由度代数曲线

将所有曲线放在一起生成 $\mathbb{R}^2$ 的一个胞腔剖分。存在非临界区域，在这些区域上式(6.6)没有发生变化；它们形成 2-胞腔。邻近 2-胞腔之间的边界是临界曲线的一部分，它们形成 1-胞腔。在临界曲线相交处也有 0-胞腔。图 6.26 显示了一个选自文献[591]的例子。这里没有绘制对应的位形都在C_{obs} 中的那些临界曲线，如果绘出这些曲线，方法相同，但是会产生不必要的胞腔边界。除了剖分C_{free} 以外，它们也可以对C_{obs} 进行胞腔剖分，当然这只是为了增加趣味性。因为C_{obs} 是要避开的，那么最好避免将时间浪费在对它进行剖分上。通过假设$\mathcal{A}$是一束作用范围为 L 的激光，那么这些不必要的情况就能够被检测到。当激光扫描时，只有那些被激光接触到的要素才是有意义的。激光的视线看不见的任何要素对应于不必要的边界。

在 $\mathbb{R}^2$ 中构造胞腔剖分以后，需要将它提升到 $\mathbb{R}^2\times[0,2\pi]/\sim$。这在每个二维非临界区域 R 以上生成一个柱状 3-胞腔。容易定义路线图使得每个 3-胞腔和 2-胞腔都有一个顶点，它们与以前的胞腔剖分方法相同；但是为了使下面的例子更容易理解，这里没有生成 2-胞

腔上的顶点。每个 3-胞腔$\{R, [f_i, f_{i+1}]\}$对应于路线图中的顶点。路线图的边连接相邻 3-胞腔，这些 3-胞腔将一个 2-胞腔作为它们公共边界的一部分。这意味着在$\mathbb{R}^2$中它们分享一条一维临界曲线。

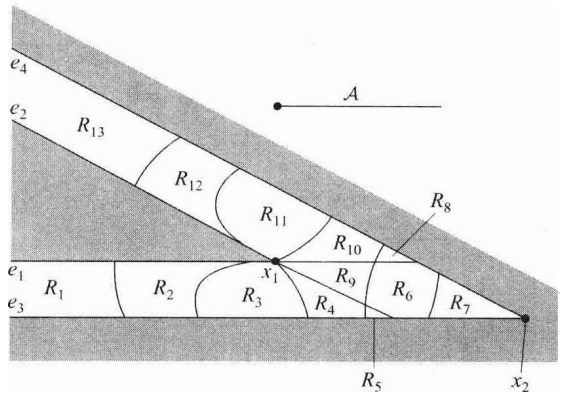


图 6.26 临界曲线形成$\mathbb{R}^2$中非临界区域的边界

构造路线图 问题是确定哪些 3-胞腔是相邻近的。图 6.27 刻画了需要进行连接的情况。xy 平面表示成一个轴（假设沿着平行于它的方向看）。考虑平面内的两个邻近 2-胞腔（非临界区域）R 和 R'。假设$\mathbb{R}^2$中的一个 1-胞腔（临界曲线）将它们分开。任务是连接 R 和 R'上方的圆柱中的 3-胞腔。如果邻近胞腔享有相同要素对，那么将它们连接起来。这意味着必须连接$\{R, [f_i, f_{i+1}]\}$和$\{R', [f_i, f_{i+1}]\}$。在某些情况下，一个要素可能发生变化，而方位的区间保持不变。例如当机器人从接触一条边变化到接触边的一个顶点时，就可能发生这种情况。在这些情况下也必须进行连接。图 6.27 阐述的一种情况是当方位区间发生分割或者合并的时候。从 R 到 R'，图中显示了两个区域合并成一个区域。在这种情况下，从原始的两个 3-胞腔必须连接到合并后的 3-胞腔。当构造路线图的边时，应该使用 3-胞腔和 2-胞腔的采样点来确保得到无碰撞路径，就像第 6.2.2 节中垂直剖分的情况一样。图 6.28 刻画了图 6.26 中例子的胞腔。每个非临界区域的上方有 1～3 个胞腔。每个胞腔由一个缩短后的机器人来指示，该机器人指向胞腔的一般方向。胞腔之间的连接也在图中显示出来。使用图 6.26 的非临界区域和要素名字，那么所产生的路线图会抽象地刻画在图 6.29 中。每

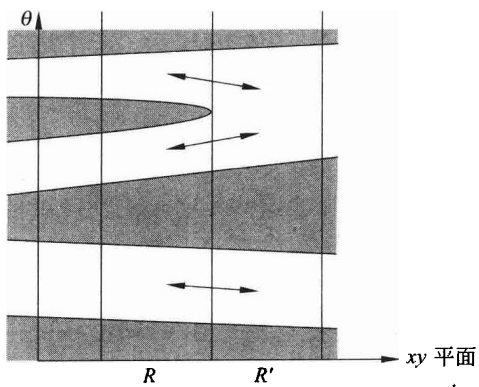


图 6.27 在相邻非临界区域上方的相邻 3-胞腔之间进行连接

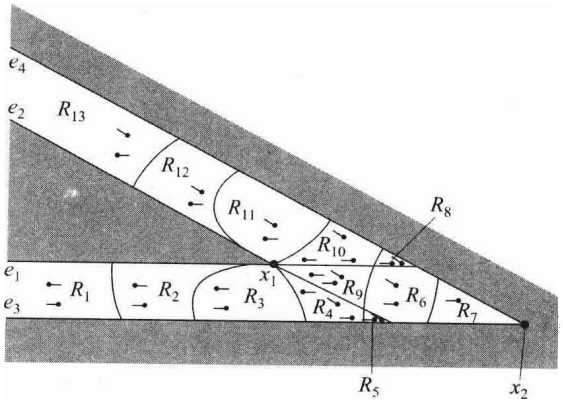


图 6.28　在非临界区域上方的 3-胞腔；显示了每个胞腔的用杆表示的采样点的方位（但是为清晰起见，缩短了杆的长度）；图 6.29 显示了胞腔之间的边

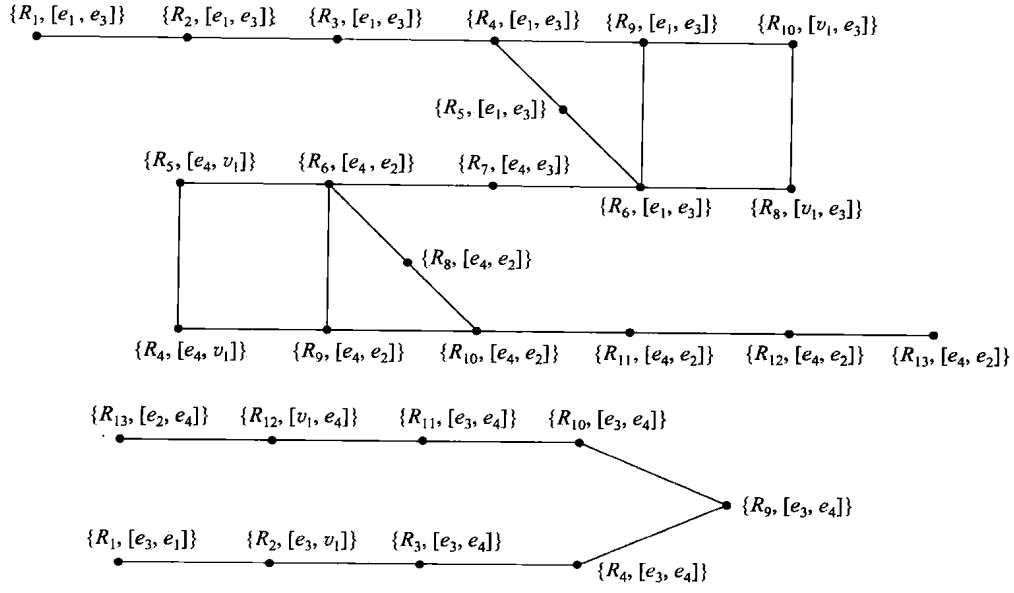


图 6.29　对应于图 6.26 中例子的路线图

个顶点代表C_{free}中的一个 3-胞腔，每条边代表邻近 3-胞腔相交形成的一个 2-胞腔。为了使路线图与以前的路线图相一致，我们给每条边都插入一个顶点，迫使路径经过相应 2-胞腔的采样点。

一旦构造了路线图，那么就可以用本章路线图方法来解决疑问。这里忽略了一些详细的实现过程。由于第五种情况，$\mathbb{R}^2$ 中的某些区域边界是 4 自由度代数曲线。避免明确描述每个非临界区域边界的方法以及其他实现的细节见文献[57]。文献[591]也对某些细节进行了总结。

复杂性　最坏情况下可能有多少胞腔？首先统计 $\mathbb{R}^2$ 中非临界区域的数量。有 $O(n)$种不同的方法来生成前 3 种类型的临界曲线，因为它们分别对应于一个单独的要素。遗憾的是，有 $O(n^2)$种不同的方法来生成双切线和 Nicomedes 螺旋线，因为它们基于要素对。假设没有

自相交，$\mathbb{R}^2$中 $O(n^2)$条曲线的族可能相交生成最多 $O(n^4)$个区域。$\mathbb{R}^2$中每个非临界区域的上方有 $O(n)$个 3-胞腔的一个圆柱。因此在最坏情况下胞腔剖分的大小是 $O(n^5)$。实际上，它们不可能都相交，所以可以认为胞腔的数量是在一个合理的范围内。文献[853]给出了构造胞腔剖分的一种 $O(n^5)$时间的算法。在第 6.5.3 中还提到更好运行时间的算法，但是它们过于复杂而难以被理解和实现。

6.4 计算代数几何

本节提出的算法很一般，虽然它们能够解决表述 4.1 的任何问题以及第 4.4 节中的封闭链问题。但是遗憾的是它们非常难以实现，并且并不足以解决大多数应用问题。本节的概念和工具大多是在计算实代数几何中得到发展的[79,252]。它们非常有力，足以解决机器人、计算机视觉、几何建模、计算机辅助设计以及几何定理证明中的大量问题。这些问题之一就是运动规划，最先认识到运动规划与计算代数几何之间联系的是文献[854]。

6.4.1 基本定义与概念

本节建立在第3.1节的半代数模型定义和第4.4.1节多项式定义的基础上。假设$C \subseteq \mathbb{R}^n$，以 2×2 或者 3×3 的矩阵形式表示每个 $SO(2)$或者 $SO(3)$时就会出现这种假设。例如一个三维刚体的情形，我们知道$C = \mathbb{R}^3 \times \mathbb{RP}^3$是一个六维流形，但是它可以嵌入$\mathbb{R}^{12}$，它由$\mathbb{R}^3$与所有 3×3 矩阵的集合的笛卡儿积得到的。迫使矩阵处在 $SO(2)$或者 $SO(3)$中的约束条件是多项式，它们因此可以加入到$\mathcal{C}_{obs}$和$\mathcal{C}_{free}$的半代数模型中。如果$\mathcal{C}$的维数小于 n，那么下面提出的算法就足够了，但是在表示上和复杂性上还存在一些问题，这些问题促使使用$\mathcal{C}$的特殊参数表示来使两者维数相同，同时将$\mathcal{C}$的拓扑变成$\mathbb{R}^n$的同胚。相关内容将在第 6.4.2 节简要讨论。

假设$\mathbb{R}^n$中的模型都用来自于$\mathbb{Q}[x_1,\cdots,x_n]$的多项式表示，其中$\mathbb{Q}[x_1,\cdots,x_n]$是有理数$\mathbb{Q}$的场上的多项式[1]集合。令$f \in \mathbb{Q}[x_1,\cdots,x_n]$表示一个多项式。

Tarski 命题 回忆第 3.1 节中的逻辑谓词。在这里再次使用它们，但现在它们的定义更加灵活。对于任何$f \in \mathbb{Q}[x_1,\cdots,x_n]$，一个原子（atom）是一个$f \bowtie 0$形式的表达，其中$\bowtie$是集合$\{=,\neq,<,>,\leqslant,\geqslant\}$中的某种关系。在第 3.1 节使用这些表达来定义逻辑谓词。这里我们假设使用除了$\leqslant$以外的关系，并且假设多项式变量的向量处在$\mathbb{R}^n$中。

一个无量词的表达式$\phi(x_1,\cdots,x_n)$是一个由原子和逻辑联结词“与”、“或”和“非”组成的逻辑谓词，逻辑联结词分别用$\wedge$、$\vee$和$\neg$表示。每个原子本身被认为是一个逻辑谓词，当且仅当在点$(x_1,\cdots,x_n) \in \mathbb{R}^n$计算多项式，关系得到满足的时候，它产生 TRUE。

例 6.2（一个谓词的例子） 令ϕ表示$\mathbb{R}^3$上的一个谓词，定义如下：

$$\phi(x_1,x_2,x_3) = \left(x_1^2 x_3 - x_2^4 < 0\right) \vee \left(\neg\left(3x^2x^3 \neq 0\right) \wedge \left(2x_3^2 - x_1x_2x_3 + 2 \geqslant 0\right)\right) \qquad (6.8)$$

联结词的优先级服从布尔代数法则。 ■

1 稍后将解释为什么$\mathbb{Q}[x_1,\cdots,x_n]$比$\mathbb{R}[x_1,\cdots,x_n]$更可取。

令一个量词Q是下列两个符号之一，$\forall$表示“对所有”，$\exists$表示“如果存在”。一个 Tarski 命题 Φ 是一个逻辑谓词，它可能还涉及关于一些或者所有变量的量词。一般情况下，一个 Tarski 命题具有如下形式：

$$\Phi(x_1,\cdots,x_{n-k})=(Qz_1)(Qz_2)\cdots(Qz_k)\phi(z_1,\cdots,z_k,x_1,\cdots,x_{n-k}) \tag{6.9}$$

其中 z_i 是一个用量词限定的变量（quantified variables），x_i 是自由变量（free variables），ϕ 是无量词表达式。一个有效 Tarski 命题中量词不一定都出现在左边；但是任何表达式都可以处理成一个等式使得所有量词在前面，如式(6.9)所示。将量词移动到前面的过程如下[708]：（1）去掉所有多余量词；（2）对一些变量重新命名，确保同一个变量不能既是自由的，又是有界的；（3）将取反符号尽可能向里移动；（4）将量词向左移。

例 6.3（几个 Tarski 命题） 没有自由变量的 Tarski 命题一般为 TRUE 或者 FALSE，因为没有结果所依赖的自变量。命题：

$$\Phi=\forall x\exists y\ \left(x^2-y<0\right) \tag{6.10}$$

为 TRUE，因为对于任何 $x\in\mathbb{R}$，总能够选择某 $y\in\mathbb{R}$ 使得 $y>x^2$。在式（6.9）的符号中，本例变成 $Qz_1=\forall x$，$Qz_2=\exists y$，$\phi(z_1,z_2)=\left(x^2-y<0\right)$。

交换量词的顺序就产生了：

$$\Phi=\exists y\forall x\left(x^2-y<0\right) \tag{6.11}$$

它为 FALSE，因为对于任何 y，总有一个 x 使得 $x^2>y$。

现在考虑一个有多个自由变量的 Tarski 命题：

$$\Phi(z)=\exists y\forall x\ \left(x^2-zx^2-y<0\right) \tag{6.12}$$

这产生一个函数：$\Phi:\mathbb{R}\to\{\text{TRUE},\text{FALSE}\}$，其中：

$$\Phi(z)=\begin{cases}\text{TRUE} & z\geqslant 1\\ \text{FALSE} & z<1\end{cases} \tag{6.13}$$

一个等价的无量词表达式 ϕ 可以定义为 $\phi(z)=(z>1)$，它与式（6.12）中的 Tarski 命题采用相同的真实值。你可能想知道是否总可能进行去掉量词的简化。这个问题称为量词消除问题（quantifier-elimination problem），在稍后将予以解释。 ■

决策问题　式（6.10）和式（6.11）中的命题导致了一个有趣的问题。考虑所有无自由变量的 Tarski 命题的集合。那些为 TRUE 的子集由实数的一阶理论组成。能不能得到一种算法来确定这样的命题是否为真？这是实数的一阶理论中的决策问题（decision problem）。乍看起来似乎没有希望得到这种算法，因为 $\mathbb{R}^n$ 是不可数无限的，而算法必须针对有限集。这是运动规划始终需要面临的一个问题。第 5 章的基于采样的方法为此提供了一种解决方案。该思路可以应用于决策问题，但是同样也存在缺乏完备性的问题。在 $\mathbb{R}^n$ 中不可能采样所有可能的点。取而代之的是构造一个组合表示，通过将 $\mathbb{R}^n$ 分拆为区域的一个有限族来精确描述决策问题，这样就可以解决决策问题。在每个区域内，仅有一个点需要检验。这应该与运动规划中的胞腔剖分有联系；那么解决决策问题的方法也能够解决运动规划问题。

量词消除问题　另外一个重要的问题如例 6.2 所示。考虑以式（6.9）的形式表示的所有 Tarski 命题的集合，它们可能有自由变量，也可能没有自由变量。是否能够得到一种算法来为 Tarski 命题 Φ 产生一个等价的无量词表达式 ϕ？令 $x_1,\cdots,x_n$ 表示自由变量。等价是指它们在自由变量 $(x_1,\cdots,x_n)$ 的集合上取得相同的值。

给定一个 Tarski 命题式（6.9），量词消除问题是对于所有 $(x_1,\cdots,x_n)\in\mathbb{R}^n$，寻找一个无量词的表达式 ϕ 使得：

$$\Phi(x_1,\cdots,x_n)=\phi(x_1,\cdots,x_n) \tag{6.14}$$

这等价于构造一个半代数模型，因为 ϕ 总能够以如下形式进行表示：

$$\phi(x_1,\cdots,x_n)=\bigvee_{i=1}^{k}\bigwedge_{j=1}^{m_i}\left(f_{i,j}(x_1,\cdots,x_n)\bowtie\right) \tag{6.15}$$

其中$\bowtie$可能是<、= 者>。这看起来与式（3.6）相同，区别在于式（6.15）使用<、= 或者>以允许开的、闭的半代数集合，而式（3.6）仅使用$\leqslant$为$\mathcal{A}$和$\mathcal{O}$构造闭的半代数集合。

问题仍然定义在 $\mathbb{R}^n$ 上，虽然 $\mathbb{R}^n$ 是不可数无限的，但是算法必须针对一个有限表示。这将用第 6.4.2 节中的胞腔剖分来完成。

半代数剖分 就像第 6.3.1 节所叙述的那样，在一个复形的每个胞腔内进行运动规划是很简单的。为了解决决策问题和量词消除问题，进行胞腔剖分使得在每个胞腔中这些问题变得更简单。所设计的胞腔剖分使得每个胞腔仅有一个点需要进行检验。

用式（6.15）表示的半代数集合 $Y\subseteq\mathbb{R}^n$ 为：

$$Y=\bigcup_{i=1}^{k}\bigcap_{j=1}^{m_i}\left\{(x_1,\cdots,x_n)\in\mathbb{R}^n\,\middle|\,\mathrm{sgn}\left(f_{i,j}(x_1,\cdots,x_n)\right)=s_{i,j}\right\} \tag{6.16}$$

其中 sgn 是符号函数，每个 $s_{i,j}\in\{-1,0,1\}$。集合论与第 3.1 节中的逻辑之间有良好的对应关系。通过将$\cup$和$\cap$分别变成$\vee$和$\wedge$，我们就将一个集合论描述转换为一个逻辑谓词。

令 $\mathcal{F}$ 表示式（6.16）中的 $m=\sum_{i=1}^{k}m_i$ 个多项式的集合。$\mathcal{F}$的符号分配是一个向量-值函数 $\mathrm{sgn}_{\mathcal{F}}:\mathbb{R}^n\to\{-1,0,1\}^m$。每个 $f\in\mathcal{F}$ 在符号分配向量中有一个相应的位置。在这个位置上出现符号 $\mathrm{sgn}\left(f(x_1,\cdots,x_n)\right)\in\{-1,0,1\}$。半代数剖分（semi-algebraic decomposition）是将 $\mathbb{R}^n$ 分拆为有限的连通区域的一个集合，每个区域的符号是不变的。这意味着在每个区域内部，$\mathrm{sgn}_{\mathcal{F}}$ 必须保持固定。这些区域不称为胞腔，因为一个半代数剖分不一定是第 6.3.1 节中所定义的奇异复形；区域可能含有孔。

例 6.4（符号分配） 回忆例 3.1 和第 3.1.2 节的图 3.4。图 3.4(a)显示了对只有一个多项式 $\mathcal{F}=\{x^2+y^2-4\}$ 的情况所进行的一个符号分配。符号分配定义为：

$$\mathrm{sgn}_{\mathcal{F}}(x,y)=\begin{cases}-1 & (x^2+y^2-4<0)\\ 0 & (x^2+y^2-4=0)\\ 1 & (x^2+y^2-4>0)\end{cases} \tag{6.17}$$

现在考虑为图 3.4(b)的姜饼面所进行的符号分配 $\mathrm{sgn}_{\mathcal{F}}$，如图 6.30 所示。半代数模型的多项式是$\mathcal{F}=\{f_1,f_2,f_3,f_4\}$，如例 3.1 所定义。依次有“头”、“左眼”、“右眼”、“口”。符号分配产生一个 4 维符号向量。如果(x,y)使$\mathcal{F}$中的一个多项式为 0，那么符号分配为 0。如果两条或者两条以上多项式的曲线相交，那么符号分配在交点上产生多个 0。

对图 6.30 中的姜饼面进行半代数剖分，有 9 个区域。5 个二维区域对应于：（1）姜饼面的外部；（2）左眼的内部；（3）右眼的内部；（4）口的内部；（5）姜饼面的内部、口和眼的外部。由 4 个一维区域，每个一维区域对应的点处在多项式的一个零集合上。所产生的剖分不是一个奇异复形，因为(−1,1,1,1)区域包含三个孔。 ■

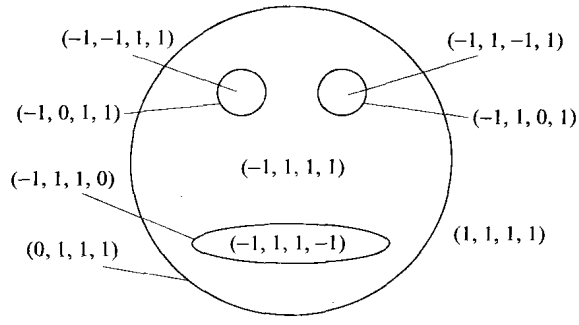


图 6.30　姜饼面的一个半代数剖分产生 9 个符号不变的区域

图 6.30 这样的剖分对于运动规划来说不是很有用，因为区域中有孔。对运动规划来说需要进一步的细分，这可以由圆柱代数剖分来产生。另一方面，任何半代数剖分对解决决策问题来说都是非常有用的。在每个区域内仅需要检验一个点就可以确定没有自由变量的某 Tarski 命题是否为真。为什么？如果多项式符号在某区域不能改变，那么相应逻辑谓词 Φ 的 TRUE/FALSE 值就不会改变。因此对每个符号不变区域仅检验一个点就足够了。

6.4.2　圆柱代数剖分

圆柱代数剖分是产生圆柱剖分的一般方法，其意义与第 6.3.2 节对 $\mathbb{R}^2$ 中多边形的剖分以及第 6.3.4 节线段形机器人的剖分相同。后来它也被称为 Collins 剖分[40,234,235]。图 6.19 所示的剖分甚至可以认为是对几何本原都是线性多项式的半代数集合所进行的圆柱代数剖分。在本节，这种剖分被推广到 $\mathbb{R}^n$ 中任意半代数集合。

思路是得到一个投影序列，它们每次将半代数集合的维数降低一维。初始情况下，集合定义在 $\mathbb{R}^n$ 上，每次投影后得到 $\mathbb{R}^{n-1}$ 中的一个半代数集合。最后投影到达 $\mathbb{R}$，得到一个单变量多项式，多项式的零值处在需要形成胞腔边界的临界位置上。通过分拆 $\mathbb{R}$ 就可以形成 1-胞腔（区间）和 0-胞腔的一个胞腔剖分。然后翻转顺序，形成从 $\mathbb{R}^2$ 到 $\mathbb{R}^n$ 的剖分。每次迭代从 $\mathbb{R}^i$ 的胞腔剖分开始，将它提升得到 $\mathbb{R}^{i+1}$ 中胞腔的一个圆柱。图 6.35 显示了姜饼面的剖分情况；因为 n=2，它仅涉及一个投影和一次提升。

半代数投影是半代数的　Tarski-Seidenberg 定理[79]有如下含义：

一个半代数集合的维数从 n 到 n−1 的一个投影是一个半代数集合。

所有半代数集合在投影下是封闭的，这确保 $\mathbb{R}^i$ 中的一个半代数集合的每个投影都会产生 $\mathbb{R}^{i+1}$ 中的一个半代数集合。在实际情况下这种性质对第 4.4.1 节介绍的（实）代数簇来说是不正确的。簇的定义只用到“＝”关系，在投影运算下它不是封闭的。因此使用半代数集合是一件非常好的事情，而不仅仅是一种巧合。

实代数数　就像以前叙述的那样，投影序列以 $\mathbb{R}$ 上的一个单变量多项式结束。胞腔的侧的定义基于多项式的根的准确定位。并且对 $\mathbb{R}^n$ 的一个复形中一个 k 维胞腔的采样点进行表示要求足够的精度。如果坐标稍微偏离，那么该点可能落入不同的胞腔。这产生了在计算机中如何表示和处理这些根的问题。

对于 4 阶或者小于 4 阶的单变量多项式，有用平方根和更高阶根的函数来计算所有根的公式。根据 Galois 理论[472,772]可以知道：对于大多数更高阶的多项式来说，这种公式

和根的表达式是不存在的，而这种多项式经常出现在运动规划的复杂半代数模型中。$\mathbb{R}$ 中的根可以是任何实数，一些实数需要用无穷表示。

避免这种混乱的一种方法是假设只使用 $\mathbb{Q}[x_1,\cdots,x_n]$ 中的多项式，而不是更一般的 $\mathbb{R}[x_1,\cdots,x_n]$。域 $\mathbb{Q}$ 不是代数封闭的，因为多项式的零值在 $\mathbb{Q}^n$ 之外。例如，如果 $f(x_1)=x_1^2-2$，那么 $f=0$ 对应 $x_1=\pm\sqrt{2}$，而 $\sqrt{2}\notin\mathbb{Q}$。但是 $\mathbb{R}$ 的一些元素从来就不是 $\mathbb{Q}[x_1,\cdots,x_n]$ 中的多项式的根。

$\mathbb{Q}[x]$ 中所有多项式的所有根的集合 $\mathbb{A}$ 被称为实代数数的集合。集合 $\mathbb{A}\subset\mathbb{R}$ 实际上表示了一个域（回忆第 4.4.1 节）。$\mathbb{A}$ 中的这些数有几个代数特征：（1）它们都有有限表示；（2）$\mathbb{A}$ 的元素的加乘运算可以在多项式时间内计算出来；（3）实代数数的不同表示之间的转换可以在多项式时间内完成。这意味着所有运算都可以有效地计算出来，而不需要借助于某种数值近似。在一些应用中，这种近似很好；但是对于代数剖分，这些近似因为对根可能产生混乱而损坏了临界信息（例如我们怎样知道一个多项式有多个离得很近的根或者只有两个离得很近的根？）。

这里没有给出详细内容，但是有几种方法表示实代数数以及相应的有效地对这些数进行运算的算法。圆柱代数剖分的运行时间本质上依赖于它的表示。实际上，求数值解的方法都有一个精度参数 ϵ，选择该参数足够小来保证根不会混乱。应用间隙定理（gap theorems）可以确定一个足够小的值，定理给出用多项式系数表达的实数根之间间隔的下界[175]。一些方法不要求精度参数。一个著名的例子是产生基于给定多项式的一个 Sturm 多项式序列。Sturm 序列中的多项式用来发现每个根的孤立区间[79]。多项式及其孤立区间可以认为是一个根的精确表达。采用这种表示所进行的代数运算可以在 $O(d\lg^2 d)$ 时间内完成，其中 d 是多项式的阶数[854]。关于实代数数的详细表示和计算，见文献[79,175,854]。

一维剖分 为了解释圆柱代数剖分方法，我们首先进行 $\mathbb{R}$ 的半代数剖分，这是投影序列的第一步。介绍一维剖分后，多维剖分的介绍就很容易了。

令 $\mathcal{F}$ 是 m 个单变量多项式的集合：

$$\mathcal{F}=\{f_i\in\mathbb{Q}[x]\,|\,i=1,\cdots,m\} \tag{6.18}$$

用它来定义 $\mathbb{R}$ 中的某半代数集合。$\mathcal{F}$ 中的多项式直接来自于无量词表达式 ϕ（它出现在 Tarski 命题中，就像式（6.9）那样）。

定义一个多项式为 $f=\Pi_{i=1}^m f_i$。假设 f 有 k 个不同的实数根，并以递增的顺序排列：

$$-\infty<\beta_1<\beta_2<\cdots<\beta_{i-1}<\beta_i<\beta_{i+1}<\cdots<\beta_k<\infty \tag{6.19}$$

一维半代数剖分通过如下 1-胞腔和 0-胞腔交替序列来给出：

$$(-\infty,\beta_1),[\beta_1,\beta_1],(\beta_1,\beta_2),\cdots,(\beta_{i-1},\beta_i),[\beta_i,\beta_i],(\beta_i,\beta_{i+1}),\cdots,[\beta_k,\beta_k],(\beta_k,\infty) \tag{6.20}$$

任何能够用 $\mathcal{F}$ 中的多项式进行表达的半代数集合也能够表示为式（6.20）中给出的某些 0-胞腔和 1-胞腔的并集。这也可以认为是一个奇异复形（它甚至可以认为是一个单纯复形，但是这不能扩展到更高维数）。

为每个胞腔生成采样点的方法如下。对于无界胞腔 $[-\infty,\beta_1)$ 和 $(\beta_k,\infty]$，有效的采样点分别是 β_1-1 和 β_k+1。对于每个有限 1-胞腔 (β_i,β_{i+1})，中间点 $(\beta_i,\beta_{i+1})/2$ 产生一个采样点。对于每个 0-胞腔 $[\beta_i,\beta_i]$，将 β_i 作为采样点。

例 6.5（一维剖分） 图 6.31 显示了 $\mathbb{R}$ 的一个半代数子集，它用两个多项式进行定义：

$f_1(x)=x^2-2x$ 和 $f_2(x)=x^2-4x+3$。这里 $\mathcal{F}=\{f_1,f_2\}$。考虑无量词表达式：

$$\phi(x)=\left(x^2-2x\geqslant 0\right)\wedge\left(x^2-4x+3\geqslant 0\right) \tag{6.21}$$

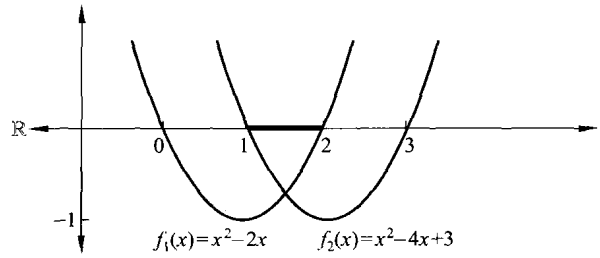


图 6.31　用两条抛物线来定义半代数集合[1,2]

图 6.32 显示了半代数剖分为 5 个 1-胞腔和 4 个 0-胞腔。每个胞腔都是符号不变的。1- 胞腔的采样点分别为–1,1/2,3/2,5/2,4。0-胞腔的采样点分别为 0,1,2,3。

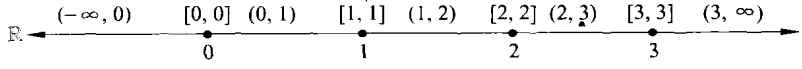


图 6.32　图 6.31 中多项式的一个半代数剖分

决策问题用剖分方法可以得到很好的解决。假设已经给定一个使用$\mathcal{F}$中的多项式的 Tarski 命题。例如：

$$\Phi=\exists x\left[\left(x^2-2x\geqslant 0\right)\wedge\left(x^2-4x+3\geqslant 0\right)\right] \tag{6.22}$$

采样点足以确定 Φ 是 TRUE 还是 FALSE。尝试 x =1，就会发现 Φ 是 TRUE。不能考虑量词消除问题，因为需要更多维数。■

通向更高维数的归纳法　现在考虑为 $\mathbb{R}^n$ 构造一个圆柱代数剖分（剖分实际上是半代数的）。图 6.33 显示了 $\mathbb{R}^2$ 情况下的一个例子。首先考虑如何迭代的将多项式投影到 $\mathbb{R}$ 来确保

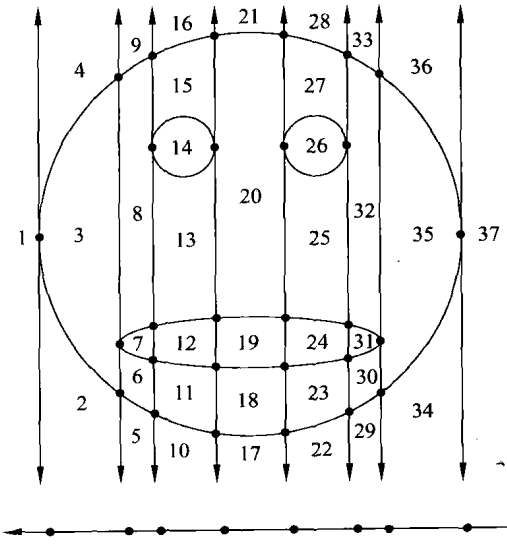


图 6.33　姜饼面的一个圆柱代数剖分；有 37 个 2-胞腔、64 个 1-胞腔和 28 个 0-胞腔；直的 1-胞腔是垂线的区间，弯曲的则是 $\mathcal{F}$ 中一个多项式的零集合的一部分；图中也显示了对 $\mathbb{R}$ 的剖分

当构造 $\mathbb{R}^n$ 的剖分时保持符号不变性。所产生的剖分对应于一个奇异复形。

有两种情况形成胞腔边界，如图 6.34 所示。令 $\mathcal{F}_n$ 表示 $\mathbb{Q}[x_1,\cdots,x_n]$ 中的多项式的原始集合，它们用来定义 $\mathbb{R}^n$ 中的半代数集合（或 Tarski 命题）。形成单个多项式 $f=\Pi_{i=1}^{m} f_i$。令 $f'=\partial f/\partial x_n$，这也是一个多项式。令 $g=GCD(f,f')$ 是 f 和 f' 的最大公约数。g 的零集合是使 f 和 f' 为零的所有点的集合。f' 为零意味着当扰动 x_n 时 $f=0$ 给出的曲面不会发生局部改变。这些是需要形成胞腔边界的地方，因为曲面可能在 x_n 方向上折叠，这对于圆柱剖分来说是不允许的。另外需要形成一个胞腔边界的地方是 $\mathcal{F}_n$ 中两个或者两个以上多项式的交点处。从 $\mathbb{R}^n$ 到 $\mathbb{R}^{n-1}$ 的投影生成 $\mathbb{Q}[x_1,\cdots,x_n]$ 中满足这些要求的多项式集合 $\mathcal{F}_{n-1}$。多项式 $\mathcal{F}_{n-1}$，至少有一个多项式包含一个零点，该零点比使 $f(x)=0$ 和 $f'(x)=0$、或 $\mathcal{F}_n$ 中多项式相交的每个点 $x\in\mathbb{R}^n$ 低一级。构造 $\mathcal{F}_{n-1}$ 的投影方法涉及计算主子结式系数，对这些系数的介绍见文献[79,855]。子结式是结式的一种扩展，对结式的介绍见文献[252]。

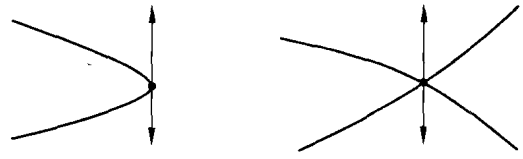


图 6.34 当曲面在垂直方向上进行折叠或者曲面相交时会产生临界点

$\mathcal{F}_{n-1}$ 中的多项式投影到 $\mathbb{R}^{n-2}$ 得到 $\mathcal{F}_{n-2}$。这个过程一直进行直到得到 $\mathcal{F}_1$，$\mathcal{F}_1$ 是 $\mathbb{Q}[x_1]$ 中的多项式的一个集合。就像以前定义的那样形成一个一维剖分。从 $\mathcal{F}_1$，通过乘法形成一个多项式，并且 $\mathbb{R}$ 被分拆为 0-胞腔和 1-胞腔。我们接下来描述将剖分从 $\mathbb{R}^{i-1}$ 提升到 $\mathbb{R}^i$ 的过程。这个过程迭代进行直到到达 $\mathbb{R}^n$。

假设已经为 $\mathbb{Q}[x_1,\cdots,x_{i-1}]$ 中的多项式 $\mathcal{F}_{n-1}$ 的一个集合进行了一个圆柱代数剖分计算。剖分由 k-胞腔组成，$0\leqslant k\leqslant i$。令 $p=(x_1,\cdots,x_{i-1})\in\mathbb{R}^{i-1}$。对于每个 k-胞腔 C_{i-1}，C_{i-1} 上的一个圆柱定义为(k+1)维集合：

$$\{(p,x_i)\in\mathbb{R}^i \mid p\in C_{i-1}\} \tag{6.23}$$

通过 $\mathcal{F}_i$ 中多项式将这个圆柱切成一个 k 维条带和多个(k+1)维胞腔。令 f_j 表示圆柱中 l 个用来进行切（片）的多项式中的一个，按照 x_i 递增的顺序进行排序 $f_1,f_2,\cdots,f_j,f_{j+1}\cdots,f_l$。产生如下类型的胞腔（见图 6.35）：

（1）无下界扇区：

$$\{(p,x_i)\in\mathbb{R}^i \mid p\in C_{i-1},x_i<f_1(p)\} \tag{6.24}$$

（2）截面：

$$\{(p,x_i)\in\mathbb{R}^i \mid p\in C_{i-1},x_i=f_j(p)\} \tag{6.25}$$

（3）有界扇区：

$$\{(p,x_i)\in\mathbb{R}^i \mid p\in C_{i-1},f_j(p)<x_i<f_{j+1}(p)\} \tag{6.26}$$

（4）无上界扇区：

$$\{(p,x_i)\in\mathbb{R}^i \mid p\in C_{i-1},f_l(p)<x_i\} \tag{6.27}$$

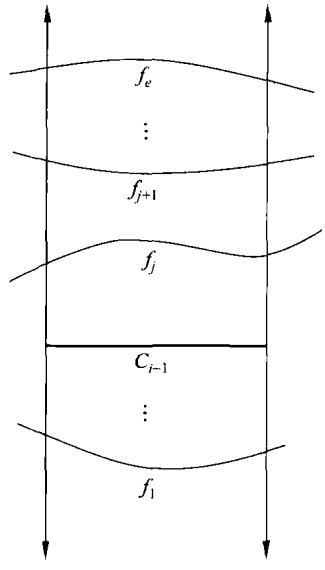


图 6.35　在每个 k-胞腔 C_{i-1} 上形成一个圆柱；多项式序列 $f_1,\cdots,f_i$ 将圆柱切成 k-维截面和$(k+1)$-维扇形

有一种退化的可能性：没有切片多项式，并且 C_{i-1} 上圆柱可以扩展成一个无界胞腔。一般情况下，在 $p\in C_{i-1}$ 中选取一个点并形成一个(p, x_i)形式的采样点的垂直柱，这样就可以计算采样点。采用一维剖分中的技术就可以生成 $\mathbb{Q}[x_i]$中的一个多项式，并放置采样点。

例 6.6（分拆姜饼面） 图 6.33 显示了姜饼面的一个圆柱代数剖分。所产生的复形与图 6.19 中所得到的非常相似。■

胞腔不必像更高维垂直剖分那样投影到一个矩形集合。例如 $\mathbb{R}^n$ 的一个剖分后的一个普通 n-胞腔 C_n 可以描述为 $x_1,\cdots,x_n\in\mathbb{R}^n$ 的开集合以使：

- 对于某些 0-胞腔 $C_0, C_0'\in\mathbb{R}$，有 $C_0 < x_n < C_0'$，其中 C_0、C_0' 是某 $f, f'\in\mathcal{F}_1$ 的根。
- 对于某些 1-胞腔 C_1 和 C_1'，(x_{n-1}, x_n)处在 C_1 和 C_1' 之间，其中 C_1、C_1' 是某 $f, f'\in\mathcal{F}_2$ 的零值。

⋮

- 对于某些 i-胞腔 C_{i-1} 和 C_{i-1}'，$(x_{n-i+1}, \cdots, x_n)$处在 C_{i-1} 和 C_{i-1}' 之间，其中 C_{i-1}、C_{i-1}' 是某 $f, f'\in\mathcal{F}_i$ 的零值。

⋮

- 对于某些$(n-1)$-胞腔 C_{n-1} 和 C_{n-1}'，$(x_1,\cdots,x_n)$处在 C_{n-1} 和 C_{n-1}' 之间，其中 C_{n-1}、C_{n-1}' 是某 $f, f'\in\mathcal{F}_n$ 的零值。

所产生的剖分是符号不变的，它使决策和量词消除问题在有限时间内得以解决。为了解决决策问题，在每个采样点对 $\mathcal{F}_n$ 中多项式进行计算以确定它们其中之一是否满足 Tarski 命题。为了解决量词消除问题，任何可以从 $\mathcal{F}_n$ 进行构造的半代数集合都可以定义为剖分中某些胞腔的一个并集。对于给定的 Tarski 命题，从命题提及的所有多项式来形成 $\mathcal{F}_n$，完成胞腔剖分。一旦得到，用符号信息来确定哪个胞腔需要包含在并集里。所产生的胞腔并集被设计成仅包括 $\mathbb{R}^n$ 中使 Tarski 命题为 TRUE 的点。

解决运动规划问题　圆柱代数剖分能够解决在第 4 章表述的任何运动规划问题。首先

假设 $C=\mathbb{R}^n$。像其他剖分一样，形成一个路线图，其中每个顶点是一个 n-胞腔，并且边穿过一个$(n-1)$-胞腔就能够连接每个邻近 n-胞腔对。确定圆柱内部邻近关系非常简单，但是对于确定来自于不同圆柱的胞腔的邻近关系问题，还有几个技术细节（文献[79]的第 152 页～第 154 页提出了一个阐述这个问题的例子）。维数低于 $n-1$ 的胞腔对运动规划来说是不需要的（就像对于第 6.2.2 节的垂直剖分来说顶点是不需要的一样）。根据疑问点 q_I 和 q_G 所处的胞腔将它们连接到相应路线图，进行离散搜索。

如果 $C\subset\mathbb{R}^n$，它的维数为 k，$k<n$，那么所有有趣胞腔都是更低维的。例如对矩阵进行约束迫使它们处在 $SO(2)$或者 $SO(3)$中，那么就会发生这种情况。对于第 4.4 节的闭链减少了自由度的问题，也会发生这种情况。圆柱代数剖分法仍然能够解决这种问题；但是当确定胞腔的邻近关系时，根的精确表示问题变得更加复杂。关于这些问题的讨论，见文献[854]。对于 $SO(2)$和 $SO(3)$情形，通过球极投影（stereographic projection）将 $\mathbb{S}^1$ 或者 $\mathbb{S}^3$ 分别映射到 $\mathbb{R}$ 或者 $\mathbb{R}^3$ 就可以避免这种复杂性。这种映射删除了一个点，但是$\mathcal{C}_{free}$ 的连通性没有受到破坏。用 $\mathbb{S}^3$ 表示的单位四元数的对径点等同也不存在问题；有$\mathcal{C}$的一个冗余副本，但是不影响连通性。

圆柱代数剖分的运行时间依赖于许多因素，但是运行时间一般与$\mathcal{F}_n$中多项式的数量、多项式最大代数阶数呈多项式关系，与维数呈两倍指数关系。关于详细的复杂性的论述见第 6.5.3 节。

6.4.3 Canny 的路线图算法

圆柱代数剖分的两倍指数关系运行时间促使研究人员研究更好的方法。已经表明：量词消除需要两倍指数时间[264]；但是运动规划是一个不同的问题。Canny 介绍了一种方法直接从半代数集合产生一个路线图，而不是沿着路线构造一个胞腔剖分。因为在圆柱代数剖分中有两倍指数多个胞腔，所以应该避免进行这种构造过程。那么所产生的 Canny 路线图方法就可以解决运动规划问题，运算时间与多项式的数量和阶数呈多项式关系，但是与维数呈单倍指数关系[172,175]；也可参见文献[79]。

像其他组合运动规划方法一样，该方法也基于发现临界曲线和临界点。主要思路是构造从 $\mathbb{R}^n$ 到 $\mathbb{R}^2$ 的线性映射，它产生半代数集合的轮廓曲线（silhouette curve）。完成原始半代数集合的这种映射就产生了一个路线图，但是它可能不保持原有的连通性。因此从 $\mathbb{R}^{n-1}$ 到 $\mathbb{R}^2$ 的线性映射在原始半代数集合中的某些$(n-1)$维切片上进行以产生更多路线图曲线。这个过程递归进行直到切片已经是一维的。那么在递归调用中得到的所有片段的并集就形成了路线图。所产生的路线图与原始的半代数集合有相同的连通性[175]。

假设 $C=\mathbb{R}^n$。令$\mathcal{F}=\{f_1,\cdots,f_m\}$表示定义半代数集合的多项式集合，假设它们是流形的一个没有交集的并集。假设每个 $f_i\in\mathbb{Q}[x_1,\cdots,x_n]$。首先，对输入多项式$\mathcal{F}$进行一个小的扰动来确保 $\mathbb{R}^n$ 中的每个符号不变集合是一个流形。通过随机扰动就可以依概率 1 迫使多项式进入一种最广位置；也可用确定性的方法来进行。对输入多项式以及二维投影方向的最广位置要求是相当强的，这促使最近人们通过更多的研究工作来解决这些问题[79]。基于这一点，那么可以假设多项式都处在最广位置。

回忆第 6.4.1 节的符号分配函数。因为最广位置的假设，每个符号不变集合是一个流形。

Canny 的方法为任意 k 维流形计算一个路线图，$k < n$。这样的流形有 $n-k$ 个符号为 0（这意味着点在$\mathcal{F}$中 $n-k$ 个多项式的零集合上）。至少有一个符号必须为 0，即 Canny 路线图实际上是在 ∂C_{free} 中（从技术上，这是不允许的，但是算法能够正确确定在C_{free}是否存在一条解路径）。

回忆到每个 f_i 都是一个函数 $f_i: \mathbb{R}^n \to \mathbb{R}$。令 x 表示$(x_1,\cdots,x_n)\in\mathbb{R}^n$。$k$ 个有零符号的多项式在一起产生一个映射 $\psi:\mathbb{R}^n \to \mathbb{R}^k$。向量 $\psi(x)$的第 i 个分量是 $\psi_i(x)=f_i(x)$。这与第 6.4.1 节中的符号分配函数有密切关系，除了在这里直接采用每个多项式的真实值，而不是多项式的符号以外。

现在引入一个函数 $g:\mathbb{R}^n \to \mathbb{R}^j$，其中 $j=1$ 或者 $j=2$（在下面提出的一般概念中 j 可以是其他值，但是对 Canny 方法来说只是 1 和 2）。函数 g 与圆柱代数剖分中投影的目的相同，但是 g 能够立刻从 n 维降到 2 维或者 1 维，而不是像圆柱投影那样维数降到 $n-1$。

令 $h:\mathbb{R}^n \to \mathbb{R}^{k+j}$ 表示如下从 ψ 和 g 直接构造的一个映射。对于第 i 个分量，如果 $i\leqslant k$，那么 $h_i(x)=\psi_i(x)=f_i(x)$。假设 $k+j\leqslant n$。如果 $i>k$，那么 $h_i(x)=g_{i-k}(x)$。令 $J_x(h)$表示 h 的 Jacobian 矩阵，它在 x 处定义为：

$$J_x(h)=\begin{pmatrix} \dfrac{\partial h_1(x)}{\partial x_1} & \cdots & \dfrac{\partial h_1(x)}{\partial x_n} \\ \vdots & & \vdots \\ \dfrac{\partial h_{m+k}(x)}{\partial x_1} & \cdots & \dfrac{\partial h_{m+k}(x)}{\partial x_n} \end{pmatrix}=\begin{pmatrix} \dfrac{\partial f_1(x)}{\partial x_1} & \cdots & \dfrac{\partial f_1(x)}{\partial x_n} \\ \vdots & & \vdots \\ \dfrac{\partial f_k(x)}{\partial x_1} & \cdots & \dfrac{\partial f_k(x)}{\partial x_n} \\ \dfrac{\partial g_1(x)}{\partial x_1} & \cdots & \dfrac{\partial g_1(x)}{\partial x_n} \\ \vdots & & \vdots \\ \dfrac{\partial g_j(x)}{\partial x_1} & & \dfrac{\partial g_j(x)}{\partial x_n} \end{pmatrix} \tag{6.28}$$

使 $J_x(h)$奇异的点 $x\in\mathbb{R}^n$ 被称为一个临界点。如果矩阵的每个$(m+k)\times(m+k)$阶子式为零，那么该矩阵被定义为奇异的。$J_x(h)$的前 k 行计算 $f_i(x)=0$ 的曲面法线。如果这些法线与最后 j 行给出的方向不是线性无关的，那么矩阵变成奇异的。下面的来自于文献[171]的例子很好地阐述了这个原理。

例 6.7（Canny 的路线图算法） 令 $n=3$，$k=1$，$j=1$。一个多项式 f_1 的零值定义了 $\mathbb{R}^3$ 的一个二维子集。令 f_1 是一个单位球 $\mathbb{S}^2$，它定义为如下多项式的零点：

$$f_1(x_1,x_2,x_3)=x_1^2+x_2^2+x_3^2-1 \tag{6.29}$$

假设 $g:\mathbb{R}^3 \to \mathbb{R}$ 定义为 $g(x_1,x_2,x_3)=x_1$。那么 Jacobian 矩阵（6.28）变成：

$$\begin{pmatrix} 2x_1 & 2x_2 & 2x_3 \\ 1 & 0 & 0 \end{pmatrix} \tag{6.30}$$

并且当可能的 3 个 2×2 子式都为零时，矩阵奇异。这种情况发生，当且仅当 $x_2=x_3=0$。这产生 $\mathbb{S}^2$ 上的临界点 $(-1,0,0)$ 和$(1,0,0)$。这也是 $\mathbb{S}^2$ 的法线平行于向量[1,0,0]的时候。

假设 j=2 以得到 $g:\mathbb{R}^3 \to \mathbb{R}^2$，并假设 $g(x_1,x_2,x_3)=(x_1,x_2)$。在这种情况下，Jacobian 矩阵（6.28）变为：

$$\begin{pmatrix} 2x_1 & 2x_2 & 2x_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{6.31}$$

当且仅当 $x_3=0$ 时矩阵奇异。因此临界点是与 $\mathbb{S}^3$ 相交的 x_1x_2 平面，这些临界点产生大圆上的点（所有使得 $x_1^2+x_2^2=1$ 的点 $(x_1,x_2)\in\mathbb{R}^2$）。在这种情况下会产生更多的点，因为对于 $\mathbb{S}^2$ 的任何平行于[1,0,0]、[0,1,0]或者它们的线性组合的曲面法线来说，矩阵发生了退化。 ■

例 6.7 中的第一个映射产生两个孤立的临界点，第二个映射产生临界点的一个一维集合，该集合称为一个轮廓。轮廓和孤立临界点的并集产生 $\mathbb{S}^2$ 的一条路线图。现在考虑将这个例子进行推广来得到对于一般 n 和 k 的完备算法。构造一个线性映射 $g:\mathbb{R}^n\to\mathbb{R}^2$，它可能不像例 6.7 那样沿轴向的，因为它必须选择在最广位置（否则在路线图中可能会发生退化）。定义 ψ 为多项式的集合，这些集合在构造路线图时所需的流形上为零。从 Jacobian 矩阵（6.28）确定轮廓。一般情况下可以用在圆柱代数剖分中也需要用到的子式来完成；参见文献[79,175]。令 g_1 表示 g 的第一个分量，它产生映射 $g_1:\mathbb{R}^n\to\mathbb{R}$。在 Jacobian 矩阵（6.28）中用 g_1 产生临界点的一个有限集合。临界点和轮廓的并集产生部分路线图。

但是到现在为止都不能保证连通性得到保护。为了处理这个问题，Canny 算法将递归进行。对于每个临界点 $x\in\mathbb{R}^n$，选择通过 x 的一个 $n-1$ 维超平面，对于该超平面来说，Jacobian 矩阵（6.28）的 g_1 行是正交的（从某种意义上，它垂直于 g_1 的流向）。在这个超平面里，形成一个新的 g 映射。这时选择一个新的方向，映射采用 $g:\mathbb{R}^{n-1}\to\mathbb{R}^2$ 的形式。将发现的轮廓和临界点都加入到路线图中。重复进行这个过程直到不用形成 g 就可以直接得到轮廓和临界点为止。

讨论一个例子。因为该方法涉及一个二维投影序列，所以它难以形象描述。在 $\mathbb{R}^4$ 或者更高维数的问题涉及两个或更多二维投影，因此也更加有趣。这里给出 $\mathbb{R}^3$ 上的一个例子，尽管它只有一个投影；$\mathbb{R}^3$ 上的另外一个例子参见文献[175]。

例 6.8（圆环面上的 Canny 算法） 考虑图 6.36 所示的三维代数集合，定义映射 $g(x_1,x_2,x_3)=(x_1x_2)$之后，得到如图 6.37 所示的路线图。从 g 得到轮廓，从 g_1（g 的第一个分量）得到临界点。实心圆环面的原始连通性没有得到保护，因为内圆环和外圆环没有相连。这说

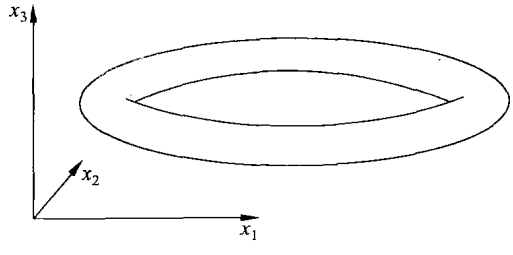


图 6.36 假设半代数集合是 $\mathbb{R}^3$ 中的一个实心圆环面

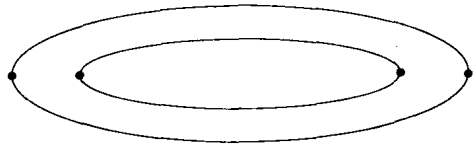


图 6.37 到 x_1x_2 平面的投影产生内环和外环的侧面轮廓以及四个临界点

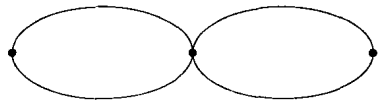


图 6.38　内临界点的一个切片平行于 x_2x_3 平面；切片的路线图连接到图 6.37 的侧面轮廓，因此保持了图 6.36 中原始集合的连通性

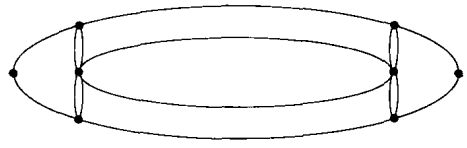


图 6.39　将所有的侧面轮廓和临界点进行合并得到路线图

明需要为更低维的切片进行路线图计算。对于 4 个临界点中的每个点来说，为这样的一个平面而计算临界曲线：它们平行于 x_2x_3，并且对于该平面来说 x_1 的位置由临界点来确定。其中一个内临界点的切片如图 6.38 所示。在这种情况下，切片有二维。新的轮廓曲线加入到路线图中得到如图 6.39 所示的最后结果。■

为了解决运动规划，在顶层递归中人为地将疑问点 q_I 和 q_G 指定为临界点。这迫使算法生成将它们连接到路线图其余部分的曲线。

算法的完备性需要仔细分析，这在文献[79,175]中有完整的论述。分析的主要组成部分为：（1）多项式可能被扰动，可以选择 g 来确保最广位置；（2）Jacobian 矩阵（6.28）的奇异性条件导致代数集合（簇）；（3）所产生的路线图具有第 6.1 节所提及的特性要求，即可达的和保护 $\mathcal{C}_{free}$ 的连通性（实际上是为 $\partial\,\mathcal{C}_{free}$）。上述方法为每个符号不变集计算路线图，但是为了得到规划问题的一个路线图，来自于每个符号不变集的路线图必须正确连接在一起；幸运的是，这通过文献[171]中的 Linking 引理已经得到解决。一个主要的问题是，在知道路线图的连通性之后，依然很难对路线图上每个曲线进行参数表示。因此以及其他技术原因，所以目前看来还没有 Canny 算法的一般实现。另外一个问题是要求 Whitney 分层（可以通过对输入的扰动来处理）。Basu-Pollack-Roy 路线图算法可以解决这个问题[79]。

6.5　运动规划的复杂性

本节对刻画运动规划复杂性的理论工作进行了概述。这并不等价于刻画特定算法的运行时间。一种算法的存在性用作问题复杂性的上界，因为实例证明：解决某具体问题要求的时间不多于算法所需要的时间。另一方面，下界也非常有用，因为它们指示了问题自身的复杂性。例如假设给出一个在 $O(n^2)$时间内解决问题的算法。那么试图寻找一个更有效的算法是否有意义呢？试图寻找一个一般意义的运动规划算法，它的运行时间与维数呈多项式关系，这是否有意义呢？下界提供了这些问题的答案。下界通常通过对奇异的、复杂的例子进行总结归纳而来，这些例子是问题的定义所允许的，但是往往不被第一个提出问题的人所考虑。沿着这种研究思路，通过下界（除非它已经是紧的）或者通过表明问题的一个更加严格的版本仍然能够允许这样奇异的例子，算法就可以不断完善提高。后一种情况在运动规划中经常出现。

6.5.1 下界

对各种运动规划问题和多种一般意义的规划问题来说已经建立了下界。为了阐述这些界限，需要对计算理论有一个基本的理解[465,892]。这个饶有兴趣的学科将在一些章节中进行概述。一个**问题**（problem）是被编码为二进制串的实例的集合。算法被认为是一个图灵机，即能够将比特位读和写到一个无界限的带子上的一个有限状态机。也存在计算的其他模型，例如整数 RAM 和实数 RAM（见文献[120]）；需要考虑的是选择哪个模型更合理，特别是当进行实数的几何计算时。从此往后都假设采用标准图灵机模型。算法通常采用二进制输出，这涉及**接受**（accepting）或者**拒绝**（rejecting）一个问题实例，实例初始时写到带子上并送给算法。在运动规划中，这等价于对给定的问题实例来说确定是否存在一条路径解。

机器代码 机器代码是与问题有关的二进制串的一个集合。它代表一个问题的实例的完整集合。如果在有限时间内算法正确接受所有属于它的串而拒绝其他串，那么我们认为算法决定了一个机器代码。有趣的是：决定机器代码需要多少时间和空间？假设用最好的算法来决定机器代码，在这样假设条件下，关于这个问题的疑问被提出来。我们可以很容易想到那些非常耗费资源的算法。

一个**复杂性类型**是一个可以在有特定资源限制情况下能够被决定的机器代码的集合。类型 P 是存在一个多项式时间算法的所有机器代码（和问题）的集合（例如对某整数 k 的算法时间是 $O(n^k)$）。从定义上讲，如果算法在多项式时间内决定了与它相关的机器代码[1]，那么认为它是有效率的。如果不存在有效率的算法，那么称问题是难处理的。在理论运动规划中出现的其他几种类型之间的关系如图 6.40 所示。NP 类型是能够用**非确定性的图灵机**在多项式时间内解决的机器代码的集合。对非确定性的图灵机的一些讨论出现在第 11.3.2 节。从直觉上看，这意味着解可以在多项式时间内加以验证，因为机器神奇般地知道当试图进行决定时进行什么选择。PSPACE 类型是算法执行中可以用不多于多项式数量的存储空间就可以决定的机器代码的集合（NPSPACE=PSPACE，所以没有非确定性的版本）。EXPTIME 类型是那些在 $O(2^{n^k})$ 时间内决定的机器代码的集合，对于某整数 k。EXPTIME 大于 P，但是并不精确知道 NP 和 PSPACE 的界限在哪里。可能是 P=NP=PSPACE 的情况（尽管很难相信它）或者可能是 NP=PSPACE=EXPTIME。

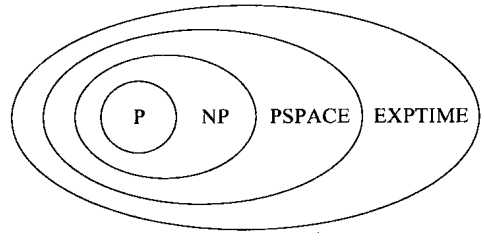


图 6.40 已知 P⊂EXPTIME 是一个真子集；但是不知道 NP 和 PSPACE 究竟有多大

1 注意到，这种定义在实际上可能是很荒谬的：运行时间为 $O(n^{90125})$的算法对于多数应用来说不太有效率。

复杂度与完备性　因为比较容易的类型作为子集被包含在比较复杂的类型中，所以关于一个类型中最难的一个机器代码（例如问题）的概念是非常有用的。令 X 表示 P、NP、PSPACE 或者 EXPTIME。一个机器代码 A 称为 X 复杂度，如果类型 X 中的每个代码 B 是多项式时间可约（polynomial-time reducible）为 A 的。简而言之，这意味着在多项式时间内，B 中任何机器代码都可以转化为机器代码 A 的实例，然后 A 所做的决定能够在多项式时间内正确地转化回来以正确地决定 B。因此如果 A 被决定，那么在多项式时间因子内，X 中每个机器代码都可以决定。复杂度的概念可以应用到不属于这个类型的机器代码（问题）中。例如可以声明一个机器代码是 NP 复杂度的，即使 $A \notin \text{NP}$（例如它可能更难，是 EXPTIME 的）。如果知道机器代码既是某 X 类型复杂度的，同时它也是 X 的一个成员，那么称之为 X-完备的（例如 NP-完备的、PSPACE-完备的等）[1]。因为存在这种关于 P、NP 和 PSPACE 的不确定性，所以我们不能因为一个问题是 NP 复杂度或者 PSPACE 复杂度的就认为它是难处理的，但是如果问题是 EXPTIME 复杂度的，我们能够说问题是难处理的。提示：PSPACE 复杂度意味着 NP 复杂度，记住这一点是很有用的。

运动规划的下界　1979 年 Reif [820]证明表述 4.1 所描述的一般运动规划问题是 PSPACE 复杂度的。实际上，问题限制在多面体障碍物和一个由球形关节连接起来的有限数量多面体机器人。假设所有多面体的坐标都在 $\mathbb{Q}$ 中（这使得可以用一个有限长度串对问题的实例进行编码）。证明引入了一个令人感兴趣的运动规划实例，该实例涉及必须穿过一个复杂的隧道系统的一些机器人，它们一起用来模拟系统图灵机的运行。Canny 后来指出：表述 4.1（用有理数作系数的多项式来表达）中的问题是 PSPACE[175]。因此一般运动规划问题是 PSPACE-完备的。

规划问题也有一些其他的下界。一个著名的例子是如图 6.41 所示的 Warehouseman 问题。在这个问题的矩形环境中有限多个能够平移的矩形。文献[464]表明它是 PSPACE 复杂度的。这个例子阐述：一个貌似简单的问题表述是如何导致一个很高的下界的。最近也揭示到：为 Sokoban 所进行的规划是一个离散二维刚体上的一个 Warehouseman 问题，它也是 PSPACE 复杂度的[257]。其他 PSPACE 复杂度的一般运动规划问题包括平面中物体链的运动规划[463,493]以及 $\mathbb{R}^3$ 中多面体障碍物中物体链的运动规划。已经为一般运动规划问题

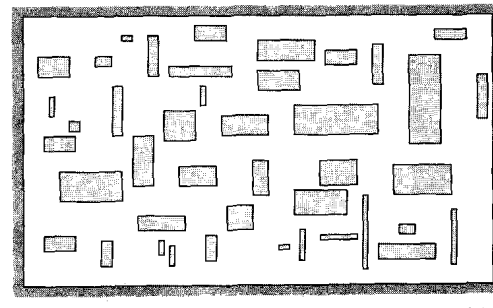


图 6.41　$\mathbb{R}^2$ 中一个矩形盒子内的一群平移矩形的运动规划是 PSPACE-复杂度的（因此也是 NP-复杂度的）

1　如果你听说过规划算法是 NP 的，但是没有记住它是否是 NP 复杂度或者 NP 完备的，那么说 NP 复杂度是比较安全的，因为 NP 完备的意味着 NP 复杂度。这也可以类似地说成是其他类型的，如 PSPACE 完备和 PSPACE 复杂度的。

的各种扩展和变形建立了一些下界。例如在文献[174]中，为三维多面体环境中的一个机器人，在存在不确定的情况下进行一种确定形式的规划是 NEXPTIME 复杂度的，这比图 6.40 中所显示的任何类型都困难；NEXPTIME 中最难的问题据信要求两倍指数时间来解决。

下界或者复杂度结果非常依赖于问题的精确表示。例如通过使实例编码比它们应有的还要长指数倍，这样可能使所产生的问题看起来更容易。所需要的运行时间或者空间用输入的大小 n 来表达。如果运动规划问题实例用比所需用到的比特位还要多指数倍的位来进行编码，那么得到属于 P 的机器代码。只要实例的编码在最优编码的一个多项式因子中（这可以用 Kolmogorov 复杂性来精确确定），那么就可以避免这种难以处理的行为。表示中另外一个重要的部分是注意问题表述中参数是如何变化的。我们可以将运动规划重新定义为所有实例，对于这些实例来说 C 的维数从来不会超过 2^{1000}。对实际应用来说维数都足够大。为这个问题所产生的机器代码属于 P 的，因为圆柱代数剖分和 Canny 算法能够在多项式时间内解决任何运动规划问题。为什么？这是因为时间复杂性表达中的维数参数可以用常数 2^{1000} 代替。这暗示对我们所关心的运动规划问题来说已知一个有效率的算法。但是这种暗示没有实际应用价值。因此在阐述理论的界限的时候必须非常仔细。

下界看起来可能是令人沮丧的。从这里可以向两个更一般的方向发展。一个是弱化对问题的要求，并且容忍产生某种分辨率完备性或者概率完备性的算法。第 5 章采用这种方法，它导致许多有效率的算法。另外一个方向是定义更加严格的问题，它不包括导致不好下界的构造方法。对于更加严格的问题，有可能设计有趣而且有效率的算法。第 6.2 节和第 6.3 节采用的就是这种方法。第 6.5.3 节给出这些问题的某些算法的上界，同时也给出了一般运动规划算法的界。几种上界涉及下面将要介绍的 Davenpot-Schinzel 序列。

6.5.2 Davenport-Schinzel 序列

Davenpot-Schinzel 序列描述了从一个函数族的上下包络产生而来的结构的特征。5 个函数的下包络如图 6.42 所示。这种包络在贯串计算几何的许多问题中都会产生，包括许多运动规划问题。它们是许多现代算法的设计和分析的重要组成部分，所产生的算法的时间复杂性通常涉及直接根据序列而得出的项。因此在阐述第 6.5.3 节的某些结果之前，需要对一些基础知识进行了解。更多关于 Davenpot-Schinzel 序列以及它们的应用的信息参见文献[868]。这里的简要介绍是基于文献[867]的。

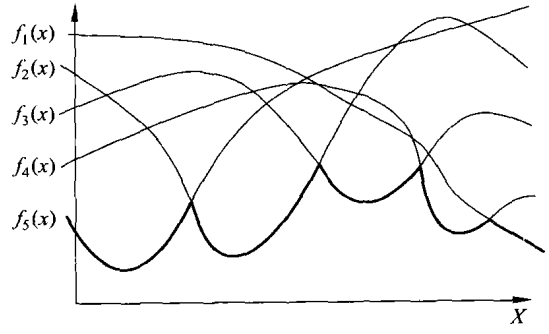


图 6.42 一个函数族的下包络线

对于正整数 n 和 s，一个(n, s)Davenpot-Schinzel 序列是一个由包含 n 个符号的一个集合组成的序列$(u_1,\cdots,u_m)$，使得：

（1）相同的符号在序列中不能相邻出现。即对于任何 i，$1\leqslant i\leqslant m$，$u_i\neq u_{i+1}$。

（2）序列不能包含任何使用两个符号且长度为 $s+2$ 的交替子序列。删除原始序列中的任何元素都可以形成一个子序列。该条件可以表达为：对某些 a 和 b，不存在 $s+2$ 个下标 $i_1<i_2<\cdots<i_{s+2}$ 使得 $u_{i_1}=u_{i_3}=u_{i_5}=a$ 以及 $u_{i_2}=u_{i_4}=u_{i_6}=b$。

作为一个例子，一个$(n, 3)$序列不能出现$(a\cdots b\cdots a\cdots b\cdots a)$，其中每个“…”用任何符号序列来填充。令 $\lambda_s(n)$表示一个(n, s)Davenpot-Schinzel 序列最大的可能长度。

图 6.42 和这些序列之间的联系现在无法进行解释。考虑可以用来访问下包络的函数下标所形成的序列。在例子中，序列是(5,2,3,4,1)。假设知道每对函数至多在 s 个地方相交。如果有 n 个连续实值函数，那么函数下标形成的序列一定是一个(n, s) Davenpot-Schinzel 序列。这种序列不能太长。对一个固定 s 来说，序列接近于是线性的。

Davenpot-Schinzel 序列的标准界为[867][1]：

$$\lambda_1(n)=n \tag{6.32}$$

$$\lambda_2(n)=2n-1 \tag{6.33}$$

$$\lambda_3(n)=\Theta(n\alpha(n)) \tag{6.34}$$

$$\lambda_4(n)=\Theta(n\cdot 2^{\alpha(n)}) \tag{6.35}$$

$$\lambda_{2s}(n)\leqslant n\cdot 2^{\alpha(n)^{s-1}+C_{2s}(n)} \tag{6.36}$$

$$\lambda_{2s+1}(n)\leqslant n\cdot 2^{\alpha(n)^{s-1}\lg\alpha(n)+C'_{2s+1}(n)} \tag{6.37}$$

$$\lambda_{2s}(n)\leqslant \Omega(n\cdot 2^{\frac{1}{(s-1)!}\alpha(n)^{s-1}+C'_{2s}(n)}) \tag{6.38}$$

在上面的表达式中，$C_r(n)$ 和 $C'_r(n)$ 是比它们前面的指数更小的项。$\alpha(n)$项是反 Ackerman 函数，Ackerman 函数是一个增长非常缓慢的函数，它在算法中频繁出现。Ackerman 函数的定义如下。令 $A_1(m)=2m$，$A_{n+1}(m)$表示 A_n 的 m 个应用，因此 $A_1(m)$实现加倍，$A_2(m)$实现幂指数，$A_3(m)$实现塔形幂指数，即产生 2 的一个堆：

$$2^{2^{\cdot^{\cdot^{2}}}} \tag{6.39}$$

它的高度为 m。Ackerman 函数被定义为 $A(n)=A_n(n)$。这个函数增长的速度如此之快以至于 $A(4)$就已经是一个 2 的指数塔，塔的高度为 65 536。因此 Ackerman 函数的反函数 α 的增长速度非常缓慢。如果 n 小于或等于一个 65 536 的指数塔，那么 $\alpha(n)\leqslant 4$。甚至当它出现在 Davenpot-Schinzel 界的指数中，也并不表示增长率很大。

例 6.9（线段的下包络） Davenpot-Schinzel 的一个有趣的应用是 $\mathbb{R}^2$ 中线段的一个集合的下包络。因为最广位置上的多个线段至多在一点相交，所以下包络中边的数量为 $\Theta(\lambda_3(n))=\Theta(n\alpha(n))$。实际上对到达这个界的线段有多种排列方式，见文献[868]。 ■

6.5.3　上界

运动规划问题的上界从解决问题的完备算法的存在性中产生。本节的进行从最一般的

1　用到如下渐近概念：$O(f(n))$表示一个上界，$\Omega(f(n))$表示一个下界，$\Theta(f(n))$表示界是紧的（在上下界之间）。这种符号在大多数关于算法的书中都在使用[245]。

界开始，这是基于第 6.4 节的方法，以简单运动规划问题的界而结束。

一般算法 表述 4.1 所描述的一般运动规划问题的第一个上界来自于圆柱代数剖分的应用[854]。令 n 表示$\mathcal{C}$的维数。令 m 表示$\mathcal{F}$中多项式的数量，它用来定义$\mathcal{C}_{obs}$。从第 4.3.3 节回忆到对于简单例子来说这生长的速度有多快。令 d 表示$\mathcal{F}$中多项式的最大阶数。产生的多项式的最大阶数以 $O(d^{2^{n-1}})$ 为界，多项式的总数以 $O((md)^{3^{n-1}})$ 为界。为运动规划进行圆柱代数剖分所用总运行时间以 $(md)^{O(1)^n}$ 为界[1]。算法是维数 n 的两倍指数关系，但是与 m 和 d 呈多项式关系。从理论上可以断言：在有界维数的运动规划问题的一个空间上它是有效率的（尽管对于任何实际意义下的运动规划来说它不是有效率的）。

因为一般问题是 PSPACE 完备的，一个完备的、一般的运动规划问题算法所需要的运行时间与维数呈指数关系，这是必然的。因为圆柱代数剖分是两倍指数的，这在 20 世纪 80 年代时促使人们思考是否能够降低上界。Canny 的路线图算法实现了这种愿望，它的运行时间以 $m^n(\lg m)d^{O(n^4)}$ 为界。因此它是单倍指数的，它非常接近最优的，因为它面对的是 PSPACE 复杂度所暗示的下界（以及这样一个事实：要求一个有$(md)^n$个连通分支的路线图[79]）。算法的一些复杂性是源于寻找一个适当的确定性扰动来将输入多项式放置到最广位置上。可以交替使用一个随机化的算法，对于该算法来说，随机化的期望运行时间以 $m^n(\lg m)d^{O(n^2)}$ 为界。对于随机化的算法（randomized algorithm）[722]，随机化的期望运行时间依然是最坏情况下的下界，但是在算法中引入的随机“投掷硬币”上进行了平均；它并不反映在期望输入的分布上任何类型的平均。因此这两个界表示一般运动规划问题的最公认的上界。Canny 的算法也可以应用于解决第 4.4 节运动的封闭问题，但是复杂性没有反映代数簇的维数 k 小于$\mathcal{C}$的维数 n 这个事实。文献[78]介绍了特别适合这个问题的一个路线图算法，它的运行时间以 $m^{k+1}d^{O(n^2)}$ 为界。这就是第 4.4 节问题的最公认的上界。

特殊算法 现在概要介绍一些更严格问题的上界，这些问题比一般问题解决起来更有效率。所有问题都涉及两个或者 3 个自由度。因此我们期望它们的界比一般问题的界小很多。在一些情况下会产生第 6.5.2 节的 Davenpot-Schinzel 序列。这里提出的大多数的界所基于的算法对实现来说并不切合实际；它们主要用来指示所能得到的最好的渐近性能。这里提及的大多数的界在文献[867]中都有论述。

考虑第 6.2 节的问题，其中机器人在 $\mathcal{W}=\mathbb{R}^2$ 中平移，$\mathcal{C}_{obs}$ 是多边形。假设$\mathcal{A}$是有 k 条边的凸多边形，$\mathcal{O}$是 m 个没有交集的、凸的多边形的并集，它们的边的总数为 n。在这种情况下，$\mathcal{C}_{free}$ 的边界（用 Minkowski 差进行计算，见第 4.3.2 节）至多有 $6m-12$ 个非反射顶点（内角小于 π）以及 $n+km$ 个反射顶点（内角大于 π）。可以在 $O((n+km)\lg^2 n)$ 时间内完成对空间$\mathcal{C}_{free}$ 的剖分和搜索[521,867]。采用随机化的算法，随机化期望时间的界减小到 $O((n+km)\cdot 2^{\alpha(n)}\lg n)$。现在假设$\mathcal{A}$是 k 条边描述的一个非凸多边形区域，$\mathcal{O}$是 n 条边描述的一个相似的多边形区域。Minkowski 差产生的边和$\mathcal{C}_{obs}$的一样多，都是 $\Omega(k^2n^2)$。如果在 $\mathcal{C}_{free}$ 的一个单连通分支中进行搜索，那么可以避免这种情况。基于采用 Davenpot-Schinzel 序列所进行的分析可以证明：最坏的连通分支可能的复杂度为 $\Theta(kn\alpha(k))$，如果采用确定性方法，那么规划问题可以在 $O(kn\lg^2 n)$ 时间内得到解决；如果采用随机化算法，那么需

1 在表达式中出现 $O(\cdot)$ 是很奇怪的。在这里，它意味着存在某 $c\in[0,\infty)$ 使得运行时间以 $(md)^{cn}$ 为界。注意到，在这个公式前面不需要另外一个 O。

要的随机化期望时间是 $O(kn \cdot 2^{\alpha(n)} \lg n)$。更一般情况下，如果 C_{obs} 由 $\mathbb{R}^2$ 中的 n 条代数曲线组成，每条曲线的自由度不超过 d，那么对于只有平移的运动规划问题，用确定性方法可以在 $O(\lambda_{s+2}(n)\lg^2 n)$ 时间内得到解决，用随机化算法则需要随机化期望时间 $O(\lambda_{s+2}(n)\lg n)$。在这些表达中，$\lambda_{S+2}(n)$ 是式（6.37）所表示的界，它是从(n, S+2) Davenpot- Schinzel 序列得到的，其中 $s \leqslant d^2$。

在用 n 条边描述的障碍区域中，对于第 6.3.4 节的一个的线段机器人，给定一个 $O(n^5)$ 时间的算法。对于解决线段问题来说，这不是最好的可能运行时间，但是与其他更有效的方法比较而言，该方法更容易理解。文献[751]给出的一个基于收缩的路线图算法将在 $O(n^2 \lg n \lg^* n)$时间内解决问题，其中 $\lg^* n$ 是为产生一个小于等于 1 的结果，lg 在 n 上不得不进行的迭代的次数（例如它是一个微小的项；在实际情况下，可以假设运行时间是 $O(n^2 \lg n)$）。已知的最紧上界是 $O(n^2 \lg n)$[628]。文献[520]指出：存在这样的例子，它们的路径解需要长度 $\Omega(n^2)$来进行编码。在 $\mathbb{R}^3$ 中，在总共有 n 个顶点的多面体障碍物中间运动的一个线段机器人，文献[550]给出了一个运行时间为 $\Omega(n^4+\epsilon)$的完备算法，$\epsilon>0$。文献[520]指出：复杂度为 $\Omega(n^4)$的路径解是存在的。

现在考虑 $C=SE(2)$，$\mathcal{A}$是有 k 条边的凸多边形，$\mathcal{O}$是用 n 条边描述的一个多边形区域。C_{free} 的边界有不多于 $O(kn\lambda_6(kn))$ 条边，可以在 $O(kn\lambda_6(kn)\lg kn)$ 时间内计算 C_{free} 的边界来解决运动规划问题[10,11]。文献[207]给出了一种运行时间为 $O(k^4 n\lambda_3(n)\lg n)$ 且能够提供机器人与障碍物之间更好间隙的算法。在文献[56]中（一些详细内容可参见文献[591]），作者提出了一种算法及其实现，该算法能够在 $O(k^3n^3\lg(kn))$时间内解决$\mathcal{A}$为非凸的这种更一般的情形。对于这个问题，C_{obs} 的面的数量可能高达 $\Omega(k^3n^3)$。通过显式表示和搜索一个连通分支，该问题最公认的上界是 $O((kn)^{2+\epsilon})$，其中选择 ϵ 任意小[431]。

最后一种情况，假设$\mathcal{A}$在 $\mathcal{W}=\mathbb{R}^3$ 中进行平移产生 $C=\mathbb{R}^3$。对于一个多面体或者多面体区域，令它的复杂度为面、边、顶点的总数。如果$\mathcal{A}$是复杂度为 k 的多面体，$\mathcal{O}$是复杂度为 n 的多面体区域，那么C_{free} 的边界是一个复杂度为 $\Theta(k^3n^3)$ 的多面体表面。至于其他问题，如果搜索限制在一个单个分支中，那么复杂度减小。这种情况下的运动规划问题可以在 $O((kn)^{2+\epsilon})$ 时间内解决[42]。如果$\mathcal{A}$是凸的，有 m 个凸障碍物，那么公认的界是 $O(kmn\lg^2 m)$ 时间。更一般情况下，如果 C_{obs} 以 n 个有最大常数阶的代数片为界，那么在 C_{free} 的一个单连通分支内垂直剖分方法用 $O(n^{2+\epsilon})$ 时间解决运动规划问题。

进一步阅读

大多数关于组合规划的文献都比基于采样的规划文献历史更悠久。文献[856]很好地汇集了以前的文献；这还包括文献[463,751,752,820,853,854,855]。Latombe 的经典运动规划[591]介绍了本章所涉及的大多数方法。但是本章介绍的内容并不完全遵循文献[591]，因为它将胞腔剖分和路线图方法列为不同类型。而构造胞腔剖分是为了产生一个路线图；因此在本章它们是统一的。关于运动规划的组合算法、计算几何和完备算法，文献[405]对它们进行了非常全面的综述。

第 6.2 节的思路是从计算几何的基本算法产生而来的。对于计算几何的介绍，包括对

垂直剖分的详细解释和说明，参见文献[266]。其他关于计算几何的资料包括文献[129,304,809]。为了理解计算最优多边形剖分的复杂性，参见文献[760]。更多计算最短路径的资料，参见文献[653,712,835]。

胞腔剖分和胞腔复形是计算几何和代数拓扑中的重要内容。第 6.3 节对它们应用于运动规划的前景进行了论述。关于代数拓扑中的单纯复形，参见文献[499,538,837]；奇异复形，参见文献[837]。在计算几何中有多种类型的胞腔剖分。其中得到广泛研究的是三角剖分[92]和布置[429]，它们是由诸如平面中的直线或者圆这样的本原集合生成的区域。早期运动规划中的胞腔剖分，参见文献[856]。文献[954]对计算拓扑进行了综述。

关于第 6.4 节内容的一个最新和最完整的参考资料是文献[79]。文献[252]则介绍了计算代数几何。关于带多项式的代数计算的详细介绍，参见文献[707]。文献[708]对计算代数几何进行了综述。除了文献[79]以外，其他关于圆柱代数剖分的参考文献还有[40,234]。它在运动规划中的使用，见文献[591,854]。关于 Canny 路线图算法的主要参考文献是[175]。对第 6.4.3 节中提到的一种算法的全面的评述出现在文献[222,591]中。文献[79]则介绍了对算法的变化及改进。文献[178]研究了 Canny 路线图算法的一个基于势函数的扩展。

更多关于运动规划的复杂性，可以参考第 6.5 节所给出的大量文献。

习　题

1. 将垂直剖分算法进行扩展，使之能够正确处理C_{obs}中有两个或者两个以上的点处在相同的垂线上这种情况。这包括垂直线段的情况。不允许有随机扰动。

2. 完整描述和证明图 6.14 所示的双切线计算方法的正确性，它没有采用三角函数。确信考虑了所有类型的双切线（在最广位置）。

3. 设计一个算法，它用平面扫描原理来有效地对两个非凸多边形的并集进行计算。

4. 将第 6.2.2 节的垂直胞腔剖分算法进行扩展，使之能够应用于描述为圆弧和线段的链的障碍物边界。

5. 将第 6.2.4 节的最短路径路线图算法进行扩展，使之能够应用于描述为圆弧和线段的链的障碍物边界。

6. 为一个线段机器人同时接触到一个障碍物顶点和边这种情况推导如图 6.24 所示的 Nicomedes 螺旋线方程。

7. 为在一个多边形障碍区域中的线段机器人的运动规划，提出一个完备分辨率算法。算法应该为任意确定方向 θ 计算精确 C-空间障碍物切片；但是算法应该用在方位的集合 $[0, 2\pi)$上进行 Van der Corput 采样。

8. 单位球体$\mathbb{S}^1,\mathbb{S}^2,\mathbb{S}^3,\mathbb{S}^4,\cdots$，每个球体$\mathbb{S}^n$都表示为$\mathbb{R}^{n+1}$中的一个单位球体，确定圆柱代数剖分的结果。用图来描述$\mathbb{S}^1$和$\mathbb{S}^2$的情况，并且研究设计一个表达式将胞腔的数量表示成 n 的函数。

9. 对如图 6.43 所示的三个圆进行圆柱代数剖分，能够得到多少胞腔？

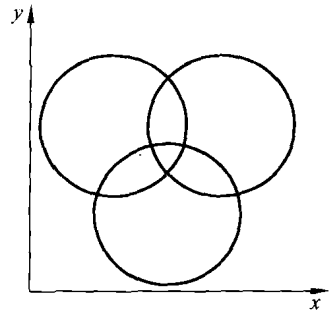


图 6.43　通过投影到 x 轴来确定圆柱代数剖分

10. 用式（6.28）中的矩阵，说明图 6.39 所示环面的 Canny 路线图结果是正确的。环面方程用：

$$\left(x_1^2+x_2^2+x_3^2-(r_1^2+r_2^2)\right)^2-4r_1^2(r_2^2-x_3^2)=0 \tag{6.40}$$

其中 r_1 是外圆，r_2 是内圆，$r_1>r_2$。

11. 为一个能够在平面中平移且能够连续改变其尺寸的多边形机器人设计一个垂直剖分算法。如何修改算法以应用于能够平移或者能够被切变的机器人？

12. 为一个平的圆环设计一个最短路径路线图算法，圆环是通过等同一个正方形对边来定义的。用欧氏距离，但是当确定最短路径时需要服从等同原理。假设机器人是一个点，障碍物是多边形的。

实现

13. 实现第 6.2.2 节的垂直胞腔剖分规划算法。

14. 实现第 6.2.3 节的最大间隙路线图规划算法。

15. 实现在 $\mathcal{W}=\mathbb{R}^3$ 中多面体障碍物中间运动的一个点机器人的规划算法。用垂直剖分方法。

16. 实现一个算法，对一个多边形障碍区域进行圆柱剖分。

17. 实现一个算法，它为线段机器人而计算第 6.3.4 节所进行的胞腔剖分。

18. 用圆柱代数剖分进行试验。利用已有的实现计算代数几何中的基本运算的程序包，将会有利于试验工作的进行。

19. 实现习题 7 中提出的算法。

第 7 章　基本运动规划的扩展

本章对第 3 章～第 6 章中所考虑的运动规划问题进行扩展。它们都可以认为是一个很容易用目前介绍的数学概念和算法进行描述的“副产品”。不像以前的章节，第 7 章没有太多的连贯性。每个问题独立对待；因此即使跳到你所感兴趣的部分也不用担心会错过重要的细节。

在本章的许多地方会出现状态空间 X。这与本书中的一般规划符号一致。在第 4 章引入 C-空间 $\mathcal{C}$，它可以认为是一个特殊的状态空间：它对可以应用于物体族的变换集合进行编码。因此第 5 章和第 6 章讨论在 $X=\mathcal{C}$ 中进行的规划。对于本章的一些问题来说，仅有 C-空间是不够的；因此采用 X，因为它更加一般。但是对于本章的大多数情形，X 从一个或者多个 C-空间产生而来。因此 C-空间和状态空间将一起使用。

7.1　时 变 问 题

本节将时间引入到运动规划表述中。尽管已经允许机器人进行运动，但到目前为止仍假设障碍区域 $\mathcal{O}$ 和目标位形 $q_G \in \mathcal{C}_{free}$ 在所有时间里是静止的。现在假设这些都随时间而变化，尽管它们的运动是可以预测的。如果运动不可预测，那么需要某种形式的反馈来对执行过程中所进行的观测产生响应。这种问题更加复杂，将在第 8 章和第 4 部分进行论述。

7.1.1　问题的表述

所设计的表述允许直接应用我们所学的工具和概念。令 $T \subset \mathbb{R}$ 表示时间区间，它是有界的或者无界的。如果 T 有界，那么 $T=[0,t_f]$，其中 0 是初始时刻，t_f 是结束时刻。如果 T 无界，$T=[0,\infty)$。初始时刻也可以不定义为 0，但是这里不这样处理。

令状态空间 X 定义为 $X=\mathcal{C}\times T$，其中 $\mathcal{C}$ 是机器人通常的 C-空间，如第 4 章所定义。一个状态 x 表示为 $x=(q,t)$，它包括位形分量 q 和时间分量 t。直接在 X 中进行规划，有许多方式将它当作目前所见到的任何 C-空间来对待，但是有一个关键的区别：时间是向前进行的。假设 X 中有一条路径，如果它首先到达状态 $(q_1,5)$，然后到达某状态 $(q_2,3)$，这要求时间反向进行。这样的时间流逝对数学来说没有问题，但是对应用来说是不现实的。因此 X 中的路径必须服从时间向前进行这个约束。

现在考虑产生表述 4.1 的运动规划的时变版本。

表述 7.1（时变运动规划问题）

1. 一个世界 $\mathcal{W}$，其中 $\mathcal{W}=\mathbb{R}^2$ 或者 $\mathcal{W}=\mathbb{R}^3$。这与表述 4.1 相同。
2. 一个时间区间 $T \subset \mathbb{R}$，如果 T 有界，那么 $T=[0,t_f]$，$t_f>0$ 是某结束时刻。如果 T

无界，那么 $T=[0,\infty)$。

3. 对于每个 $t\in T$，有一个半代数、时变障碍区域 $\mathcal{O}(t)\subset\mathcal{W}$。假设障碍区域是刚体的一个有限族，刚体经历的是连续的、与时间相关的变换。

4. 机器人 $\mathcal{A}$（或者连杆组 $\mathcal{A}_1,\cdots,\mathcal{A}_m$）和位形空间 $\mathcal{C}$ 的定义与表述 4.1 相同。

5. 状态空间 X 是笛卡儿积 $X=\mathcal{C}\times T$，一个状态 $x\in X$ 表示为 $x=(q,t)$，它表示位形 q 和时间 t，如图 7.1 所示。状态空间中障碍区域 X_{obs} 定义为：

$$X_{obs}=\{(q,t)\in X\,|\,\mathcal{A}(q)\cap\mathcal{O}(t)\neq\phi\} \tag{7.1}$$

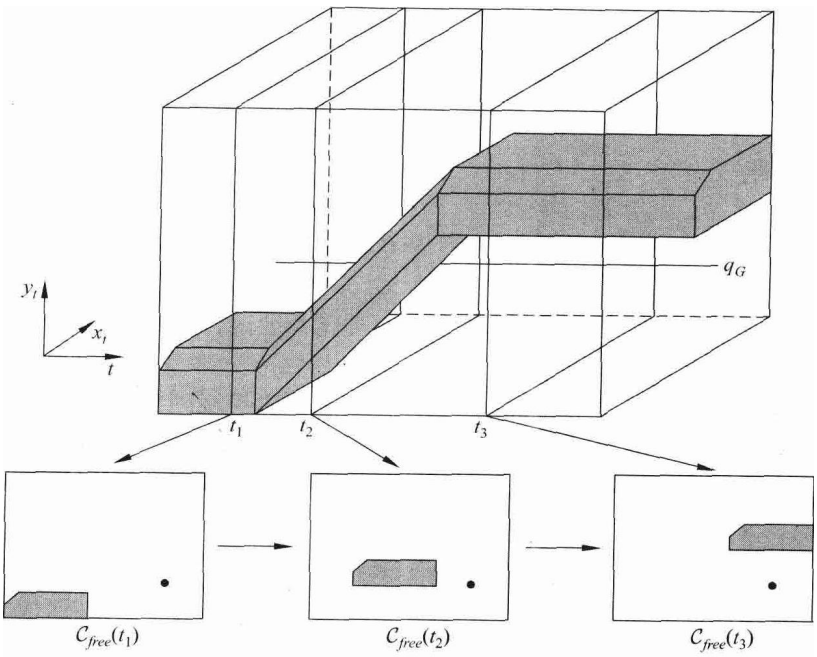


图 7.1　有分段线性的障碍物运动的一个时变例子

$X_{free}=X\setminus X_{obs}$。对于给定的 $t\in T$，可以得到 X_{obs} 和 X_{free} 片段。它们分别表示 $\mathcal{C}_{obs}(t)$ 和 $\mathcal{C}_{free}(t)$，其中（假设 $\mathcal{A}$ 是一个物体）：

$$\begin{aligned}\mathcal{C}_{obs}(t)&=\{q\in\mathcal{C}\,|\,\mathcal{A}(q)\cap\mathcal{O}(t)\neq\phi\}\\ \mathcal{C}_{free}&=\mathcal{C}\setminus\mathcal{C}_{obs}\end{aligned} \tag{7.2}$$

6. 状态 $x_I\in X_{free}$ 指定为初始状态，对于某 $q_I\in\mathcal{C}_{free}(0)$，存在约束 $x_I=(q_I,0)$。也就是说在初始时刻，机器人不能有碰撞。

7. 子集 $X_G\in X_{free}$ 指定为目标区域。一个典型的定义是选取某 $q_G\in\mathcal{C}$，并且令 $X_G=\{(q_G,t)\in X_{free}\,|\,t\in T\}$，即目标在所有时间内是静止的。

8. 一个完整算法必须计算一条连续的、时间单调的路径 $\tau:[0,1]\to X_{free}$，使 $\tau(0)=x_I$，$\tau(1)=x_G$，或者报告这样的路径不存在。时间单调意味着对于任何 $s_1,s_2\in[0,1]$，$s_1<s_2$，都有 $t_1<t_2$，其中 $(q_1,t_1)=\tau(s_1)$ 和 $(q_2,t_2)=\tau(s_2)$。

例 7.1（分段线性的障碍物运动） 图 7.1 显示了一个在 $\mathcal{W}=\mathbb{R}^2$ 中平移的凸多边形机器

人 $\mathcal{A}$。有一个凸多边形障碍物 $\mathcal{O}$。它们一起产生了一个凸多边形 C-空间障碍物 $\mathcal{C}_{obs}(t)$，图中显示了 t_1, t_2 和 t_3 三个时刻的情况。障碍物按照分段线性运动模型进行运动，即 $\mathcal{O}$ 的变换是时间的分段线性函数。例如令 (x, y) 为 $\mathcal{O}$ 上的一个固定点。作为一个线性运动模型，该点必须变换为 $(x + c_1 t, y + c_2 t)$，其中常数 $c_1, c_2 \in \mathbb{R}$。作为分段线性的，在有限个临界时刻它可以变为另外一种不同的线性运动。在这些临界时刻之间，运动必须保持线性的。本例有两个临界时刻。如果 $\mathcal{C}_{obs}(t)$ 是多边形的，并且采用分段线性运动模型，那么 X_{obs} 是多面体的，如图 7.1 所示。图中也显示了静止的目标，该目标是一条平行于 T 轴的直线。 ■

在一般表述中，对路径 τ 没有额外约束，即机器人运动模型允许无限加速度和无界速度。机器人的速度可以在瞬间发生变化，但是 $\mathcal{C}$ 中的路径必须总是连续的。这些问题在第 4 章中并没有提出，因为在那里没有必要提及时间。现在这个问题变得必要了[1]。

7.1.2 直接解

基于采样的方法 一些基于采样的方法从 $\mathcal{C}$ 改为 X 并不困难。当验证路径段是无碰撞的时候，必须考虑障碍模型的时间性；扩展第 5.4.3 节的内容就可以处理该问题。需要关注的是 X 的度量。某些算法允许使用伪度量，这是非常重要的，因为对称性被时间打破（时间不能向后进行）。

例如假设 C-空间 $\mathcal{C}$ 是一个度量空间 $(\mathcal{C}, \rho)$。度量可以扩展到与时间有关来得到一个伪度量 ρ_X。对于状态对 $x = (q, t)$ 和 $x' = (q', t')$，令：

$$\rho_X(x, x') = \begin{cases} 0 & (q = q') \\ \infty & (q \neq q', t' \leqslant t) \\ \rho(q, q') & \text{其他情况} \end{cases} \tag{7.3}$$

使用 ρ_X，那么有几种基于采样的方法就可以很自然地得到应用。例如第 5.5 节的 RDT 可以改为 X。在单树方法中使用 ρ_X 可以确保所有路径段随时间向前行进。对时变问题来说使用双向方法更加困难，因为 X_G 通常不是一个点。不知道哪个 (q,t) 应该是目标树的开始顶点；一种可能是将目标树初始化为整个时不变部分。第 5.6 节的基于采样的路线图方法可能是最容易进行适应性修改的。需要有向图这个概念，其中每条边必须是有向的以产生一条时间单调的路径。对每个状态对 (q,t) 和 (q',t')，$t \neq t'$，对都存在一个有效的方向来说产生一条潜在的边。如果 $t = t'$，那么不需要产生边，因为它要求机器人从 $\mathcal{W}$ 的一个点处瞬间“远距传输”到另外一个点。因为有向边已经考虑到时间向前的进行，所以可以采用对称度量，而不是基于采样的路线图方法所采取的（见式（7.3））。

组合方法 在某些情况下，组合方法可以解决时变问题。如果运动模型是代数的（例如用多项式表达），那么 X_{obs} 是半代数的。这使得可以应用第 6.4 节的基于计算实代数几何的一般规划器。关键的问题还是产生的路线图必须是有向的，并且所有边是时间单调的。对于 Canny 路线图算法来说，这个要求似乎难以保证。圆柱代数剖分则很容易进行适应性

1 无限加速度和无界速度这种假设使具有力学和控制论背景的人员非常苦恼。在这种情况下，假设目前的模型是对慢速运动、可以忽略动力学这种情况的近似。如果这还不满意，那么请跳至第 VI 部分，在那里考虑了一般非线性系统。考虑从本章概念所产生的含义是有益的，因为对于涉及动力学的更复杂的问题来说，这些难题依然存在。

修改，只要在投影序列中选择时间作为最后需要考虑的变量。这产生 $\mathbb{Q}[t]$ 中的多项式以及 $\mathbb{R}$ 被分拆为时间区间和时间场合。然后将一个圆柱的一个胞腔连接到后来的一个圆柱的邻近胞腔。

如果 X_{obs} 是如图 7.1 所示的多面体，那么可以采用垂直剖分。最好首先沿时间轴扫过平面，停在当线性运动发生改变的临界时间点上。这样所产生的剖面将进一步递归剖分，如第 6.3.3 节所述，这也有利于连接邻近胞腔以得到时间单调路径段。不难想象针对一个四维状态空间 X 的方法，其中 $C_{obs}(t)$ 是第 6.3.3 节所示的多面体，时间增补为第四维。仍然可以先进行关于时间轴第一次扫描。

如果在每个非临界时间区间上 X 没有分解为圆柱片段，那么可能还要使用胞腔剖分，但是需要仔细正确地连接胞腔。图 7.2 阐明这样的问题，其邻近胞腔之间的传递性被打破。这使得为胞腔而进行的采样点的选择更加复杂。

有界速度　到现在都没有考虑机器人避开障碍物的运动速度。在许多应用中，要求机器人运动速度任意快是不现实的。迈向建立一个现实的模型的一个步骤是给机器人的速度加一个界限（进一步的步骤请参见第 13 章）。为简单起见，假设 $C=\mathbb{R}^2$，这对应于 $\mathcal{W}=\mathbb{R}^2$ 中的一个平移刚体机器人 $\mathcal{A}$。一个位形 $q\in C$ 表示为 $q=(y,z)$（因为 x 已经用来表示整个状态向量）。机器人速度表示为 $v=(\dot{y},\dot{z})\in\mathbb{R}^2$，其中 $\dot{y}=\mathrm{d}y/\mathrm{d}t$ 和 $\dot{z}=\mathrm{d}z/\mathrm{d}t$。机器人的速度 $\|v\|=\sqrt{\dot{y}^2+\dot{z}^2}$。速度的界 b 是正值常数 $b\in(0,\infty)$，那么 $\|v\|\leqslant b$。

按照图 7.1，这意味着路径解 τ 的斜率是有界的。假设 τ 的定义域为 $T=[0,t_f]$，而不是 $[0,1]$。这产生 $\tau:T\to X$ 和 $\tau(t)=(y,z,t)$。使用这种表示，$\mathrm{d}\tau_1/\mathrm{d}t=\dot{y}$ 和 $\mathrm{d}\tau_2/\mathrm{d}t=\dot{z}$，其中 τ_i 表示 τ 的第 i 个分量（因为它是一个向量值函数）。因此可以看到 b 约束了 X 中 $\tau(t)$ 的斜率。为了形象起见，假设仅有 y 方向运动，假设 b=1。如果 τ 保持机器人固定，那么速度为零，该速度满足任何界限。如果机器人运动速度为 1，那么 $\mathrm{d}\tau_1/\mathrm{d}t=1$ 和 $\mathrm{d}\tau_2/\mathrm{d}t=0$，满足速度限制。在图 7.1 中，产生的路径在 yt 平面内的斜率为 1，在 zt 平面内是水平的。如果 $\mathrm{d}\tau_1/\mathrm{d}t=\mathrm{d}\tau_2/\mathrm{d}t=1$，那么超出界限，因为速度为 $\sqrt{2}$。一般情况下，任何状态 (y,z,t) 的速度向量指向一个圆锥，这个锥从 (y,z) 开始，对准正 t 方向；如图 7.3 所示。在 $t+\Delta t$ 时刻，状态必须在锥内，即：

$$\big(y(t+\Delta t)-y(t)\big)^2+\big(z(t+\Delta t)-z(t)\big)^2\leqslant b^2(\Delta t)^2 \tag{7.4}$$

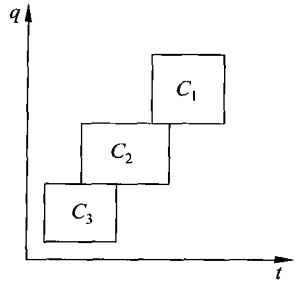


图 7.2　如果在 T 上圆柱体内没有形成胞腔，那么传递性就被打破；从 C_1 到 C_2、从 C_2 到 C_3 都存在一条时间单调的路径，但是这并不意味着存在一条从 C_1 到 C_3 的路径

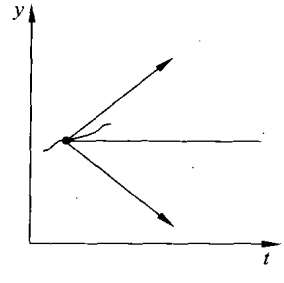


图 7.3　有界速度问题的圆锥约束的一个投影

这种约束使得第 5 章和第 6 章的算法更难进行适应性修改。甚至对于障碍物的分段线性运动，问题被确认为 PSPACE 复杂度的[821,822,929]。文献[822]提出的完备算法与第 6.2.4 节的最短路径路线图算法很相似。第 5.6 节的基于采样的路线图可能是最容易进行适应性修改的采样算法之一。为试图进行连接而确定的点 q 的邻近点必须处在表示速度界限的圆锥内。如果施加该约束，会产生一个分辨率完备或者概率完备的规划算法。

7.1.3 速度调节方法

另外一种在 $C\times T$ 中定义问题的方法是将它解耦成一个路径规划部分和一个运动定时部分[509]。基于这种方法的算法是不完备的，但是速度调节是一个可以应用于任何其他地方的重要思路。假设既有静止障碍物，也有运动障碍物。对于静止障碍物来说，假设某路径 $\tau:[0,1]\rightarrow C_{free}$ 已经用第 5 章和第 6 章中的方法进行了计算。

然后在第二阶段进行定时部分的处理。设计一个定时函数 $\sigma:T\rightarrow[0,1]$，它指示 t 时刻机器人沿路径 τ 的位置。这由定义代数运算 $\phi=\tau\circ\sigma$ 来完成，运算将 T 映射到通过[0,1]的 C_{free}。因此 $\phi:T\rightarrow C_{free}$。$t\in T$ 时刻的位形表示为 $\phi(t)=\tau(\sigma(t))$。

一个二维状态空间可以像图 7.4 那样定义。目的是将对 σ 的设计转换为我们所熟知的一个规划问题。机器人必须沿着它的路径从 $\tau(0)$ 到 $\tau(1)$，同时一个障碍物 $\mathcal{O}(t)$ 在时间区间 T 内沿着它的路径运动。令 $S=[0,1]$ 表示 τ 的定义域。一个状态空间 $X=T\times S$ 如图 7.4(b)所示，其中每个点 (t,s) 指示时间 $t\in T$ 和沿着路径 $s\in[0,1]$ 的位置。X 中的障碍区域定义为：

$$X_{obs}=\{(t,s)\in X\,|\,\mathcal{A}(\tau(s))\cap\mathcal{O}(t)\neq\phi\} \tag{7.5}$$

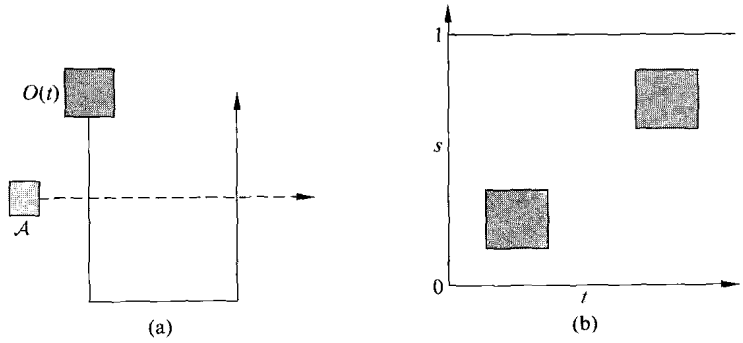


图 7.4 阐述路径调节；（a）如果机器人跟随它所计算得到的路径，那么它可能与运动障碍物发生碰撞；（b）所产生的状态空间

X_{free} 被定义为 $X_{free}=X\setminus X_{obs}$。任务是寻找一条连续的路径 $g:[0,1]\rightarrow X_{free}$。如果 g 是时间单调的，那么为每个时间 $t\in T$ 分配一个位置 $s\in S$。这些分配可以成为定时函数 $\sigma:T\rightarrow S$，从这个函数，通过 $\phi=\tau\circ\sigma$ 可以得到 ϕ 来确定机器人在每个时刻将要到达的位置。这里的时间单调是指路径必须总是从图 7.4(b)的左边行进到右边。在正 s 方向它可以不是单调的，即可以沿着 τ 前后运动，这样会引起重复访问有些位形。

表述 7.1 中描述的任何方法都可以在这里应用。在这个情况下 X 的维数总是 2。注意到 X_{obs} 是多边形，如果 $\mathcal{A}$ 和 $\mathcal{O}$ 都是多边形区域并且它们的路径是分段线性的。在这种情况

下，通过沿着时间轴进行扫描，就可以应用第 6.2.2 节的垂直剖分方法来产生一个完备的算法（对 τ 进行约束后，它是完备的，但是对表述 7.1 来说它是不完备的）。结果如图 7.5 所示。仅在如果可能沿向前的时间方向从一个胞腔行进到另外一个胞腔时，才连接胞腔。当 X_{obs} 不是多边形时，基于采样的方法可能是更可取的，作为这样一个例子，在 X 上放置一个栅格，应用第 5.4.2 节描述的经典搜索方法之一。在 X 中只允许按时间向前运动的路径段。

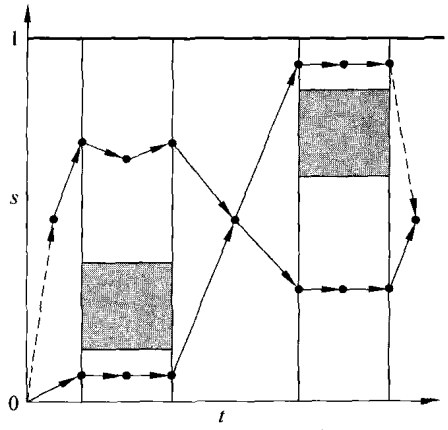


图 7.5　垂直胞腔剖分可以解决路径调节问题；注意这个例子没有处在最广位置，因为存在垂直的边；目的是到达顶部的水平线，这可以从任意相邻 2-胞腔来完成；对于这个例子来说，它甚至可能从第一个 2-胞腔来完成，如果机器人的运动速度足够快

7.2　多 机 器 人

假设在一个世界 $\mathcal{W}$ 中有多个机器人。必须为每个机器人计算一条避免与障碍物和其他机器人发生碰撞的路径。在第 4 章中，每个机器人都是一个刚体 $\mathcal{A}$ 或者由 k 个连接体 $\mathcal{A}_1,\cdots,\mathcal{A}_k$ 组成。为了避免混淆，本节使用上标表示不同的机器人。第 i 个机器人表示为 $\mathcal{A}^i$。假设有 m 个机器人 $\mathcal{A}^1,\mathcal{A}^2,\cdots,\mathcal{A}^m$。每个机器人 $\mathcal{A}^i$ 都有其相应的 C 空间 C^i 以及它的初始位形 q^i_{init} 和目标位形 q^i_{goal}。

7.2.1　问题的表述

同时考虑所有机器人的状态空间定义为：

$$X = C^1 \times C^2 \times \cdots \times C^m \tag{7.6}$$

一个状态 $x \in X$ 指定所有机器人位形，它可以表示为 $x = (q^1, q^2, \cdots, q^m)$。$X$ 的维数为 N，其中 $N = \sum_{i=1}^{m} \dim(C^i)$。

在状态空间中有两种情况产生障碍区域：（1）机器人-障碍物碰撞；（2）机器人-机器人碰撞。对于每个 i，$1 \leqslant i \leqslant m$，对应于与障碍区域 $\mathcal{O}$ 发生碰撞的机器人 $\mathcal{A}^i$ 的 X 的子集为：

$$X^i_{obs} = \left\{ x \in X \middle| \mathcal{A}^i(q^i) \cap \mathcal{O} \neq \phi \right\} \tag{7.7}$$

这是机器人-障碍物碰撞的情况。

对于每个机器人 $\mathcal{A}^i$，与机器人 $\mathcal{A}^j$ 发生碰撞的 X 的子集为：

$$X_{obs}^{ij}=\left\{x\in X\middle|\mathcal{A}^i(q^i)\cap\mathcal{A}^j(q^j)\neq\phi\right\} \tag{7.8}$$

式（7.7）和式（7.8）在式（7.10）中组合在一起产生了 X_{obs}。

表述 7.2（多机器人运动规划）

1. 世界 $\mathcal{W}$ 和障碍区域 $\mathcal{O}$ 与表述 4.1 相同。
2. 有 m 个机器人 $\mathcal{A}^1,\cdots,\mathcal{A}^m$，每个机器人由一个或者多个物体组成。
3. 对于 i 从 1 到 m，每个机器人 $\mathcal{A}^i$ 都有一个相应的位形空间 $\mathcal{C}^i$。
4. 状态空间 X 定义为笛卡儿积：

$$X=\mathcal{C}^1\times\mathcal{C}^2\times\cdots\times\mathcal{C}^m \tag{7.9}$$

X 中的障碍区域为：

$$X_{obs}=\left(\bigcup_{i=1}^{m}X_{obs}^{i}\right)\cup\left(\bigcup_{ij,i\neq j}X_{obs}^{ij}\right) \tag{7.10}$$

其中 X_{obs}^{i} 和 X_{obs}^{ij} 分别是机器人-障碍物碰撞状态（见式（7.7））和机器人-机器人碰撞状态（见式（7.8））。

5. 状态 $x_I\in X_{free}$ 被指定为初始状态，其中 $x_I=(q_I^1,\cdots,q_I^m)$。对于每个 i，$1\leqslant i\leqslant m$，q_I^i 指定 $\mathcal{A}^i$ 的初始位形。
6. 状态 $x_G\in X_{free}$ 被指定为目标状态，其中 $x_G=(q_G^1,\cdots,q_G^m)$。
7. 任务是计算一条连续路径 $\tau:[0,1]\to X_{free}$ 使得 $\tau(0)=x_{init}$ 和 $\tau(1)=x_{goal}$。

这是一个普通的运动规划问题 看起来关于多机器人问题没有什么不寻常，因为第 4 章已经涵盖了多机器人的情形。多个机器人不一定连在一起；因此 X 可以认为是一个普通的 C 空间。第 5 章和第 6 章的规划算法不需要进行适应性修改就可以加以应用。但是需要注意的是 X 的维数关于机器人的数量呈线性增长。例如，如果有 12 个刚体，它们每个有 $\mathcal{C}^i=SE(3)$，那么 X 的维数为 6×12=72。完备算法要求的时间至少是维数的指数关系，所以对这种问题来说它们不可能是候选方案。当有许多机器人时，采用基于采样的算法进行概略估算可能更好，但是 X 的维数依然增长太快。

研究多机器人运动规划的原因 即使多机器人运动规划能够像其他运动规划问题一样进行处理，仍然还有单独研究它的原因：

（1）机器人的运动可以用许多有趣的方法进行解耦。这导致了几种有趣的方法，首先对每个机器人单独进行某种部分规划，然后考虑部分规划之间的相互影响来产生规划解。这种思路称为解耦规划。

（2）由于机器人-机器人之间碰撞而引起的 X_{obs} 中的一部分具有一个圆柱结构，如图 7.6 所

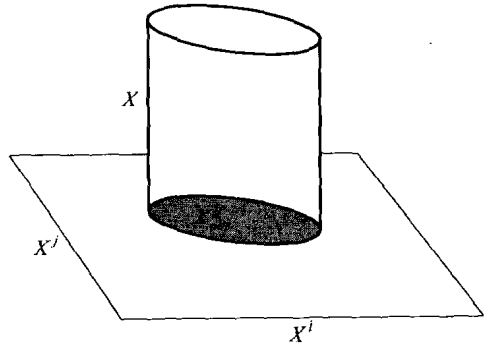


图 7.6 集合 X_{obs}^{ij} 及其在 X 上的圆柱结构

示，这种圆柱结构可以用来产生更加高效的规划算法。由式（7.8）定义的每个 X_{obs}^{ij} 仅仅依赖于两个机器人。如果存在 i,j 使得 $1\leqslant i,j\leqslant m$ 以及 $\mathcal{A}^i(q^i)\cap\mathcal{A}^j(q^j)\neq\phi$，而不管其他 $m-2$ 个机器人的位形，那么一个点 $x=(q^1,\cdots,q^m)$ 在 X_{obs} 中。对于某些解耦方法，这甚至暗示 X_{obs} 的特征可以用二维投影来刻画，如图 7.7 所示。

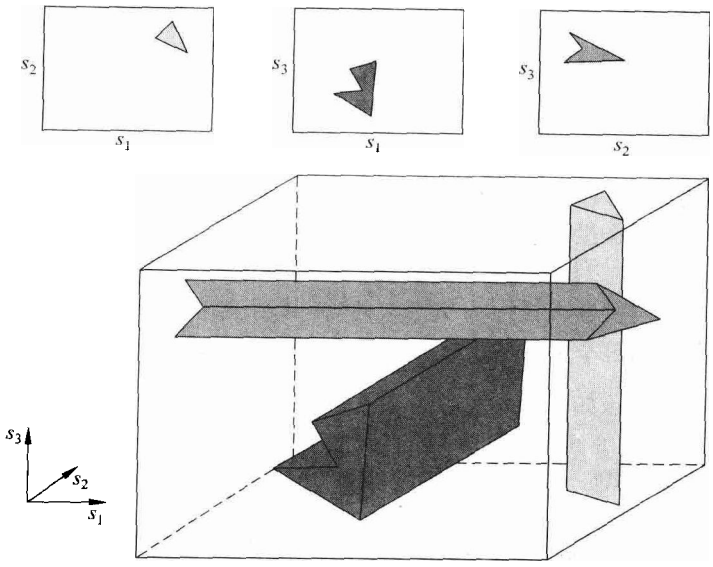


图 7.7　在协调 m 个机器人时出现的障碍物总是圆柱形的；所有 $\frac{1}{2}m(m-1)$ 个轴向二维投影的集合能够完全刻画 X_{obs} 的特征

（3）如果最优性很重要，那么为多机器人的情况产生一个唯一集合。它不是一个标准最优问题，因为标准最优问题需要对每个机器人的性能进行优化。无法在不损失某些信息的情况下将这些目标组合为一个单个优化问题。Pareto 优化对多机器人运动规划来说非常合适，这将在第 7.7.2 节进行阐述。

装配规划　多机器人运动规划的一个重要的派生是装配规划；回忆第 1.2 节中装配规划在应用中的重要性。在自动化制造中，许多复杂的结构从单个组件一步一步地装配起来。对机器人来说将这些组件一个一个地插入到相应位置是很方便的（见第 7.3.2 节）。假设有一族组件，每个组件都作为一个机器人，如图 7.8（a）所示。目标是将这些组件装配成一个有机的整体，如图 7.8（b）所示。从目标位形开始问题逐渐得到解决，目标位形受到严格约束，并且由内向外发展。问题的表述可以允许组件发生接触，但是它们的内部不能重

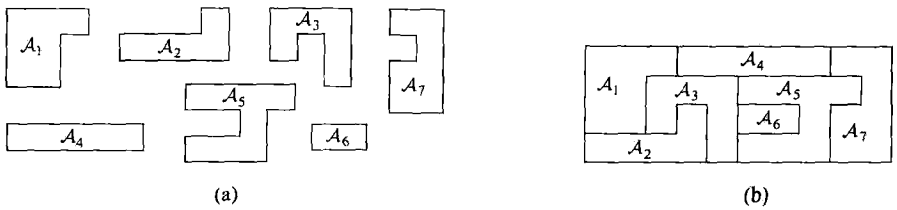


图 7.8　（a）用来定义装配规划问题的组件的集合；（b）装配规划涉及确定装配的运动序列，这里所显示的物体是从组件装配而来

叠。一般情况下，有任意多个组件的装配规划问题是 NP 复杂度的。已经为多个特殊情形研究了一些多项式时间算法。对于直线路径序列可以删除一些组件这种情况，文献[973,974]给出了一个多项式时间算法。

7.2.2 解耦规划

解耦方法首先为机器人设计运动，同时忽略机器人-机器人的相互作用。当考虑这些相互作用时，每个机器人可用的选择已经被所设计的运动所约束。如果出现问题，这些方法一般情况下是不可逆的。因此就丧失了完备性。尽管如此，解耦方法是非常实用的，在某些情况下，可以恢复完备性。

优先规划 一种很直接的解耦方法是用优先级将机器人进行排序，对优先级高的机器人首先进行规划[322,95]。对低优先级的机器人进行规划时将高优先级的机器人当作可移动的障碍物来对待。假设机器人排序为 $\mathcal{A}^1,\cdots,\mathcal{A}^m$，其中 $\mathcal{A}^1$ 的优先级最高。

假设对从 1 到 n 的每个 i 已经计算了无碰撞的路径 $\tau_i:[0,1]\to\mathcal{C}^i_{free}$。从直觉上讲，优先规划方法进行的过程应该如下所示。

基础步骤： 采用第 5 章和第 6 章的任何运动规划算法为 $\mathcal{A}^1$ 计算一条无碰撞的路径 $\tau_1:[0,1]\to\mathcal{C}^1_{free}$。为 τ_1 计算一个定时函数 σ_1 来产生 $\phi_1=\tau_1\circ\sigma_1:T\to\mathcal{C}^1_{free}$。

归纳步骤： 假设已经为 $\mathcal{A}^1,\cdots,\mathcal{A}^{i-1}$ 计算了 $\phi_1,\cdots,\phi_{i-1}$，并且这些函数能够避免前 $i-1$ 个机器人之间的碰撞。将前 $i-1$ 个机器人表述为在 $\mathcal{W}$ 中运动的障碍物。对每个 $t\in T$ 以及 $j\in\{1,\cdots,i-1\}$，$\mathcal{A}^j$ 的位形 q^j 是 $\phi_j(t)$。这产生 $\mathcal{A}^j\left(\phi_j(t)\right)\subset\mathcal{W}$，它可以作为障碍物 $\mathcal{O}(t)$ 的一个子集。采用第 7.1 节的任何时变运动规划方法设计一条路径 τ_i 和定时函数 σ_i，形成 $\phi_i=\tau_i\circ\sigma_i$。

尽管在许多环境下它是实用的，但是图 7.9 阐述了完备性是如何丧失的。

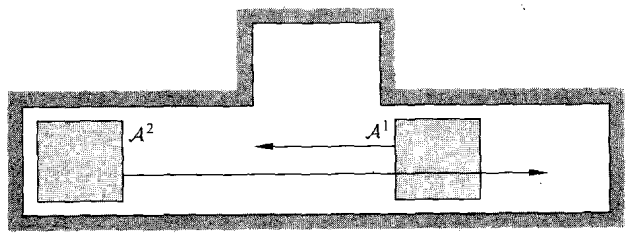


图 7.9 如果 $\mathcal{A}^1$ 忽略 $\mathcal{A}^2$，那么当用有优先规划方法时就会丧失完备性；这个例子在一般情况下是有解的，但是优先规划不能找到它

优先规划的一个特殊情形是在第一阶段设计所有路径 $\tau_1,\tau_2,\cdots,\tau_m$，然后将归纳步骤表述为一个速度调节问题。这产生了容易解决的一个二维规划问题序列。但是代价是昂贵的，因为选择受到更多的约束。如果能够仔细处理机器人之间的协调问题，那么可以对所有机器人预先规划路径和路线图，这种思路可以产生一个强有力的方法。这是下一步的课题。

固定路径的协调 假设每个机器人 $\mathcal{A}^i$ 被约束到一条路径 $\tau_i:[0,1]\to\mathcal{C}^i_{free}$，该路径可以用任何普通的运动规划方法来计算。对于 m 个机器人，定义一个 m 维状态空间，它使机器人沿着它们各自的路径运动，互相不发生碰撞[749]，这样的状态空间称为协调空间

(coordination space)。在协调空间中一个重要的性质是时间是隐含表示的。一个算法必须在协调空间中计算一条路径，并能够很容易地提取时间。

对于 m 个机器人，协调空间 X 定义为 m 维单位方体 $X=[0,1]^m$。图 7.7 描述了一个 $m=3$ 的例子。X 的第 i 个坐标表示路径 τ_i 的定义域 $S_i=[0,1]$。令 s_i 表示 S_i 中的一个点（也是 x 的第 i 个分量）。一个状态 $x\in X$ 指示每个机器人的位形。对每个 i，位形 $q^i\in C^i$ 由 $q^i=\tau_i(s_i)$ 给出。在状态 $(0,\cdots,0)\in X$，每个机器人处在它的初始位形 $q_I^i=\tau_i(0)$，在状态 $(1,\cdots,1)\in X$，每个机器人处在它的目标位形 $q_G^i=\tau_i(1)$。任何连续路径 $h:[0,1]\to X$，其中 $h(0)=(0,\cdots,0)$ 和 $h(1)=(1,\cdots,1)$，将机器人运动到它们的目标位形。路径 h 不一定是单调的，这与优先规划不同。

到目前为止我们一直都忽视了一个重要的问题。是什么妨碍我们在 $[0,1]^m$ 的对角之间将 h 设计成一条直线路径？我们还没有考虑机器人之间的碰撞。这形成了一个障碍区域 X_{obs}，这个障碍区域在设计路径时必须加以避开。因此任务是设计 $h:[0,1]\to X_{free}$，其中 $X_{free}=X\setminus X_{obs}$。

X_{obs} 的定义与式（7.8）和式（7.10）非常相似，除了这里的状态空间的维数非常小以外。每个 q^i 用一个参数代替。柱状结构依然得到保持，如图 7.7 所示。X_{obs} 的每个柱是：

$$X_{obs}^{ij}=\left\{(s_1,\cdots,s_m)\in X \,\middle|\, \mathcal{A}^i(\tau_i(s_i))\cap\mathcal{A}^j(\tau_j(s_j))\neq\phi\right\} \tag{7.11}$$

它们组合产生：

$$X_{obs}=\bigcup_{ij,i\neq j} X_{obs}^{ij} \tag{7.12}$$

标准运动规划算法可以应用于协调空间，因为在 h 上没有单调性的要求。如果（1）$\mathcal{W}=\mathbb{R}^2$；（2）$m=2$（两个机器人）；（3）障碍物和机器人都是多边形的；（4）路径 τ_i 是分段线性的，那么 X_{obs} 是 X 中的一个多边形区域。这使得在明确构造 X_{obs} 的表示之后就可以直接应用第 6.2 节中针对一个多边形 C_{obs} 的方法。对于 $m>2$，可以应用第 6.3.3 节中为 $m=3$ 给出的垂直胞腔剖分的多维版本。对于一般协调问题，可以应用圆柱代数剖分或者 Canny 路线图算法。对于在 $\mathcal{W}=\mathbb{R}^2$ 中进行平移或者沿圆形路径运动的机器人问题，基于采用特殊的碰撞检测方法准确确定 X_{obs}，文献[887]给出了一个分辨率完备的规划算法。

对于非常有挑战性的协调问题，基于采样的解法可能产生实用的解。其中可能最简单的一种解法是在 X 上放置一个栅格，并且适应性修改经典搜索算法，如第 5.4.2 节所示[609,749]。其他方法包括采用第 5.5 节的 RDT；或者如果应用多疑问架构，那么第 5.6 节的基于采样的路线图方法是合适的。第 5.3.4 节中介绍的验证路径段的方法很容易转变为针对处理协调空间的情况。

因此到目前一直都忽略了机器人的特定速度。为便于解释，考虑 $m=2$ 的情况。在 X 中垂直运动或者水平运动保持一个机器人固定，而其他机器人以某最大速度运动。在 X 中对角运动使两个机器人运动，它们之间的相对速度依赖于路径的斜率。为了仔细控制这些速度，有必要用距离来参数化路径。在这种情况下，X 的每个轴表示运动的距离，而不是[0,1]。

固定路线图的协调　如果独立设计路径，固定路径协调方法仍然不可能解决图 7.9 中的问题。幸运的是，固定路径协调可以扩展到使每个机器人在一个路线图或者拓扑图上运

动。这产生了每个机器人仅有一维的一个协调空间，所产生的规划算法非常接近于完备的，好像每个机器人利用了一个有许多可选路径的路线图。自动导引车辆（AGV）也是研究这个问题的动机，这种车辆经常在工厂的预先设定路径的网络线上运动。在这种情况下，协调机器人就是规划问题，这与表述 7.2 所进行的简化有所不同。

得到表述 7.2 的完备性的一个方法是设计独立路线图使得每个机器人有其自己的“车库”（garage）位形。一个位形 q^i 成为 $\mathcal{A}^i$ 的车库的条件是：（1）在 q^i 位形下，不可能与任何其他机器人发生碰撞（例如在第 i 个坐标是 q^i 的所有状态下，没有碰撞发生）；（2）对于从 x_I 开始的 $\mathcal{A}^i$ 来说 q^i 总是可达的。如果每个机器人有一个路线图和车库，并且如果规划方法对 X 来说是完备的，那么整个规划算法就是完备的。如果 X 中的规划方法采用某个较弱的完备性概念，那么完备性也得到保持。例如 X 的一个分辨率完备的规划器为表述 7.2 中的问题产生一个分辨率完备的方法。

方体复形 当每个机器人有多个路径时如何表示协调空间？这样得到一个方体复形（cube compex），它是奇异复形的一种特殊形式。m 个固定路径的协调空间可以认为是一个奇异 m-复形。例如图 7.7 中的问题可以被认为是一个奇异 3-复形 $[0,1]^3 \to X$。在第 6.3.1 节中，一个 k-复形的定义域用一个 k 维球 B^k 来定义；但是也可以用一个方体 $[0,1]^k$ 来定义，因为 B^k 和 $[0,1]^k$ 是同胚的。

对于一个拓扑空间 X，令一个 k-维方体（也是一个奇异 k-复形）$\square_k$ 为一个连续映射 $\sigma:[0,1]^k \to X$。通过将不同维数的 k-维方体连接在一起得到一个方体复形。对 $k \geqslant 1$，每个 k-维方体有 $2k$ 个面，每个面都是一个用如下方法得到的 $(k-1)$-维方体。令 $(s_1,\cdots,s_k)$ 表示 $[0,1]^k$ 中的一个点。对每个 $i \in \{1,\cdots,k\}$，设置 $s_i = 0$ 可以得到一个面，设置 $s_i = 1$ 可以得到另外一个面。

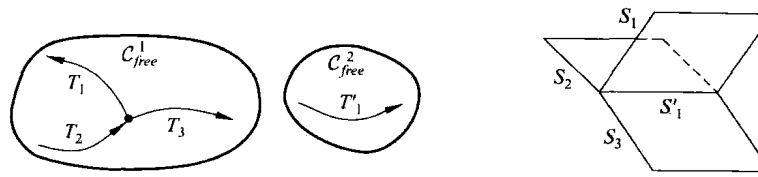
方体必须相互匹配，这与要求单纯复形的单形之间互相匹配的方法相同。为了成为一个方体复形，$\mathcal{K}$ 必须是满足如下要求的单形的一个族：

（1）一个方体 $\square_k \in \mathcal{K}$ 的任何面 $\square_{k-1}$ 也在 $\mathcal{K}$ 中。

（2）任何两个 k-维方体 $\square_k, \square'_k \in \mathcal{K}$ 的像的交集要么是空的，要么存在某方体 $\square_i \in \mathcal{K}$，$i < k$，它是 $\square_k$ 和 $\square'_k$ 的公共面。

令 $\mathcal{G}_i$ 表示机器人 $\mathcal{A}^i$ 的一个拓扑图（它也可能是一个路线图）。图的边是 $\tau:[0,1] \to \mathcal{C}^i_{free}$ 形式的路径。在介绍所产生的复形的正式定义之前，先考虑图 7.10（a），在该图中，$\mathcal{A}^1$ 沿着在一个“T”交叉点处连接在一起的三条路径运动，$\mathcal{A}^2$ 仅沿着一条路径运动。在这种情况下，三个二维固定路径协调空间沿着一条公共边连接起来，如图 7.10（b）所示。所产生的方体复形用三个两维方体（例如正方形）、一个一维方体（例如线段）以及八个 0-维方体（例如角点）来定义。

现在假设更一般的情况，有两个机器人 $\mathcal{A}^1$ 和 $\mathcal{A}^2$，它们的拓扑图分别为 $\mathcal{G}_1(V_1,E_1)$ 和 $\mathcal{G}_2(V_2,E_2)$。假设 $\mathcal{G}_1$ 和 $\mathcal{G}_2$ 分别有 n_1 和 n_2 条边。如下可以这样得到一个二维方体复形 $\mathcal{K}$。令 τ_i 表示 $\mathcal{G}_1$ 的第 i 条路径，令 σ_j 表示 $\mathcal{G}_2$ 的第 j 条路径。从每个图选择一条边，在 $\mathcal{K}$ 中都存在一个相应的二维方体（正方形）。因此有 $n_1 n_2$ 个 2-维方体，每个二维方体对应一对 (τ_i,σ_j)，$\tau_i \in E_1$，$\sigma_j \in E_2$。为 (v_i,σ_j) 对生成一维方体，$v_i \in V_1$，$\sigma_j \in E_2$，或者为 (τ_i,v_j) 对产生 1-



(a)　　(b)

图 7.10　（a）$\mathcal{A}^1$ 沿着 3 条路径运动，$\mathcal{A}^2$ 沿着 1 条路径运动；（b）相应的协调空间

维方体，$\tau_i \in E_1$，$v_j \in V_2$。为每个(v_i, v_j)对生成 0-维方体（角点），$v_i \in V_1$，$v_j \in V_2$。

如果有 m 个机器人，那么会产生一个 m-维方体复形。每个 m-维方体对应于唯一一个路径组合，每个机器人有一个路径。$(m-1)$-维方体是 m-维方体的面。过程继续直到到达 0-维方体。

在方体复形上的规划　第 5 章和第 6 章中的任何规划方法做适当修改后都可以在这里应用，但是因为 X 是一个复形，所以方法稍微复杂些。为了使用基于采样的方法，在 X 上应该生成一个稠密序列。例如，如果采用随机采样，那么可以随机选择一个 m-维方体，然后选择方体中的一个随机点。局部规划方法（LPM）必须考虑方体复形的连通性，该方法要求识别什么时候拓扑图发生分叉。组合方法也必须考虑这些连通性。例如可以应用扫描技术来产生一个垂直胞腔剖分，但是扫描线（或者扫描平面）必须扫描复形的各种 m-胞腔。

7.3　离散与连续空间的混合空间

一些重要的应用涉及离散和连续变量的组合。这样产生的状态空间是 C-空间和一个称为模式空间的有限集的一个笛卡儿积。所产生的空间可以形象化为具有可以用模式来进行索引的 C-空间的层，如图 7.11 所示。这里给出的主要应用是操作规划；也存在其他的应用，特别是当诸如动态性和不确定性等其他复杂性加入到问题中的时候。这部分内容是受到控制论中混合系统（hybrid systems）的启发[411]，在那里通常建立与模式有关的动态模型。本节主要关心的是所允许的机器人位形和（或）障碍物都依赖于模式。

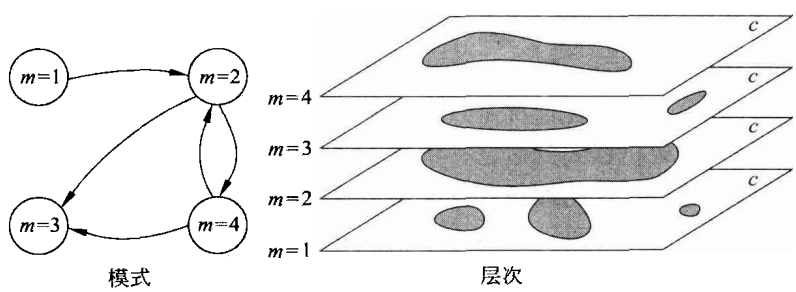


图 7.11　可将一个混合状态空间想象成有层次的 C-空间，层次是由模式来索引的

7.3.1　混合系统的架构

如图 7.11 所示，一个混合系统涉及离散空间和连续空间的相互作用。首先给出正式模

型，然后进行说明。这个表述可以认为是一个表述 2.1 的离散可行规划和表述 4.1 的基本规划的一个组合。

表述 7.3（混合系统运动规划）

1. 包括表述 4.1 中的 $\mathcal{W}$ 和 $\mathcal{C}$。

2. 一个非空模式空间 M 是模式的一个有限集合或者可数无限集合。

3. 对于每个 $m \in M$，有一个半代数障碍区域 $\mathcal{O}(t)$。

4. 对于每个 $m \in M$，有一个半代数机器人 $\mathcal{A}(m)$。它可以是刚体机器人或者连杆族。假设 $\mathcal{C}$-空间与模型无关；只有机器人的几何形状依赖于模型。变换到位形 q 的机器人用 $\mathcal{A}(q,m)$ 表示。

5. 一个状态空间 X 定义为笛卡儿积 $X = \mathcal{C} \times M$。一个状态表示为 $x = (q,m)$，其中 $q \in \mathcal{C}$ 和 $m \in M$。令：

$$X_{obs} = \{(q,m) \in X \mid \mathcal{A}(q,m) \cap \mathcal{O}(m) \neq \phi\} \tag{7.13}$$
$$X_{free} = X \setminus X_{obs}$$

6. 对于每个状态 $x \in X$，存在一个有限行动空间 $U(x)$。令 U 表示所有可能的行动的集合（所有 $x \in X$ 上的 $U(x)$ 的并集）。

7. 存在一个模式转换函数 f_m，它为每个 $x \in X$ 和 $u \in U(x)$ 产生一个模式 $f_m(x,u) \in M$。假设 f_m 的定义不产生快速转换（模式在固定时间内进行振荡）条件。即如果 q 确定下来，模式可以立即改变。然后它保持固定，仅在 q 发生改变时才改变。

8. 状态转移函数 f，它是从 f_m 通过改变模式和保持位形固定来产生的。因此 $f(x,u) = (q, f_m(x,u))$。

9. 一个位形 $x_I \in X_{free}$ 被指定为初始状态。

10. 一个集合 $X_G \in X_{free}$ 被指定为目标区域。定义一个区域，而不是一个点，这有利于描述不依赖于最后模式的目标位形。

11. 算法必须计算一条(连续)路径 $\tau : [0,1] \to X_{free}$ 以及一个行动轨迹 $\sigma : [0,1] \to U$ 使得 $\tau(0) = x_I$ 和 $\tau(1) \in x_G$ 或者算法正确地报告这种路径和行动轨迹的组合不存在。

根据问题的不同，障碍区域和机器人可能与模式有关，也可能与模式无关。稍后将给出相应的例子。模式的变化依赖于机器人采取的行动。在大多数状态下，我们通常假设有一个“不做任何事情”的行动，它使得模式不发生变化。在确定状态下，机器人可以选择一个行动来改变模式。一个退化情况是只有一个行动可用。这意味着机器人对那个状态的模式没有进行控制。如果机器人到达这种状态，那么模式的改变将不可避免地发生。

求解方法会更加复杂，因为路径和行动轨迹都需要确定。仅确定一条路径是不够的，因为知道采取什么行动才能正确地进行模式转换。因此 σ 指示行动在什么时候发生。注意到 τ 和 σ 是紧密耦合的；不能简单地将任何 σ 与一条路径 τ 联系起来；它必须对应于生成 τ 所要求的行动。

例 7.2（portiernia 的力量） 本例中，一个机器人 $\mathcal{A}$ 建模为一个在 $\mathcal{W} = \mathbb{R}^2$ 中平移的正方形。因此 $\mathcal{C} = \mathbb{R}^2$。$\mathcal{W}$ 中的障碍区域与模式相关，因为它有两个门，如图 7.12（a）所示

的编号为“1”和“2”。在左上角是 portiernia[1]，它给机器人提供一把钥匙，如果机器人的位形如图 7.12（b）所示。portiernia 每次仅向机器人提供一把钥匙，该钥匙开“1”号门或者“2”号门。机器人能够重新回到 portiernia 归还钥匙。如图 7.12（c）和 7.12（d）所示，机器人能够接触一个门并打开它，只要它握有正确的钥匙。

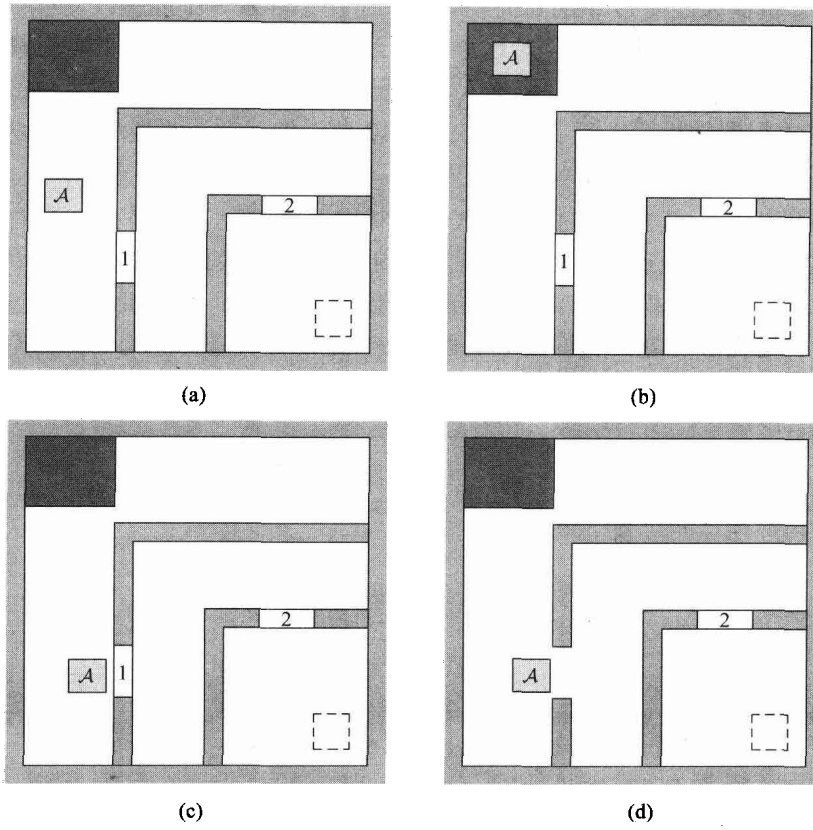


图 7.12　在左上角（portiernia 处），机器人能够捡起和放下打开其中一扇门的钥匙；如果机器人握有正确的钥匙，并接触到一扇门，那么门就会被打开

模式的集合 M 需要对机器人掌握哪一把钥匙进行编码，也必须对门的状态进行编码。机器人可能有：（1）开“1”号门的钥匙；（2）开“2”号门的钥匙；（3）没有钥匙。门可能的状态：（1）两个门都开着；（2）“1”号门开着，“2”号门关着；（3）“1”号门关着，“2”号门开着；（4）两个门都关着。综合考虑钥匙和门就产生了 12 种可能模式。

如果机器人处在 portiernia 位形，如图 7.12（b）所示，那么它的可行的行动对应于不同的捡起和放下钥匙的方式。例如，如果机器人握有“1”号门的钥匙，它可以放下该钥匙而捡起“2”号门的钥匙。这就改变了模式，但是当应用 f 时，门的状态和机器人的位形必须保持不变。机器人能够改变模式的其他位置是当它接触“1”号门或者“2”号门的时候。仅当机器人握有正确的钥匙时，模式才能改变。在所有其他位形下，机器人只有一个单独的行动（例如：不选择），该行动保持模式不变。

1　这是波兰人在某些公共设施处存放钥匙的地方。

任务是到达右下方点画线所示的位形。解决问题通过：（1）在 portiernia 处捡起“1”号门的钥匙；（2）打开“1”号门；（3）在 portiernia 处交还“1”号门的钥匙，得到“2”号门的钥匙；（4）进入最里边的房间到达目标位形。作为最后一个条件，我们可能想要求机器人将钥匙归还给 portiernia。 ■

例 7.2 允许机器人改变$\mathcal{O}$中的障碍物。下一个例子涉及一个能够改变形状的机器人。这是一个可重配置机器人的例子。对这种机器人的研究已经是一个非常普遍的研究课题[211,387,555,990]；在该研究领域中机器人重新配置的可能性远比这里所考虑的例子复杂。

例 7.3（可重配置的机器人） 为了解决图 7.13 所示的问题，机器人必须改变它的形状。有两种可能的形状，它们直接对应于两种模式：拉伸和压缩。图 7.14 显示了$\mathcal{C}_{free}(m)$是如何出现的。假设机器人从左边的拉伸模式下的状态开始运动，它必须到达右边最后一个房间。解决这个问题必须通过：（1）将机器人重构为压缩模式；（2）通过中间的走廊；（3）将机器人重构为拉伸模式；（4）穿过走廊到达最右边的房间。机器人通过重构来改变模式。为了使问题更加有趣，我们要求机器人在特定的位置进行重构（例如有足够空隙的地方，或者在有其他机器人辅助的地方）。

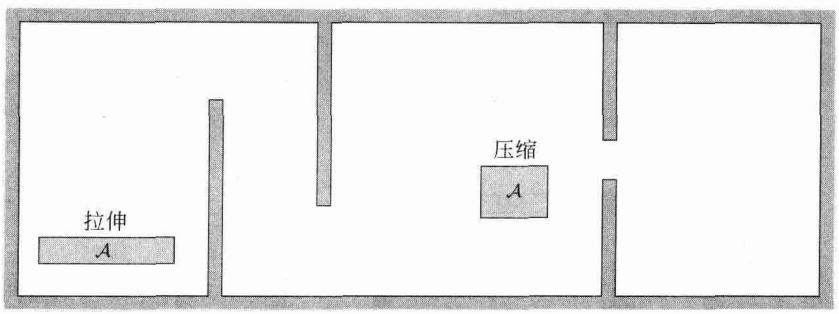


图 7.13 机器人必须变形才能解决问题；有两种模式：拉伸和压缩

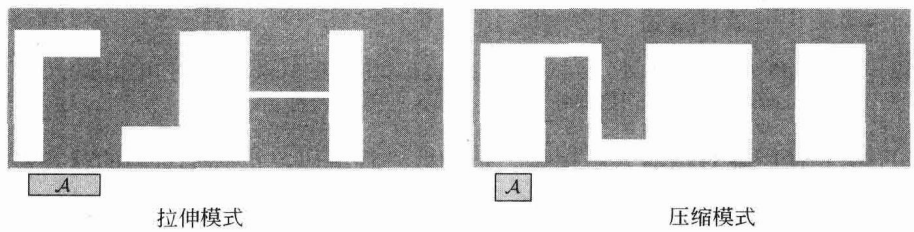


图 7.14 当机器人变形时，$\mathcal{C}_{free}(m)$发生变化，这样才能使问题得以解决

这里提到的例子才触及到能够定义的混合运动规划问题。例如，在构造视频游戏人物和数字演员时，会出现这种问题。为了解决这些问题，可以采用标准运动规划算法，如果给它们提供关于如何改变模式的信息。X中能够改变模式的地方可以表示为目标的子目标。除了试图直接到达目标以外，规划工作应该将注意力集中于改变模式上。应用基于采样的方法要求在X上定义一个度量，该度量考虑到模式和位形的改变。可以表述各种各样的混合问题，范围从那些实际上不可能解决的问题到对标准运动规划的扩展。一般情况下，混合运动规划模型对表述一个层次方法非常有用，如第 1.4 节所描述的那样。我们将会介绍适合这个模型的一类特别有趣的问题，有许多成功的算法可以解决这类问题。

7.3.2　操作规划

本节概要介绍操作规划；其中的概念主要来源于文献[16,17]。回到例 7.2，假设机器人必须搬运一把钥匙，而这把钥匙太大以至于改变了 $\mathcal{C}_{free}$ 的连通性。对于操作规划问题，机器人被称为机械臂，它和部件相互作用。在某些位形中，它能够抓住部件并运动到环境中的其他位置。操作的任务通常要求将部件移动到 $\mathcal{W}$ 中的特殊位置，而不关心机械臂是如何完成任务的。这里考虑的模型大大简化了抓握、稳定性、摩擦、力学以及不确定性问题，而仅关注于几何方面（一些问题将在第 12.5 节中进行讨论）。对于完整介绍操作规划的其他方面的内容，参见文献[684]；也可参见第 13.1.3 节和第 12.5 节。

可允许的位形　假设采用表述 4.1 中的 $\mathcal{W}$、$\mathcal{O}$ 和 $\mathcal{A}$。对于操作规划，$\mathcal{A}$ 被称为机械臂，令 $\mathcal{C}^a$ 表示机械臂位形空间。令 $\mathcal{P}$ 表示一个部件，部件是一个用几何本原进行建模的刚体，如第 3.1 节所描述。假设允许 $\mathcal{P}$ 进行刚体变换，因此 $\mathcal{P}$ 有其自己的位形空间 $\mathcal{C}^p = SE(2)$ 或者 $\mathcal{C}^p = SE(3)$。令 $q^p \in \mathcal{C}^p$ 表示部件的一个位形。变换后的部件模型用 $\mathcal{P}(q^p)$ 表示。

组合的位形空间 $\mathcal{C}$ 定义为笛卡儿积：

$$\mathcal{C} = \mathcal{C}^a \times \mathcal{C}^p \tag{7.14}$$

其中每个位形 $q \in \mathcal{C}$ 为 $q = (q^a, q^p)$。第一步是删除所有必须避免的位形。图 7.15 中的部件显示了这些集合的例子。机械臂与障碍物发生碰撞的位形是：

$$\mathcal{C}^a_{obs} = \left\{ (q^a, q^p) \in \mathcal{C} \middle| \mathcal{A}(q^a) \cap \mathcal{O} \neq \phi \right\} \tag{7.15}$$

下一步是删除部件与障碍物发生碰撞的位形。可以允许部件与障碍物发生“接触”。例如可以对一个放在桌子上的部件进行建模。因此，令：

$$\mathcal{C}^p_{obs} = \left\{ (q^a, q^p) \in \mathcal{C} \middle| \text{int}(\mathcal{P}(q^p)) \cap \mathcal{O} \neq \phi \right\} \tag{7.16}$$

表示部件的内部与 $\mathcal{O}$ 相交的开集合。当然如果部件穿入 $\mathcal{O}$，那么这种位形是应该避免的。

考虑 $\mathcal{C} \setminus (\mathcal{C}^a_{obs} \cup \mathcal{C}^p_{obs})$。其余的位形就能够保证机器人和部件不会与 $\mathcal{O}$ 发生碰撞。下面考虑 $\mathcal{A}$ 和 $\mathcal{P}$ 的相互作用。必须允许机械臂接触部件，但是不允许穿入。因此，令：

$$\mathcal{C}^{ap}_{obs} = \left\{ (q^a, q^p) \in \mathcal{C} \middle| \mathcal{A}\left(q^a\right) \cap \text{int}(\mathcal{P}(q^p)) \neq \phi \right\} \tag{7.17}$$

删除所有这些坏的位形就产生了：

$$\mathcal{C}_{adm} = \mathcal{C} \setminus \left(\mathcal{C}^a_{obs} \cup \mathcal{C}^p_{obs} \cup \mathcal{C}^{ap}_{obs} \right) \tag{7.18}$$

它被称为可允许的位形集合。

稳定的位形和抓握位形　在操作规划问题中使用 $\mathcal{C}_{adm}$ 的两个重要的子集，如图 7.15 所示。令 $\mathcal{C}^p_{sta}$ 表示稳定部件位形，它表示在这些位形中，部件处在静止状态，没有受到来自于机械臂的外力影响。这意味着部件不能飘浮在空中。它也不能处在可能下落的位形中。稳定位形取决于诸如部件的几何形状、摩擦、质量分布等因素。这里不考虑这些问题。令 $\mathcal{C}_{sta} \subseteq \mathcal{C}_{adm}$ 为相应的稳定位形，它被定义为：

$$\mathcal{C}_{sta} = \left\{ (q^a, q^p) \in \mathcal{C}_{adm} \middle| q^p \in \mathcal{C}^p_{sta} \right\} \tag{7.19}$$

$\mathcal{C}_{adm}$ 的另外一个重要子集是使机器人能够抓住部件的所有位形的集合（如果需要，也要能够搬运该部件）。这称为抓握位形，用 $\mathcal{C}_{gr} \subseteq \mathcal{C}_{adm}$ 表示。对于每个位形 $(q^a, q^p) \in \mathcal{C}_{gr}$，机械臂

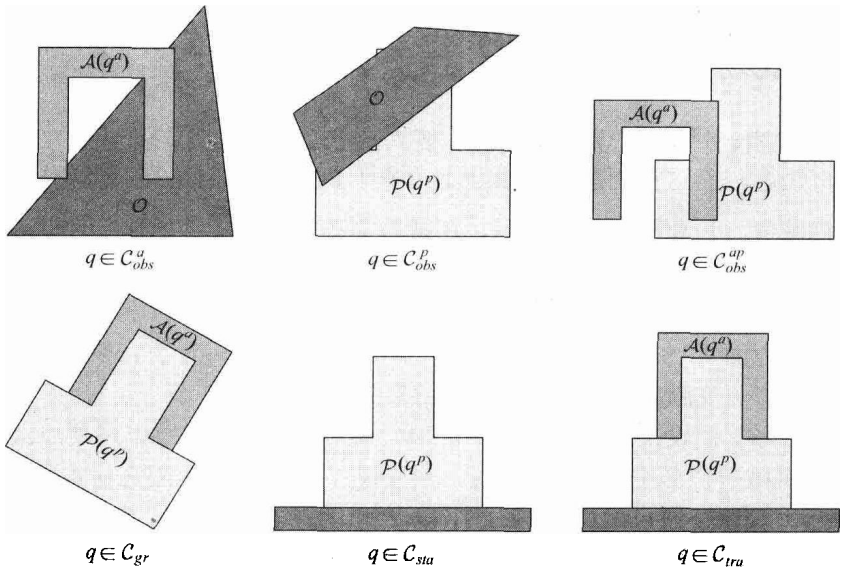


图 7.15 操作规划中 $\mathcal{C}$ 的几种重要的子集

接触部件。这意味着 $\mathcal{A}(q^a) \cap \mathcal{P}(q^p) \neq \phi$（仍然不允许穿入，因为 $\mathcal{C}_{gr} \subseteq \mathcal{C}_{adm}$）。一般情况下，$\mathcal{A}(q^a)$ 和 $\mathcal{P}(q^p)$ 发生接触的一些位形不一定在 $\mathcal{C}_{gr}$ 中。一个点落在 $\mathcal{C}_{gr}$ 中的条件取决于机械臂、部件以及它们之间接触面的特征。例如一个典型的机械臂仅通过接触物体块的一个角是不可能抓起该物体块的。更详细的内容这里没有给出；更多关于抓握的信息，请参见文献[684]。

我们必须确保 $x \in \mathcal{C}_{sta}$ 或者 $x \in \mathcal{C}_{gr}$。因此令 $\mathcal{C}_{free} = \mathcal{C}_{sta} \cup \mathcal{C}_{gr}$ 表示 $\mathcal{C}_{adm}$ 的子集，它对于操作规划来说是可允许的。

模式空间 M 包含两种模式，它们称为过渡模式（transit）和转移模式（transfer）。在过渡模式下，机械臂不能搬运部件，这要求 $q \in \mathcal{C}_{sta}$。在转移模式下，机械臂搬运部件，这要求 $q \in \mathcal{C}_{gr}$。所以只有 $q \in \mathcal{C}_{sta} \cap \mathcal{C}_{gr}$，模式才能发生改变。因此当机械臂处在这些位形时，它有两个可用的行动。而在其他位形下，模式保持不变。为简单起见，令 $\mathcal{C}_{tra} = \mathcal{C}_{sta} \cap \mathcal{C}_{gr}$ 表示**转换位形**（transition configurations）的集合，它们是模式发生改变的位置。

使用第 7.3.1 节的架构，模式空间 M 和 $\mathcal{C}$ -空间 $\mathcal{C}$ 组合产生状态空间 $X = \mathcal{C} \times M$。因为只有两种模式，所以仅有 $\mathcal{C}$ 的两个副本，每种模式对应一个。通过忽略模式，基于状态的集合 X_{free}、X_{tra}、X_{sta} 以及 X_{gr} 可以直接从 $\mathcal{C}_{free}$、$\mathcal{C}_{tra}$、$\mathcal{C}_{sta}$、$\mathcal{C}_{gr}$ 得到。例如：

$$X_{tra} = \{(q, m) \in X \mid q \in \mathcal{C}_{tra}\} \tag{7.20}$$

集合 X_{free}、X_{sta} 以及 X_{gr} 的定义与此相似。

现在定义任务。指定一个初始部件位形 $q^p_{init} \in \mathcal{C}_{sta}$ 和一个目标部件位形 $q^p_{goal} \in \mathcal{C}_{sta}$。计算一条路径 $\tau : [0,1] \rightarrow X_{free}$ 使得 $\tau(0) = q^p_{init}$ 和 $\tau(1) = q^p_{goal}$，并且必须指定行动轨迹 $\sigma : [0,1] \rightarrow U$ 来指示当 $\tau(s) \in X_{tra}$ 时相应的模式发生变化。解可以认为是**过渡路径**（transit path）和**转移路径**（transfer path）的一个交替序列，路径的名字来源于模式。图 7.16 描述了这种情况。

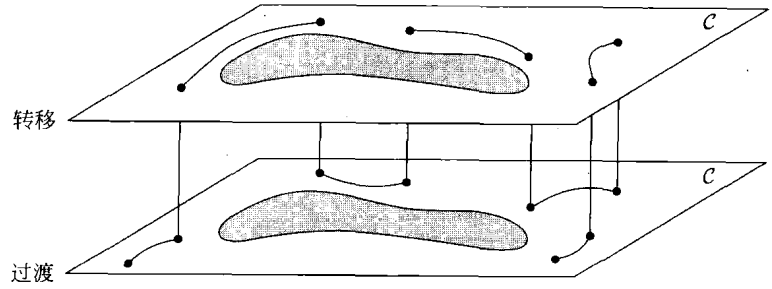


图 7.16　一个操作规划问题的解在 X 的两个层之间交替变化；仅在 $x \in X_{tra}$ 模式时发生转换

操作图　通过形成一个操作图 $\mathcal{G}_m$ [16,17]，操作规划问题可以逐步得到解决。令 X_{tra} 的一个连通分支表示通过忽略模式而提升到状态空间的 $\mathcal{C}_{tra}$ 的任何连通分支。$\mathcal{C}_{tra}$ 的连通分支有两个副本，每种模式对应一个。对于 X_{tra} 的每个连通分支，在 $\mathcal{G}_m$ 中存在一个顶点。为每条过渡路径和转移路径定义一条边，这条边连接 X_{tra} 的两个连通分支。操作规划的一般方法如下：

（1）计算 X_{tra} 的连通分支来产生 $\mathcal{G}_m$ 的顶点。

（2）对 $\mathcal{G}_m$ 的每对顶点应用普通的运动规划方法来计算 $\mathcal{G}_m$ 的边。

（3）对 X_{tra} 的每个可能的顶点应用运动规划方法来连接初始状态和目标状态，这些顶点是指不需要进行模式转换就能够到达的顶点。

（4）为连接初始状态和目标状态的路径而搜索 $\mathcal{G}_m$。如果存在，那么提取相应的解作为过渡路径和转移路径的一个序列（这产生引起所需要的模式变化的行动 σ）。

这可以被认为是一个层次规划的例子，如第 1.4 节所述。

多个部件　操作规划的架构可以扩展到多个部件 $\mathcal{P}_1,\cdots,\mathcal{P}_k$。每个部件有其自己的 C-空间，所有部件的 C-空间和机械臂的 C-空间的笛卡儿积可以形成 $\mathcal{C}$。集合 $\mathcal{C}_{adm}$ 用相似的方法进行定义，但是现在除了考虑部件-机械臂、机械臂-障碍物、部件-障碍物之间的碰撞以外，还需要消除部件-部件之间的碰撞。$\mathcal{C}_{sta}$ 的定义要求所有部件都处在稳定位形中；允许部件相互堆积起来。$\mathcal{C}_{gr}$ 的定义要求当抓住一个部件时，所有其他部件都是稳定的。仍然有两种模式，这取决于机械臂是否正在抓握一个部件。模式转换仍发生在机器人处在 $\mathcal{C}_{tra} = \mathcal{C}_{sta} \cap \mathcal{C}_{gr}$ 时。任务涉及将每个部件从一个位形移动到另外一个位形。这仍可以通过定义一个操作图和所得到的过渡路径（没有部件移动）和转移路径（搬运一个部件，其他部件不动）的一个序列来完成。通过运动规划算法来解决的具有挑战性的操作问题如图 7.17 和图 7.18 所示。

还可能有其他形式的扩展。扩展到 k 个机器人将导致 2^k 种模式，每种模式指明某机器人是否正在抓握部件。多个机器人可以抓握同一个物体。另外一种可能是允许单个机器人抓握多个物体。

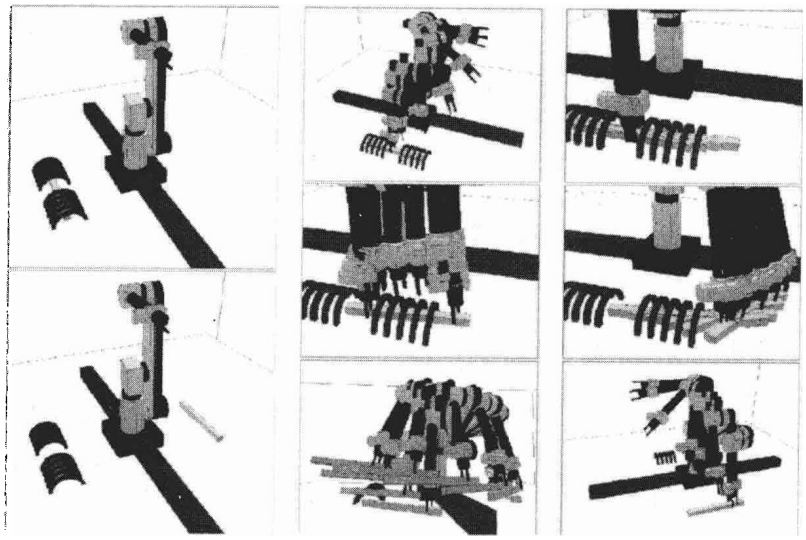


图 7.17 这个例子在文献[246]中是用操作规划架构、基于可视路线图规划器来解决的；它很具有挑战性，因为必须在不同地方再次抓起同一个部件

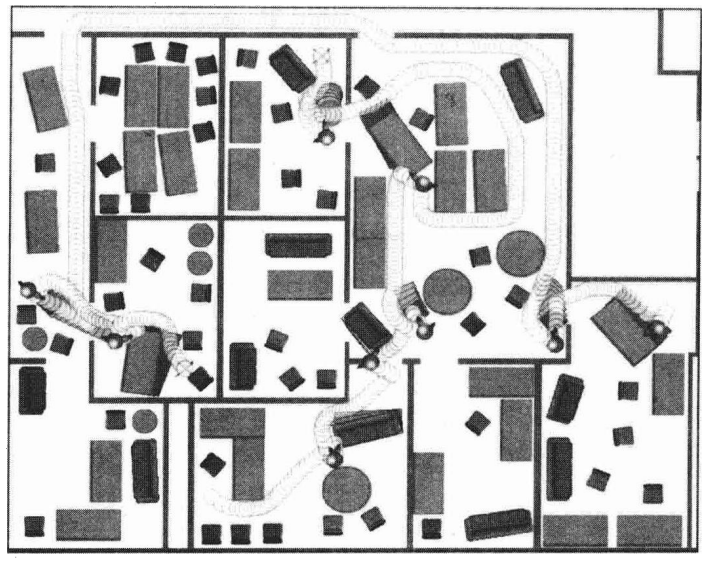


图 7.18 文献[916]解决了这个操作规划的例子，它涉及 90 件可移动的家具；其中一些必须被推开以解决该问题，图中显示了两个不同请求的路径

7.4 闭运动链的规划

本节继续讨论第 4.4 节遗留下来的问题。通过一些例子来定义和阐述，产生于保持运动封闭的 C 的子空间。这里的规划要求路径保持在无法进行参数表示的低维簇上。在保持这些约束的同时，一些重要的应用还要求运动规划。例如存在一个涉及多个机械臂抓握同一个物体的操作规划问题，这形成了一个如图 7.19 所示的闭环。存在一个环，那是因为两个机械臂固定在地面上，地面可以认为是一个连杆。电影和视频游戏中虚拟演员的开发也

涉及相关的操作问题。当人的肢体接触一个固定面时也会形成环（例如地面上的两条腿）。我们故意设计一类被称为平行机械臂的机器人，它们有内部的环[696]。例如考虑如图 7.20 所示的 Stewart-Gough 平台[409,915]。当通过球形关节与地面和平台 $\mathcal{A}^7$ 保持连接的同时，六个臂 $\mathcal{A}^1,\cdots,\mathcal{A}^6$ 的长度可以独立变化。每个臂实际上都可以被想象为用棱柱关节连接起来的二连杆。由于可变长度带来的总共六个自由度，平台实际上有全部六个自由度（因此为 $\mathcal{A}^7$ 可以得到 $SE(3)$ 中的某 6 维区域）。对 Stewart-Gough 平台或者基于平台的机器人（图 7.27 所示的机器人用到多个这种机械）的运动进行规划都要求处理一些必须得到保持的封闭约束。另外一个应用是计算生物学，在其中对分子的 C-空间的进行搜索，这些空间产生于有键的柔性封闭链的分子[247]。

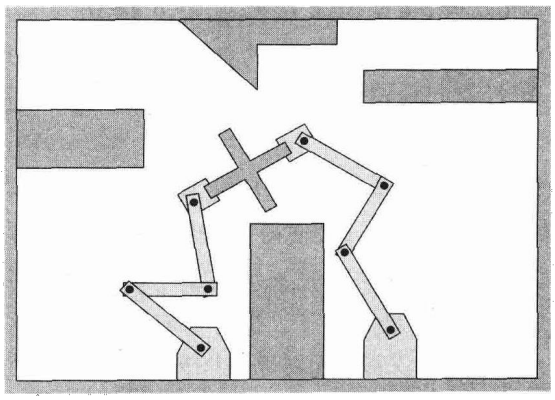


图 7.19　两个或者多个操作臂操作同一个物体就形成了闭运动链；每个黑点对应于一个旋转关节

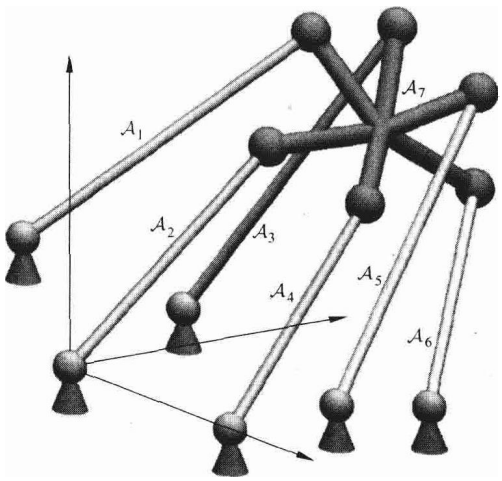


图 7.20　Stewart-Gough 平台（将 Frank Sottile 所做的图进行修改得到的）

7.4.1　运动规划算法的适应性修改

表述 4.1 的一般运动规划问题中所有组成部分包括 $\mathcal{W}$、$\mathcal{O}$、$\mathcal{A}_1,\cdots,\mathcal{A}_m$、$\mathcal{C}$、$q_I$ 和 q_G。假设机器人是可能连接成环的 r 个连杆的族。

这里假设 $\mathcal{C}=\mathbb{R}^n$。如果该假设没有得到满足，那么有两种方法来解决。第一种是将

$SO(2)$ 和 $SO(3)$ 分别表示为 $\mathbb{S}^1$ 和 $\mathbb{S}^3$，将圆和球方程作为这里所考虑的部分约束。这避免了拓扑问题。另外一种方法是不使用 $\mathbb{R}^n$，而是用相应维数的 C 的一种参数表示。为了在这种流形上进行积分，要求一个平滑结构，这种结构将在第 8.3.2 节进行介绍。但是在这里，$\mathbb{R}^n$ 上的向量积分就足够了。

封闭约束 第 4.4 节提出的封闭约束可以总结如下。有多项式 $f_1,\cdots,f_k$ 的一个集合 $\mathcal{P}$，多项式属于 $\mathbb{Q}[q_1,\cdots,q_n]$，它们表示机器人连杆上特定点的约束。第 4.4.3 节中详细介绍了这些参数的确定。就像以前提到的那样，在旋转情况下迫使点落在一个圆或者球上的多项式可能也包括在 $\mathcal{P}$ 中。

令 n 表示 $\mathcal{C}$ 的维数。封闭空间定义为：

$$\mathcal{C}_{clo}=\{q\in\mathcal{C}\,|\,\forall f_i\in\mathcal{P}, f_i(q_1,\cdots,q_n)=0\} \tag{7.21}$$

它是 $\mathcal{C}$ 的一个 m 维子空间，它对应于满足封闭约束的所有位形。必须考虑障碍集合。$\mathcal{C}_{obs}$ 和 $\mathcal{C}_{free}$ 由式（4.34）定义。可行空间被定义为 $\mathcal{C}_{fea}=\mathcal{C}_{clo}\cap\mathcal{C}_{free}$，它们是满足封闭约束且避免碰撞的位形。

运动规划问题是寻找一条路径：$\tau:[0,1]\to\mathcal{C}_{fea}$ 使得 $\tau(0)=q_I$ 和 $\tau(1)=q_G$。新的问题是没有对 $\mathcal{C}_{fea}$ 进行显式的参数表示，$m<n$ 会使得问题更加复杂（m 是 $\mathcal{C}_{clo}$ 的维数）。

组合方法 因为约束用多项式表示，那么第 6.4 节的计算代数几何方法能够解决带有闭运动链的一般运动规划问题就不足为奇了。也可以应用圆柱代数剖分或者 Canny 路线图算法。就像第 6.5.3 节所提到的，文献[78]对路线图算法进行适应性修改，对 $m<n$ 的问题来说算法被最优化了。

基于采样的方法 第 5 章的大多数方法并不容易进行适应性修改，那是因为它们要求在 $\mathcal{C}_{fea}$ 中进行采样，而 $\mathcal{C}_{fea}$ 不存在参数表示。如果随机选择 $\mathbb{R}^n$ 的边界区域中的点，那么得到 $\mathcal{C}_{fea}$ 上的点的概率为零。通过构造适合带封闭约束的问题的一个局部规划方法（LPM），有些递增采样和搜索方法可以进行适应性修改。基于采样的路线图方法要求一些采样点直接在 $\mathcal{C}_{fea}$ 上生成。第 7.4.2 节为特定类型的问题提出一些可以用来生成这种采样点的技术，这些技术促进了基于采样的规划器的发展，也提高了递增搜索算法的效率。在介绍这些技术之前，我们首先介绍一种方法，它产生更一般的基于采样的规划器，并且很容易实现。但是如果设计巧妙，基于第 7.4.2 节技术的规划器会更加有效率。

现在考虑对第 5.5 节的 RDT 进行适应性修改以使之能够解决带有封闭约束的问题。第 5.4 节的诸如随机化的势场规划器等其他递增采样和搜索方法也可能进行适应性修改。在 $\mathbb{R}^n$ 的一个有界 n 维子集上生成一个稠密搜索序列 α，如图 7.21 所示。实际上并不要求采样点在 $\mathcal{C}_{clo}$ 上，因为它们不一定是拓扑图 $\mathcal{G}$ 的一部分。它们主要用来将搜索树引向不同的方向。选择的包围区必须包括 $\mathcal{C}_{clo}$（至少是包括 q_I 的连通分支），但是它一定不能太大。通过分析特定连杆组的运动极限就可以得到这样的界。

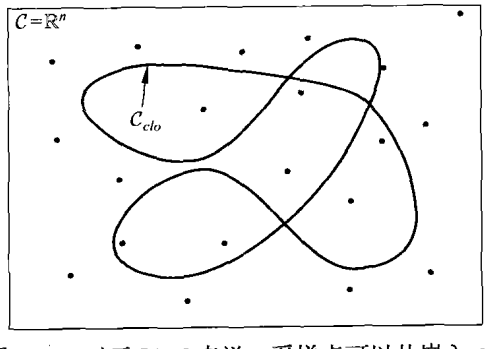


图 7.21 对于 RDT 来说，采样点可以从嵌入 $\mathcal{C}_{clo}$ 的空间 $\mathbb{R}^n$ 的一个区域中产生

沿着 C_{clo} 步进　可以直接应用图 5.21 给出的 RDT 算法；但是第 4 行中的 STOPPING-CONFIGURATION 函数必须加以改变以考虑障碍物和对 C_{clo} 的约束。图 7.22 显示了一种基于数值延拓的一般方法[18]。另外也可以使用逆运动学，它是第 7.4.2 节所描述的方法的一部分。首先计算距离采样点 $\alpha(i)$ 最近的 RDT 顶点 $q \in \mathcal{C}$。令 $v = \alpha(i) - q$ 表示如果没有约束，那么从 q 构造边的方向。然后通过将 v 投影到 q 点处 C_{clo} 的切平面[1]来计算一个局部运动。因为 C_{clo} 一般是非线性的，局部运动产生的点并不准确在 C_{clo} 上。允许存在一些允许偏差，采取足够小的步长来保证允许偏差。计算关于新的点的 v，并且在投影到新的切平面的 v 的方向上移动，该过程迭代进行。如果超过误差门限，那么必须在法线方向上进行运动回到 C_{clo}。当由于切平面对准（几乎垂直于 v）或者由于障碍物的缘故而迭代结果没有明显提高时，那么切向和法向运动过程结束。最后产生停止位形 q_s。新的路径不再是一条“直线”，这种“直线”对于第 5.5 节中的一些问题来说是可能的；因此应该采用第 5.5.2 节的近似方法来沿着路径创造中间顶点。

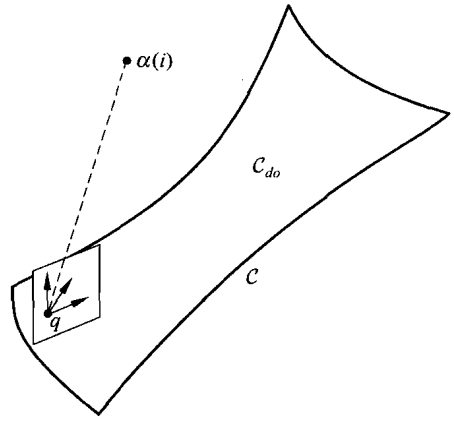


图 7.22　对于每个采样点 $\alpha(i)$，发现最近的点 $q_n \in \mathcal{C}$，然后局部规划器在局部切平面内生成一个运动；运动是从 q_n 到 $\alpha(i)$ 的向量到切平面的投影

在每次迭代中，在某 $q \in C_{clo}$ 处计算切平面。微分位形向量 $\mathrm{d}q$ 处在约束 $f_i(q) = 0$ 的切平面中，如果：

$$\frac{\partial f_i(q)}{\partial q_1}\mathrm{d}q_1 + \frac{\partial f_i(q)}{\partial q_2}\mathrm{d}q_2 + \cdots + \frac{\partial f_i(q)}{\partial q_n}\mathrm{d}q_n = 0 \tag{7.22}$$

这为 $\mathcal{P}$ 中的所有 k 个多项式产生如下齐次系统，该系统定义封闭约束：

$$\begin{pmatrix} \frac{\partial f_1(q)}{\partial q_1} & \frac{\partial f_1(q)}{\partial q_2} & \cdots & \frac{\partial f_1(q)}{\partial q_n} \\ \frac{\partial f_2(q)}{\partial q_1} & \frac{\partial f_2(q)}{\partial q_2} & \cdots & \frac{\partial f_2(q)}{\partial q_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_k(q)}{\partial q_1} & \frac{\partial f_k(q)}{\partial q_2} & \cdots & \frac{\partial f_k(q)}{\partial q_n} \end{pmatrix} \begin{pmatrix} dq_1 \\ dq_2 \\ \vdots \\ dq_n \end{pmatrix} = 0 \tag{7.23}$$

1　在第 8.3 节中，我们更严格地定义了切平面。

如果矩阵的秩 $m \leqslant n$，那么可以独立选择 m 个位形位移，其余 $n-m$ 个参数必须满足式(7.23)。采用诸如奇异值分解（SVD）等线性代数方法为 q 点的切平面计算一个正交基，问题就可以解决。令 $e_1,\cdots,e_m$ 表示这些 n 维基向量。从 $v=\alpha(i)-q_n$ 得到运动方向的分量。首先构造内积 $a_1=v\cdot e_1$、$a_2=v\cdot e_2$、$\cdots$、$a_m=v\cdot e_m$。那么 v 在切平面上的投影是 n 维向量 w：

$$w=\sum_i^m a_i e_i \tag{7.24}$$

它用作运动的方向。数量必须按比例缩小到足够小的步长。因为 $\mathcal{C}_{clo}$ 通常是弯曲的，一个线性运动会离开 $\mathcal{C}_{clo}$。需要在向内法线方向上的一个运动来回到 $\mathcal{C}_{clo}$。

因为 $\mathcal{C}_{clo}$ 的维数 m 小于 n，刚才描述的过程仅产生 $\mathcal{C}_{clo}$ 中路径的数值近似。这个问题在图和几何模型的隐含曲线跟踪中也会出现[457]。因此每个约束 $f_i=(q_1,\cdots,q_n)=0$ 实际上被稍微弱化到 $|f_i(q_1,\cdots,q_n)|<\epsilon$，对某固定允许偏差 $\epsilon>0$。从本质上讲，这“加厚了” $\mathcal{C}_{clo}$ 以使它的维数为 n。除计算切平面以外，运动的方向可以直接从这个加厚区域内进行采样，而不需要计算切平面。这很容易实现，但是效率较低[979]。

7.4.2 主动-被动连杆的分解

除基于采样的方法以外，还可以进行主动-被动分解，它通过直接采样主动变量来在 $\mathcal{C}_{clo}$ 中生成采样点，并采用逆运动方法来计算被动变量的封闭值。这种方法由文献[435]提出，并在文献[246,248]中的随机环路生成器(random loop generator)的发展中得到改进和提高。它作为一个可以对第 5 章的任何方法进行适应性修改来处理闭运动链的一般框架，实验显示其性能比第 7.4.1 节介绍的方法更好。不足之处是该方法要求对连杆组进行一些仔细分析来确定最佳分解及其可移动性的界限。对很普通的连杆组都存在这样的分析[246]。

主动和被动变量 这里令 $\mathcal{C}$ 表示从所有关节变量得到的 C-空间，而不是像第 7.4.1 节那样要求 $\mathcal{C}=\mathbb{R}^n$。这意味着 $\mathcal{P}$ 只包括建模封闭约束的多项式，这不同于允许表示旋转的约束。采用第 4.4.3 节的 3 种表示，这意味着在没有任何约束的情况下为连杆组的每个旋转关节赋予一个变量时，$\mathcal{C}$ 是 n 维的。令 $q\in\mathcal{C}$ 表示位形变量的向量。主动-被动分解将 q 的变量分拆形成两个向量：主动变量 q^a 和被动变量 q^p。被动变量的值总是通过施加封闭约束和采用逆运动技术从主动变量产生而来的。如果 $\mathcal{C}_{clo}$ 的维数为 m，那么总有 m 个主动变量和 $n-m$ 个被动变量。

临时地，假设连杆组形成了一个环，如图 7.23 所示。图 7.24 给出了一种可能的主动变量 q^a 和被动变量 q^p 的分解。如果约束为形成一个环，那么连杆组有四个自由度，假设底部的连杆固定在地面上。这意味着可以为四个主动关节角 q^a 选择值，其余的三个 q^p 可以通过解决逆运动来产生。为了确定 q^p，有三个方程和三个未知数。遗憾的是，这些方程是非线性的，而且非常复杂。尽管如此，对于这种情况及其三维扩展情况来说依然存在高效的求解方法[678]。对于一个由旋转关节组成的三维环，有 6 个被动变量。$\mathbb{R}^2$ 中的被动连杆的数量为 3，$\mathbb{R}^3$ 中的被动连杆的数量为 6，它们分别来自于 $SE(2)$ 和 $SE(2)$ 的维数。这是可以通过施加封闭约束而从系统中去除的自由度。文献[30]给出了许多有效地计算二维或三维逆

运动的方法。这些方法也可以应用于第 7.4.1 节的 RDT 步进方法，而不使用延拓。

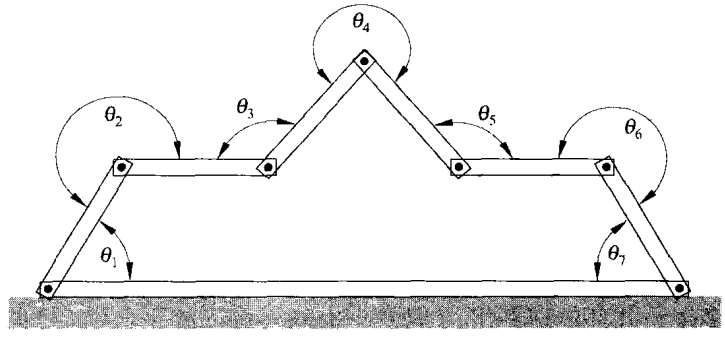


图 7.23　平面中的连杆链：由七个连杆和七个关节构成一个环；C 的维数是七，但是 C_{clo} 的维数是四

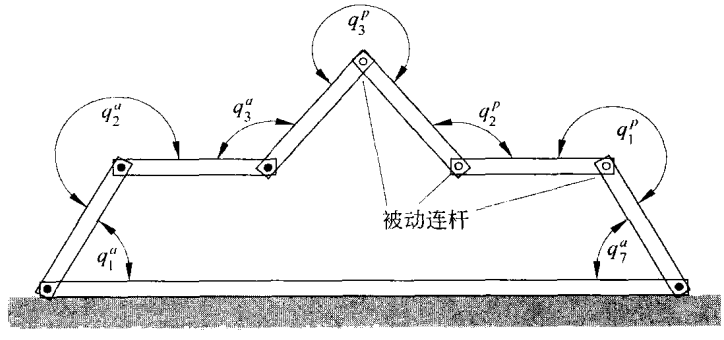


图 7.24 三个关节变量可以用逆运动学自动确定；因此四个关节被指定为主动的，其余三个将是被动的

如果使用最大数量的被动变量，那么对逆运动学问题来说至多有有限数量的解；这意味着被动变量的值通常有多种选择。可能有这种情况：对某些主动变量值，逆运动无解。图 7.25 描述了一个例子。假设我们想通过对 q^a 选择随机值、对 q^p 采用逆运动学来生成 C_{clo} 中的采样点。那么逆运动的解存在的概率是多少？对于所示的例子来说，在大多数试验中，解是不存在的。

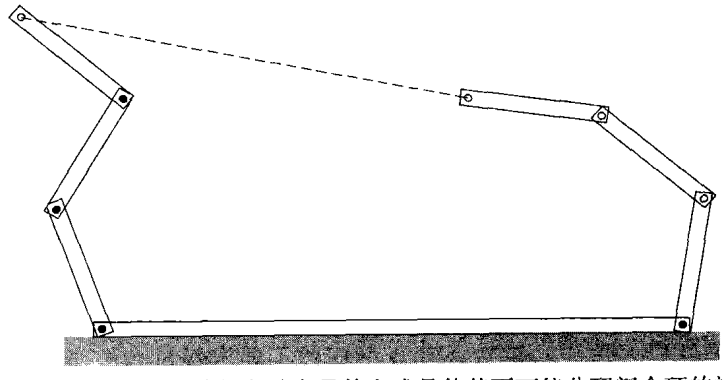
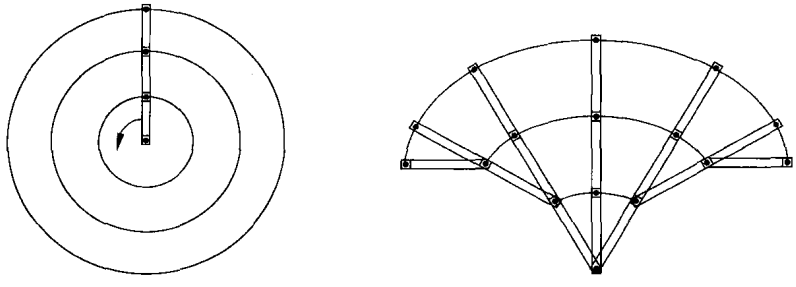


图 7.25　在这种情况下，选择主动变量的方式是使其不可能分配闭合环的被动变量

环路生成器　通过迭代地限制每个主动变量的范围，随机环路生成器（random loop generator）可以极大地提高闭合的机会。方法要求主动变量沿着链依次出现（例如主动变

量和被动变量没有重叠）。下面依次得到 q^a 的 m 个坐标。首先计算 q_1^a 的允许值的区间 I_1。这个区间作为如下意义下的一个宽松的界：对于任何值 $q_1^a \notin I_1$，绝对不可能得到封闭。通过对下面将要进行解释的连杆组仔细地几何分析，这可以得到保证。下一步是在 $q_1^a \in I_1$ 中生成一个采样点，文献[246]是通过在 I_1 中获取一个随机点来实现这一步的。使用值 q_1^a，可以计算 q_2^a 的允许值的一个包围区间 I_2。通过在 I_2 中进行采样得到值 q_2^a。这个过程迭代进行直到得到 I_m 和 q_m^a，除非过程因为某 $I_i = \phi$ 而提前停止，$i < m$。如果过程得到 I_m 和 q_m^a 而停止，那么可以用主动变量 q^a 来寻找被动变量的值 q^p。这一步虽然还可能会失败，但是成功的概率非常高。通过将连杆组拆为一个树并且当使用环路生成器的时候闭合环路，该方法也可以应用于有多个公共环的连杆组，例如在 Stewart-Gough 平台中，性能取决于如何分解连杆组[246]。

计算关节角的界 该方法的成功应用主要是要求能够计算连杆链在关节变量的某范围上在 $\mathcal{W}$ 中能够运行多远。例如对于一个有不受限的旋转关节的平面链，链能够扫出一个圆，如图 7.26（a）所示。假设知道连杆之间的角度必须保持在 $-\pi/6$ 和 $\pi/6$ 之间。那么可以得到一个紧的包围区，如图 7.26（b）所示。在三维情况下的这些限制及其必要的详细说明，参见文献[246]。然后在采样算法的每次迭代中使用这些界限来计算 I_i。



(a) (b)

图 7.26 （a）如果关节的角度可以是任意的，那么在极限状况下连杆扫出一个圆；（b）如果对关节的角度有限制，那么对连杆组的可达性来说得到的是一个更紧的界

既然有高效的方法来从 $\mathcal{C}_{clo}$ 直接生成采样点，那么对第 5 章中任何基于采样的规划方法进行适应性修改就非常简单了。对 $\mathcal{C}$ 的维数高达 97、$\mathcal{C}_{clo}$ 的维数达到 25 这样复杂的问题，在文献[246]中得到了一些令人注目的结果，如图 7.27 所示。这些方法是基于将新的采样技术应用到第 5.5 节的 RDT 和第 5.6.2 节的基于可视性采样的路线图中。对于这些算法，在进行路径验证时，对主动变量应用局部规划方法，对被动变量采用逆运动学算法。这意味着逆运动学算法和碰撞检测同时进行，而不是像第 5.3.4 节那样只进行碰撞检测。

这里所忽视的一个重要问题是运动奇点的存在性，该奇点引起在某些点附近的 $\mathcal{C}_{clo}$ 的维数降低。这里的方法已经假设：解决运动规划问题不要求经过一个奇点。对有多余自由度的机器人系统来说这种假设是合理的，但是在一般情况下会丧失完备性，因为基于采样的方法并不明确处理这些退化，理解这一点是很重要的。从某种意义上，它们发生在比采样分辨率更低的层次上。更多关于运动奇点及其相关问题，参见文献[696]。

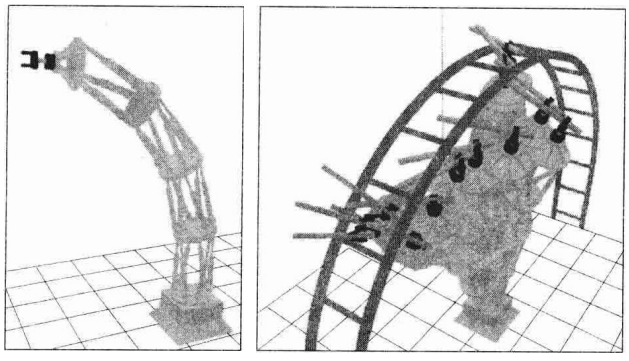


图 7.27　Logabex LX4 型机器人的规划[189]；将主动-被动分解应用于基于 RDT 的规划器，在一分钟内计算出规划解[246]；在这个例子中，$\mathcal{C}$ 的维数是 97，$\mathcal{C}_{clo}$ 的维数是 25

7.5　机器人学和生物学中的折叠问题

越来越多的运动规划涉及某种形式的折叠。这方面的例子包括自动纸箱折叠、计算机辅助药物设计、蛋白质折叠、可重配置的机器人、机器人折纸术等。这些问题一般情况下都建模为一个连杆组，其中所有的物体用旋转关节连接。在机器人学中通常必须避免成对物体之间的自我碰撞。在生物学的应用中，能量函数代替障碍物。能量函数可以被想象为“软的”障碍物，其中为每个 $q \in \mathcal{C}$ 定义一个真实值，而不是定义一个集合 $\mathcal{C}_{obs} \subset \mathcal{C}$。对于一个给定的门限值，通过将 $\mathcal{C}_{obs}$ 定义为能量大于门限值的位形，这样的能量函数就可以转换为一个障碍区域。但是能量函数包含更多信息，因为这个门限是任意的。这里简要介绍一些折叠问题的例子以及从最新的运动规划文献中得到的处理折叠问题的技术。

纸箱折叠　文献[664]提出了一个有趣的运动规划应用于纸箱的自动折叠的例子。图 7.28 显示了一个纸箱处在初始平面形态和折叠形态。如图 7.29 所示，问题可以建模为一个用旋转关节连接起来的物体树。一旦建立这种模型，那么第 5 章和第 6 章的许多方法都可以应用，只是需要适应性修改。文献[664]提出了一个专门为箱子的折叠而进行了优化的规划算法。它是对文献[661]中为运动链而开发的近似胞腔剖分算法的一个适应性修改。它的复杂性与纸箱自由度呈指数关系。但是在实际例子中它给出更好的性能。图 7.30 显示了由运动规划所发现的这样的一个解。为了在工厂中使用这些解，还必须考虑操作问题。例如文献[664]所提到的，机械臂可以和一个夹具一起使用。夹具固定纸箱，而机械臂在相应位置施加外力，这样就产生了所需要的折叠。因为夹具的可行性取决于特定的折叠路径，规划算法生成从初始位形（箱子完全没有折叠）开始的所有可能的路径。

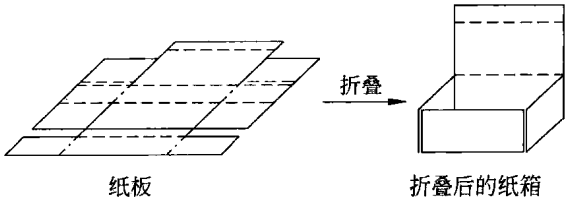


图 7.28　一个重要的包装问题是将打过孔的纸板自动折叠成纸箱

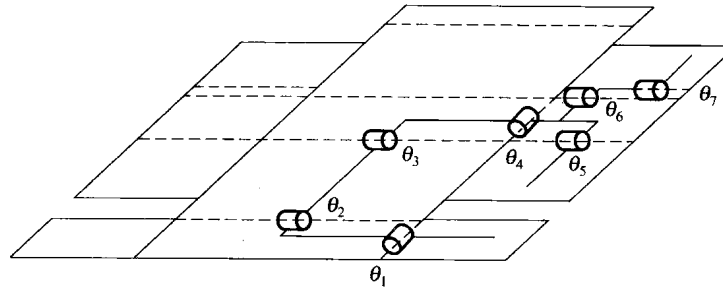


图 7.29 纸箱可以被建模为用旋转关节连接起来的物体的树

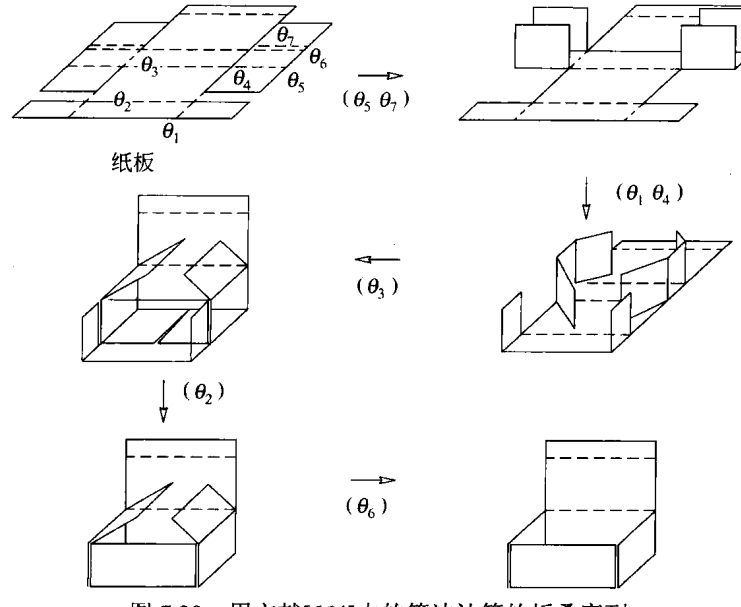


图 7.30 用文献[664]中的算法计算的折叠序列

简化节点 一个节点是一条闭曲线，它自身不交叉、嵌入 $\mathbb{R}^3$ 中、不能解开产生一个简单环（诸如一条圆形路径）。如果允许节点自身交叉，那么任何节点都可以解开；因此需要仔细定义解开节点是什么含义。对于一条嵌入 $\mathbb{R}^3$（它自身不交叉）的闭曲线 $\tau:[0,1]\rightarrow\mathbb{R}^3$，曲线不能到达的点的集合 $\mathbb{R}^3\setminus\tau([0,1])$ 称为 τ 的环境空间（ambient space）。在节点理论中，嵌入 $\mathbb{R}^3$ 中的两条闭曲线 τ_1 和 τ_2 之间的一个环境合伦（ambient isotopy）是它们的环境空间之间的一个同胚。从直觉上讲，这意味着在不允许任何自身交叉的情况下 τ_1 可以弯曲进入 τ_2 中。因此确定两个环是否等价似乎与运动规划紧密相连。这种等价关系产生的群刻画了节点空间的特征，并与第 4.1.3 节描述的基本群密切相关。更多关于节点理论的信息见文献[8,454,514]。

文献[574]发展了一种运动规划方法来确定一条闭曲线是否等价于完全解开的开节点。这可以表达为嵌入 $\mathbb{R}^3$ 中、映射到 $\mathbb{S}^1$ 的一条曲线。算法将节点作为输入，节点表示为嵌入 $\mathbb{R}^3$ 中的线段的一个圆形链。在这种情况下，解开的节点表示为 $\mathbb{R}^3$ 中的一个三角形。图 7.31 显示了由规划器解决的一个最具挑战性的例子。规划器是基于采样的，它与第 5.5 节中的 RDT 算法、第 5.4.4 节中的 Ariadne 线索和扩展空间规划器有许多相似性。因为任务不是产

生无碰撞路径，那么与运动规划比较而言，它有几个独特之处。在线段族上定义一个能量函数来试图将搜索引向更简单的位形。规划器产生两种局部操作：（1）试图使一个顶点向环境空间中所选择的子目标移动，这通过用随机采样来生长一个搜索树就可以得到；（2）试图删除一个顶点，并用直线连接邻近顶点。如果沿着中间位形没有碰撞发生，那么节点已经得到简化。当不能进一步简化节点时，算法终止。

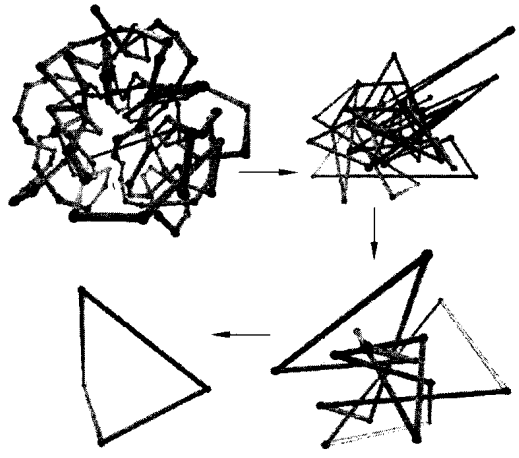


图 7.31　文献[574]中的规划器在标准 PC 上仅用几分钟时间就能够解开著名的 Ochiai 的结

药物设计　文献[604]提出了用于药物设计的一个基于采样的运动规划方法。药物的发展是一个长期的、递增的过程，通常需要经过多年的研究和试验。设计目标是发现一个相对小的、被称为配位体的分子，这样的分子一般是由多个原子组成，它与一个特定蛋白质的受体腔对接[618]。图 1.14（第 1.2 节中）阐述了这种情况。图 1.14 也给出药物分子的例子。蛋白质-配位体的对接可以刺激或者抑止某生物活性，这最终将导致所期望的药理学效果。由于能量和配位体的灵活性的缘故，发现合适配位体的问题很复杂。除了满足结构方面的考虑以外，诸如可合成性、药物药理学和毒物学等方面的因素都增加了问题的复杂性，加长了对有效药物分子的搜索过程。

化学家在进行药物设计时用到的一个流行的模型是一个药效团（pharmacophore），它作为想得到的配位体的一个模板[231,341,385,862]。药效团表示为有效的配位体应该拥有的一个要素集合和这些要素中空间约束的集合。要素是指特定的原子、苯环中心、正负电荷、恐水中心或者亲水中心、氢键的供体或受体等。要素通常要求分子的组成部分必须在 $\mathbb{R}^3$ 中保持固定，这产生运动封闭约束。化学家提出的这些要素用来概述假设：配位体的绑定根本上是由于配位体的某些要素与受体的“互补”要素的相互作用。相互作用的要素包括在药效团中，它们是遴选候选药物的一个模板，其他配位体原子仅提供一个平台使得药效团要素保持在它们的空间位置上。图 7.32 阐述了药效团的概念。

如图 1.14 所示的候选药物分子（配位体）可以建模为物体的一个树，如图 7.33 所示。某些键可以旋转产生模型中的旋转关节；其他键必须保持固定。药物设计问题等同于搜索位形空间以试图发现一个低能量的位形，该位形也将特定原子放置到 $\mathbb{R}^3$ 中的特定位置。附加的约束从药效团产生，并引起第 7.4 节的 $\mathcal{C}_{obs}$ 上的规划，因为药效团可以表示为封闭约束。

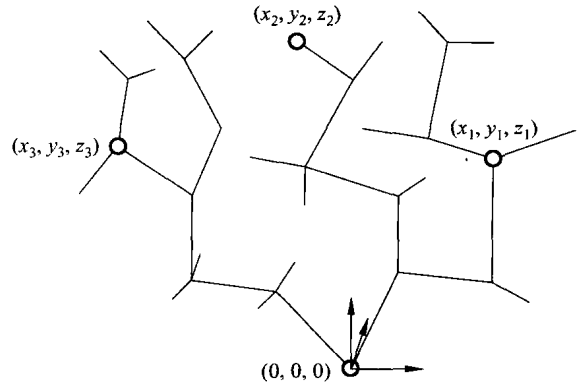


图 7.32 药效团是一个模型，化学家用它来简化一个配位体（候选的药物分子）和一个蛋白质之间的相互作用过程；它通常相当于将分子的要素在 $\mathbb{R}^3$ 中保持固定；在这个例子中，相对于任意选定的一个根原子所建立的体坐标系，三个原子的位置必须固定；假设在蛋白质的腔中这些要素与某些互补要素相互作用

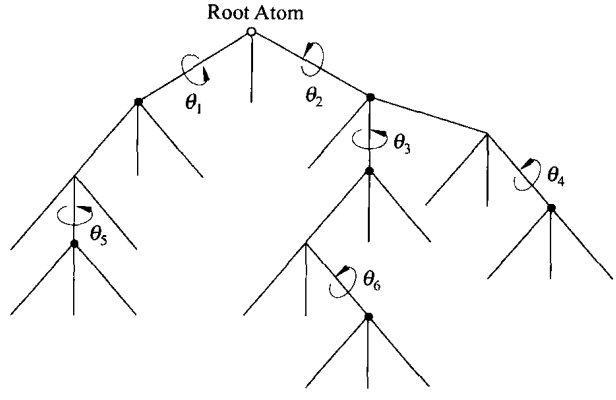


图 7.33 一个柔性分子的模型类似于一个机器人的模型；其中一个原子被指定为根，其余的排列成一棵树；如果在分子中有环链，那么必须施加第 4.4 节介绍的约束；典型情况下，仅有某些键能够旋转，而其他的必须保持刚性

一个能量函数的目的与碰撞检测器的目的相似。为药物设计所进行的配位体计算要求确定配位体是否能够实现满足药效团约束的低能量形态。因此任务与标准运动规划的不同之处在于没有预先确定的目标位形。最大的困难在于能量函数是极其复杂的、非线性的、经验的。这里有一个典型的例子（在文献[604]中使用）：

$$\begin{aligned} e(q) = & \sum_{bonds} \tfrac{1}{2} K_b (R - R')^2 + \sum_{ang} \tfrac{1}{2} K_a (\alpha - \alpha')^2 + \\ & \sum_{torsions} K_d \left[1 + \cos(p\theta - \theta')\right] + \\ & \sum_{i,j} \left\{ 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] + \frac{c_i c_j}{\epsilon r_{ij}} \right\} \end{aligned} \tag{7.25}$$

能量考虑了扭转角的变形、van der Waals 势以及 Coulomb 势。在式（7.25）中，第一个和式项涉及的是所有的键，第二项涉及所有的键角，第三项涉及所有可旋转的键，最后一项涉及所有成对的原子。变量是外力常数 K_b、K_a、K_d；电介质常数 ϵ；周期常数 p；

Lennard-Jones 半径 σ_{ij}；深度 ϵ_{ij}；局部电荷 c_i；测量的键的长度 R；平衡键的长度 R'；测量的键角 α；平衡键角 α'；测量的扭转角 θ；平衡扭转角 θ'；原子中心间的距离 r_{ij}。尽管能量的表达式非常复杂，但是它仅依赖于位形变量；其他的则是提前估计的常数。

蛋白质折叠　在计算生物学中，蛋白质折叠与药物设计的相似之处在于分子有可旋转的键以及能量函数用来表达好的位形。但是问题更加复杂，因为蛋白质分子通常比药物分子大很多。不像药物分子那样通常只有若干个自由度，蛋白质分子有成百上千个自由度。当蛋白质在自然界中出现时，它们通常处于折叠着的、低能量位形。结构问题涉及精确确定蛋白质是如何折叠的，这样就可以完全了解它的生物活性。在一些研究中，生物学家甚至对蛋白质到达折叠状态的路径感兴趣[24,25]。这直接导致了一个运动规划的扩展，这种扩展涉及到达分子折叠这个目标状态。在文献[24,25]中，应用基于采样的规划算法来计算蛋白质的折叠的路径。蛋白质从展开的位形开始，必须在不违背沿着路径的能量约束的条件下到达一个特定的折叠位形。图 7.34 显示了一个来自于文献[25]的例子。这也展示了蛋白质折叠与前面介绍的箱子折叠之间有趣的联系。

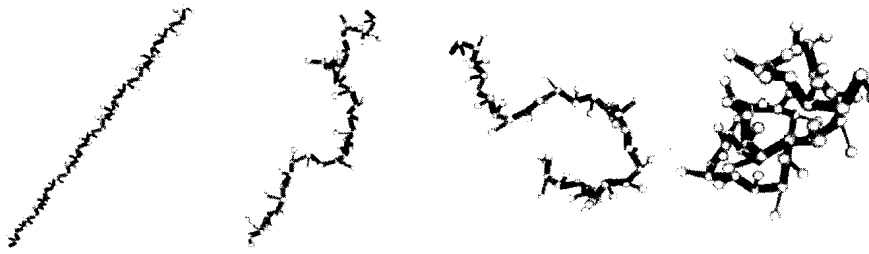


图 7.34　用基于采样的路线图规划算法所计算的一个多肽蛋白质折叠[24]

7.6　覆 盖 规 划

假设在一座有许多障碍物的庄园中，一个割草机正在运动，其中的障碍物包括房屋、树、车库以及复杂的庄园边界等。割草机最好的运动是什么？多余的往返移动能否最小化？割草机需要停止和旋转的次数能否最小化？这就是覆盖规划的一个例子，它是受到诸如割草机割草、自动耕作、涂漆、真空吸尘器清扫以及清理地雷等应用的激发。这方面的一个概述参见文献[219]。甚至对于 $\mathcal{W}=\mathbb{R}^2$ 中的一个区域，即使简缩到与覆盖规划密切相关的旅行商问题[36,712]，发现覆盖规划的一个最优长度的解仍然是 NP-复杂度的。因此甚至 $\mathbb{R}^2$ 中我们也宁愿容忍一般覆盖问题的近似解或者启发性的解。

boutrophedon 剖分　解决覆盖问题的一个方法是将 $\mathcal{C}_{free}$ 剖分为胞腔，然后在每个胞腔中进行 boustrophedon 运动，如图 7.35 所示[224]。假设机器人是 $\mathcal{W}=\mathbb{R}^2$ 中的一个点，但是它带有一个能够均匀地贴到机器人的侧面、厚度为 ϵ 的工具。这使它在向前运动时能够涂上或者抹去一部分 $\mathcal{C}_{free}$，该部分是从机器人的任意一侧所开始的 $\epsilon/2$ 厚度。这种运动经常在打印机中使用来减少换行回车的次数。

如果 $\mathcal{C}_{obs}$ 是多边形的，那么通过适应性修改第 6.2.2 节的垂直剖分方法就可以得到一个合理的剖分。在那个算法中定义了几种临界事件，这些事件对 boustrophedon 运动来说是不

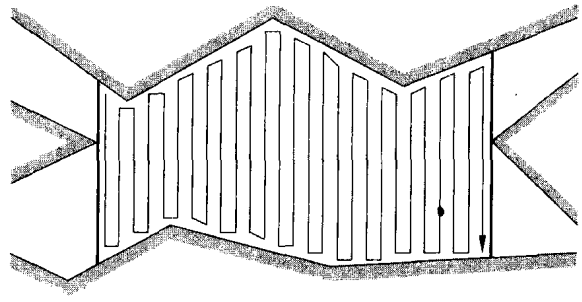
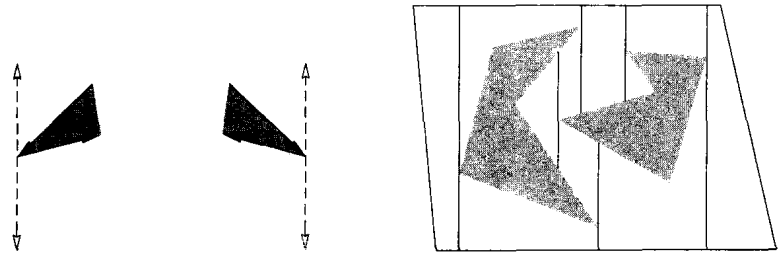


图 7.35 犁耕式运动的一个例子

相关的。需要处理的事件如图 7.36（a）所示[218]。这产生一个有较少胞腔的剖分，如图 7.36（b）所示。即使胞腔是非凸的，它们总可以切为垂直的带，这使得它们适合 boustrophedon 运动。也可以使用原始的垂直剖分，但是额外的胞腔边界将引起不必要的机器人重定位。文献[471]提出一个相似的方法，它可进一步优化机器人转向的次数。



(a) (b)

图 7.36 （a）仅需要图 6.2 的第一种情况：向上和向下延伸，忽略其他情况；（b）显示了所产生的剖分，它比图 6.3 中垂直剖分所产生的胞腔少

生成树的覆盖 Gabriely 和 Rimon 提出了一个有趣的近似方法；该方法将一个正方形格片放置到 C_{free} 里面，计算所产生的连通图的生成树[374,375]。假设 C_{free} 是多边形。考虑图 7.37（a）的例子。第一步是用正方形格片覆盖 C_{free} 的内部，如图 7.37（b）所示。每个正方形的宽度应该为 ϵ，ϵ 为常数 $\epsilon > 0$。接下来将一个顶点放置到正方形的中心、定义一条连接每个相邻方体对的中心的边，这样就可以构造一个路线图 $\mathcal{G}$。下一步计算 $\mathcal{G}$ 的生成树。这是一个连通子图，它没有循环，并且与 $\mathcal{G}$ 的每个顶点相切；如果 $\mathcal{G}$ 有 n 条边，那么它的计算可以在 $O(n)$ 内完成[686]。有许多可能的生成树，可以定义并优化一个指标来进行选择。图 7.37（c）显示了一个可能的生成树。

一旦产生生成树，那么从生成树附近的一个点开始、沿着它的周长就可以得到机器人的路径。该路径可以精确指定，如图 7.38 所示。加倍格片覆盖的分辨率，并形成相应的路线图。路线图的一部分对应于生成树，但是也包括一个围绕该生成树的环形路径。该路径访问新的正方形格片的中心。为图 7.37（a）的例子所产生的路径如图 7.37（d）所示。一般情况下，给定逐次逼近，方法就能够产生一个最优的路线。文献[374,375]给出了不要求初始地图的方法；这涉及将在第 11 章进行讨论的信息空间。

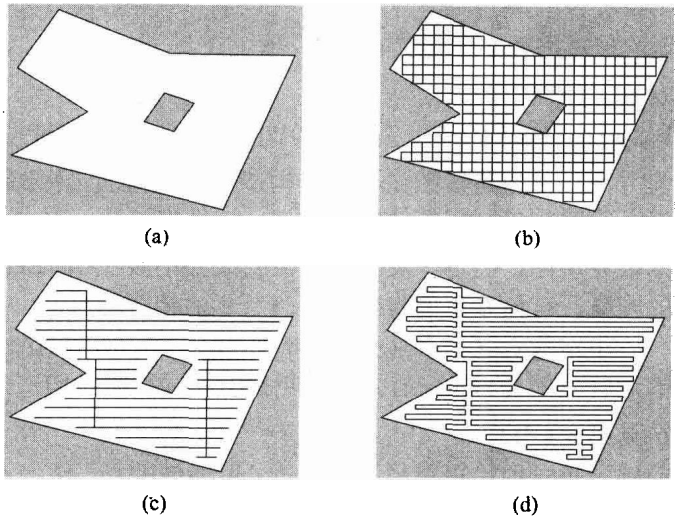


图 7.37　(a) 生成树覆盖的一个例子；(b) 第一步是用正方形格片覆盖内部；(c) 邻近栅格所形成的一个路线图的生成树；(d) 所产生的覆盖路径

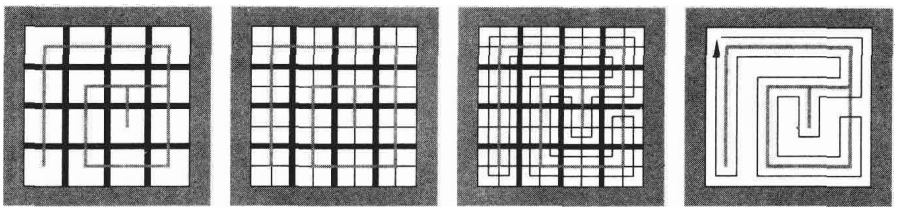


图 7.38　将分辨率加倍并沿着生成树的周长就产生一条迂回路径

7.7　最优运动规划

到目前为止，关于运动规划，我们主要关心的是可行性，而不是最优性。这就意味着应着力于发现任意解，而不是进一步要求解的最优性。在本书的以后部分，特别是当引入不确定性后，最优性将会得到更多的关注。甚至决策论（见第 9 章）的最基本形式也集中在进行最优选择上。在非常一般的情况下对最优性的要求通常需要在状态空间上进行穷举搜索，这等于计算连续尚需代价函数。一旦知道这个函数，就可以得到反馈规划，该规划比仅有一条路径要有力得多。因此最优性也经常出现在反馈规划的设计中，因为它有时并不需要额外的代价。在第 8 章这将会变得更加清晰。寻求最优解也会引起有趣的关于如何将一个连续问题近似成一个离散问题的问题。在联系连续规划问题和离散规划问题时，时间离散和空间离散之间的相互影响变得非常重要。

7.7.1　一个机器人的最优性

欧氏最短路径　关于最优性的最直截了当的概念之一就是 $\mathbb{R}^2$ 或者 $\mathbb{R}^3$ 中欧氏最短路径。假设 $\mathcal{A}$ 是一个只能在 $\mathcal{W}=\mathbb{R}^2$ 或者 $\mathcal{W}=\mathbb{R}^3$ 中进行平移的刚体机器人，其中包含一个障

碍区域$\mathcal{O}\subset\mathcal{W}$。回忆到，$\mathcal{C}_{free}$通常是一个开集合，这意味着任何路径$\tau:[0,1]\to\mathcal{C}_{free}$都可以缩短。因此运动规划的最短路径必须定义在闭包$cl(\mathcal{C}_{free})$上，这允许机器人接触障碍物；但是它们的内部不能交叉。

对于$\mathcal{C}_{obs}$是一个多边形区域的情形，第 6.2.4 节的最短路径路线图方法已经给定。这可以认为是一种多疑问方法，因为路线图完全获得构造任何疑问的最短路径所需要的结构。有可能用连续 Dijkstra 来产生单疑问算法[446,711]。这种方法从q_I传播一个波前，并且在保持波前的同时跟踪临界事件。当事件发生，波前变成由小波（wavelet）组成，小波是以障碍物顶点为中心的圆弧。可能发生的事件是：（1）小波消失；（2）一个小波与障碍物的顶点发生碰撞；（3）一个小波与其他小波发生碰撞；（4）一个小波与障碍边内部中的一个点发生碰撞。方法的运行时间为$O(n\lg n)$，它使用$O(n\lg n)$空间。构造一个使用$O(n)$空间的路线图。相关方法见第 8.4.3 节。

这种方法给人的感觉是发现最短路径并不困难，但遗憾的是，它们不能推广到$\mathbb{R}^3$和多面体$\mathcal{C}_{obs}$。图 7.39 显示了一个简单的例子，其中的最短路径不一定经过$\mathcal{C}_{obs}$的一个顶点。它可能经过一条边内的任何地方；因此尚不清楚在哪里绘制将形成最短路径路线图的双切线。这个问题的下界也很令人沮丧。文献[174]显示在多面体环境中三维最短路径问题是 NP 复杂度的。大多数困难都来自于对描述三维最短路径的精度要求。因此存在高效的多项式时间的近似算法[217,766]。

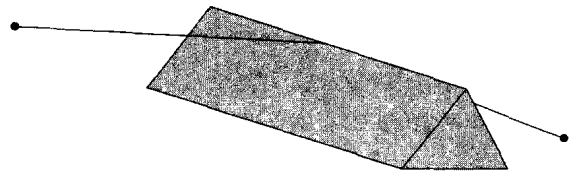


图 7.39 对于一个多面体环境来说，最短路径不一定经过顶点；因此第 6.2.4 节中的最短路径路线图方法不能扩展到三维

一般最优性准则 很难定义更一般的 C-空间的最优性。在$SE(2)$和$SE(3)$中的最短路径意味着什么？考虑一个能够平移和旋转的平面刚体机器人。一条路径可以最小化旋转的次数，而另外一条路径试图最小化平移的次数。如果没有更多信息，那么就不清楚哪个更好。Ulam 距离是一种可能的选择方法，它最小化k个固定点移动的距离[477]。第 13 章引入微分模型，这些模型可以产生非常有意义的关于最优性的定义。例如第 15.3 节显示了缓慢移动的汽车的最短路径；这些都要求约束汽车运动的一个精度指标（汽车的侧向运动比向前运动付出的代价更多）。

本节表述一些最优运动规划问题，逐步向后面的概念平滑过渡。到现在为止采用的是第 2 章离散规划问题中的行动，但是通过直接描述映射到$\mathcal{C}_{free}$的路径，基本运动规划成功地避开行动。再次用它们也很方便。回忆到第 2.3 节，它们能够很方便地定义代价和最优规划。

为了避开复杂的微分方程，考虑构造运动规划的一个近似模型，模型中的每条路径必须由$\mathcal{C}_{free}$中的一系列最短路径线段组成。通常这是直线段；但是对于$SO(3)$情况，通过球面线性插值得到的圆弧可能更可取。考虑将第 2.3.2 节中的表述 2.3 扩展到运动规划问题。

令 C-空间 C 嵌入 $\mathbb{R}^m$（例如 $C \subset \mathbb{R}^m$）。将一个行动定义为一个 m 维向量。给定一个比例常数 ϵ 和一个位形 q，一个行动 u 产生一个新的位形 $q' = q + \epsilon u$。这可以认为是一个位形转移方程 $q' = f(q,u)$。用行动 u 表示的路径线段是 q 和 q' 之间的最短路径（通常是一条直线段）。像第 2.3 节那样，令 π_K 表示一个 K 步规划，它是 K 个行动的序列 $(u_1,u_2,\cdots,u_K)$。注意到如果给定 π_K 和 q_I，那么一个状态序列 $q_1,q_2,\cdots,q_{K+1}$ 可以用 f 产生。初始情况下，$q_1 = q_I$，下一个状态由 $q_{k+1} = f(q_k,u_k)$ 得到。从此可以产生一条路径 $\tau:[0,1] \to C$。

一个近似最优规划问题表述如下所示。

表述 7.4（近似最优运动规划）

1. 如下组成部分与表述 4.1 相同：$\mathcal{W}$、$\mathcal{O}$、$\mathcal{A}$、$\mathcal{C}$、$\mathcal{C}_{obs}$、$\mathcal{C}_{free}$ 和 q_I。假设 C 是一个 n 维流形。

2. 对每个 $q \in C$，有一个可能的无限行动空间 $U(q)$。每个 $u \in U$ 是一个 n 维向量。

3. 一个正常数 $\epsilon > 0$ 称为步长。

4. 用 k 表示的阶段的集合从 $k=1$ 开始，并且无限进行。每个阶段用下标表示，得到 q_k 和 u_k。

5. 一个位形转移函数 $f(q,u) = q + \epsilon u$，其中 $q + \epsilon u$ 在 $\mathbb{R}^m$ 上用向量加法进行计算。

6. 定义一个目标区域 X_G，而不是定义一个目标状态。

7. 令 L 表示一个实值代价泛函，它应用于 K 步规划 π_K 中。这意味着行动序列 $(u_1,u_2,\cdots,u_K)$ 和位形序列 $(q_1,q_2,\cdots,q_K)$ 可能出现在 L 的表达式中。令 $F = K+1$，那么代价泛函为：

$$L(\pi_K) = \sum_{k=1}^{K} l(q_k,u_k) + l_F(q_F) \tag{7.26}$$

最后一项 $l_F(q_F)$ 在和式之外，它定义为如果 $q_F \in X_G$，$l_F(q_F) = 0$；否则 $l_F(q_F) = \infty$。就像表述 2.3 中那样，K 不必是一个常数。

8. 每个 $U(q)$ 包含特殊的终止行动 u_T，它的行为与表述 2.3 中的行为相同。如果 u_T 在第 k 阶段应用于 q_k，那么行动将永远重复进行，位形永远保持在 q_k 中，代价也不进一步积累。

我们的任务是计算一个优化式（7.26）的行动序列。表述 7.4 可以用来定义一簇最优规划问题。参数 ϵ 可以认为是近似分辨率。在许多表述中，它可以阐述为一个时间步 $\epsilon = \Delta t$；但是不需要明确的时间，因为问题只要求在 $\mathcal{C}_{free}$ 构造一条路径。当 ϵ 接近于零时，表述就接近于一个精确的最优规划问题。为了正确表达精确问题，需要采用微分方程，这将在第Ⅳ部分进行介绍。

例 7.4（Manhattan 运动模型） 假设除了 u_T，行动的集合 $U(q)$ 还包括 $2n$ 个向量，每个向量只有一个分量是非零的并且值为 1 或者-1。例如如果 $C = \mathbb{R}^2$，那么：

$$U(q) = \{(1,0),(-1,0),(0,-1),(0,1),u_T\} \tag{7.27}$$

当在位形转移方程中使用时，行动的集合产生“上”、“下”、“左”、“右”运动和一个“终止”命令。这按照第 5.4.2 节给出的 1-邻域模型（5.37）产生了一个拓扑图。这个例子和接下来的两个例子的行动集合如图 7.40 所示。对于所有 $q_k \in \mathcal{C}_{free}$ 和 u_k，代价项 $l(q_k,u_k)$ 定义

为 1。如果 $q_k \in \mathcal{C}_{obs}$，那么 $l(q_k,u_k)=\infty$。这些行动所能够到达的位形的集合处在一个栅格上，其中的 1-邻域之间的间隔为 ϵ。这对应于一个特殊情形，其中时间离散化（用 ϵ 实现）导致一个正则空间离散化。考虑图 7.41，不可能沿着对角来获取一条最短路径，因为行动不允许它。因此所有沿着栅格的单调路径产生相同的代价。

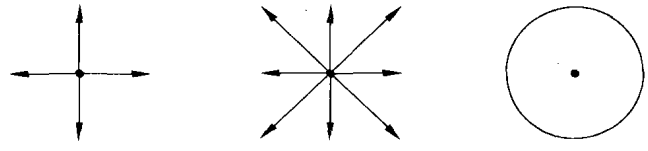


图 7.40 例 7.4、例 7.5 和例 7.6 的行动集合 $U(q)$

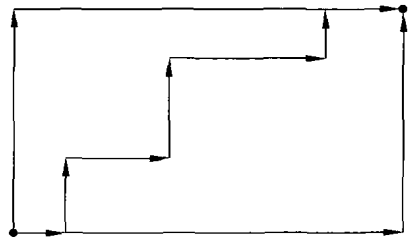


图 7.41 在 Manhattan(L_1)运动模型下，跟随栅格方向的所有单调路径有相等的长度

应用第 2 章的动态规划算法可以得到最优路径。这个例子将第 2.2 节的栅格搜索概念和第 5.4.2 节的阐述如何为运动规划而对搜索方法适应性修改的概念统一起来。在目前状态下，只有在相应的图上产生最优解的算法才是可接受的。

这种优化形式似乎与我们的问题无关，因为它并不表示 $\mathbb{R}^2$ 上的欧氏最短路径解。下面的模型将增加更多行动，该模型对应于机器人学中一类重要的优化问题。■

例 7.5（独立关节运动模型） 现在假设 $U(q)$包括 u_T 和所有 3^n 个向量的集合，向量的元素为-1、0 或者 1。基于这种模型，沿着任何对角线可以取得一条路径。这依然没有改变所有可达位形处都在一个栅格中的事实。因此可以再次使用标准栅格算法。不同之处在于现在从每个顶点发出 3^n-1 条边，而在例 7.4 中只有 $2n$ 条。这种模型很适合由旋转关节连接起来的连杆族所构造的机器人。如果每个关节独立操作，那么每个关节向前运动、向后运动，或者保持静止才有意义。这些运动对应于行动。但是这种模型不能很好地近似欧氏最短路径，这就激发产生了下面的例子。■

例 7.6（欧氏运动模型） 为了近似欧氏最短路径，令 $U(q)=\mathbb{S}^{n-1}\cup\{u_T\}$，其中 $\mathbb{S}^{n-1}$ 是以 $\mathbb{R}^n$ 的原点为中心的 m 维单位球。这意味着在 k 个阶段中，任何分段线性的路径都可以用一系列输入来形成，其中每条线段的长度为 ϵ。因此可达状态的集合并不局限于一个栅格。考虑取 $\epsilon=1$，选取任何点，如 $(\pi,\pi)\in\mathbb{R}^2$。你能够到达距离该点多么近？如果不考虑障碍物，那么这种模型所能够到达的点的集合在 $\mathbb{R}^n$ 中是稠密的。这意味着我们可以到达距离 $\mathbb{R}^n$ 中的点任意近的地方。因此一个有限栅格不能用来表达这种问题。通过数值计算 C 上一个最优尚需代价函数，可以得到近似解。这种方法将在第 8.5.2 节介绍。

对该问题还需要考虑目标的精度。如果目标区域相对 ϵ 而言非常小，那么为了精确到达目标，可能不得不选择复杂的路径。■

例 7.7（加权区域问题） 在室外和星际导航中，用目前所使用的硬性方法定义障碍物是没有意义的。对于每块地形，定义一个代价将是非常方便的，该代价指示经过该地形的估计难度。这有时被认为是障碍物的一个“灰色尺度”模型。在代价项 $l(q_k,u_k)$ 中很容易描述该模型。可以借鉴例 7.4 或者例 7.5 的行动空间。第 12.3.2 节介绍的 Stentz 算法[914]为

该问题生成最优导航规划，甚至假设在开始不知道地形的情况下。最优加权区域规划问题的理论界限由文献[712,713]给出。一个近似算法出现在文献[823]中。 ■

7.7.2　多机器人的最优性

假设如图 7.42 所示有两个机器人。有足够大的房间使得机器人能够沿着走廊平移。期望机器人到达如箭头所指的底部；但是在一个时刻只有一个机器人可以通过水平走廊。假设在任意时刻，每个机器人都可以“开”或者“关”。当机器人“开”时，它以最大速度运动，当“关”时，它就停止[1]。现在假设每个机器人期望尽快地到达目标。这意味着每个机器人将最小化“关”的次数。在这个例子中可能出现两个有意义的选择：（1）$\mathcal{A}_1$“开”，并直接向目标运动，同时$\mathcal{A}_2$“关”着，直到$\mathcal{A}_1$通过以后才开始向它的目标运动；（2）相反的过程发生，即$\mathcal{A}_2$“开”而$\mathcal{A}_1$等待。当一个机器人等待时，有多个等待的位置，并且产生相同的到达时间。重要的是机器人“关”多久。

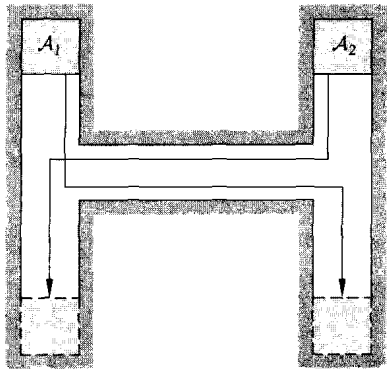


图 7.42　根据哪个机器人不得不进行等待，这个问题有两个 Pareto 最优协调规划

因此，两个有趣的规划是$\mathcal{A}_2$“关”的一段时间$t_{off}>0$，或者$\mathcal{A}_1$“关”的一段时间t_{off}。考虑一个(L_1,L_2)形式的代价向量，其中每个分量表示一个机器人的代价。规划的代价可以用等待的时间来衡量。对应于这两个规划的代价向量分别为$(0,t_{off})$和$(t_{off},0)$（等价地，我们可以用每个机器人运动的总时间来定义代价；每个机器人“开”的时间相同，可以从每个机器人总时间中减去）。这两个规划比其他规划更好，或者等价。有这种性质的规划称为 Pareto 最优（或者非占优的）。例如，如果$\mathcal{A}_2$再多等 1 秒，那么所产生的代价是$(0,t_{off}+1)$，它明显比$(0,t_{off})$差，那么所产生的规划不是 Pareto 最优的。关于 Pareto 最优的更详细的内容将出现在第 9.1.1 节。

另外一个解决问题的方法是通过将代价映射为单个值来进行量化。例如我们可以寻找优化平均消耗时间的规划。在这种情况下，可以得到两个最好规划之一，所产生的平均消耗时间是t_{off}。但是关于机器人的损失的信息却没有保留。量化代价通常加入某种人为的性能指标或者机器人的优先级。最后仅选择一个规划，保留多个规划解似乎是不合适的。但是发现和提出可选的 Pareto 最优解却能够提供有价值的信息，例如，如果这些机器人被卷入一个涉及许多其他时序过程的复杂应用中。Pareto 最优解等价于抛弃所有更差的规划而选择最好的。在某些应用中，机器人之间的优先级可能发生改变，如果机器人的调度程序能够访问 Pareto 最优解，那么很容易通过在 Pareto 最优规划之间进行切换来改变优先级，而不需要每次都生成新的规划。

现在更精确和更一般情况下描述 Pareto 最优性概念。假设有 m 个机器人$\mathcal{A}^1,\cdots,\mathcal{A}^m$。

1　这个模型允许有无限加速度。假设速度足够慢以允许这种近似。如果这还不能令你满意，那么可跳至第 13 章。

令 γ 表示一个给出所有机器人的路径和定时函数的运动规划。对于每个 $\mathcal{A}^i$，令 L_i 表示它的代价泛函，对给定的规划 γ，该泛函产生一个值 $L_i(\gamma)\in[0,\infty]$。一个 m 维向量 $L(\gamma)$ 定义为：

$$L(\gamma)=\left(L_1(\gamma),L_2(\gamma),\cdots,L_m(\gamma)\right) \tag{7.28}$$

两个规划 γ 和 γ' 称为等价的，如果 $L(\gamma)=L(\gamma')$。一个规划 γ 称为比 γ' 占优，如果对于所有 i，$1\leqslant i\leqslant m$，它们不等价而且 $L(\gamma)\leqslant L(\gamma')$。一个规划称为 Pareto 最优的，如果它没有被其他规划占优。因为一些 Pareto 最优规划可能是等价的，那么任务是从每个等价类中确定一个代表。这称为发现 Pareto 最优规划。对于图 7.42 中的例子，有两个 Pareto 最优规划，这在前面已经给出了。

标量化 对于运动规划问题来说，一个 Pareto 最优解对于由多个代价线性组合而成的一个标量代价泛函来说也是最优的。令 $\alpha_1,\cdots,\alpha_m$ 是正实常数，且令：

$$l(\gamma)=\sum_{i=1}^{m}\alpha_i L_i(\gamma) \tag{7.29}$$

容易看到：关于式（7.29）最优的任何规划也是一个 Pareto 最优解[609]。如果这样生成一个 Pareto 最优解，那么就不易得到其他可选规划。

计算 Pareto 最优解 因为一个机器人的优化已经非常困难，所以计算 Pareto 最优解更难就不奇怪了。对于某些问题，甚至可能存在 Pareto 最优解的一个连续统（见例 9.3），这是令人沮丧的。幸运的是，对于在第 7.2.2 节中考虑的拓扑图上协调机器人的问题，只有有限数量的解[388]。文献[609]提出一种基于栅格的算法，算法基于动态规划，它计算所有 Pareto 最优解协调规划。对于在分段线性的路径的树上运动的两个多边形机器人这种特殊情况，文献[214]提出了一个完备算法。

进一步阅读

本章介绍了一些对基本运动规划问题的最直接的扩展。扩展涉及整个第III部分所介绍的不确定性，所介绍的对运动规划的微分约束是第IV部分的核心内容。其他大量的扩展可以从关于机器人研究的出版物或者网络上搜索到。

第 7.1 节中的时变运动规划假设所有运动都是可预测的。多数的内容是基于以前的工作[155,509,821,822]；其他相关工作包括文献[369,370,535,815,877,906]。为了引入不确定性，请参见第 10 章。第 2.4 节的基于逻辑的表示已经扩展到时间逻辑以允许离散规划问题的时变情况（见文献[384]中的第IV部分）。

对于更多多机器人运动规划，参见文献[15,34,41,322,347,366,410,609,783,887]。与此密切相关的是可重配置的机器人的规划问题[182,211,387,54,990]。在这两种情况下，非正曲率(NPC)是一个重要条件，它能够极大地简化最优路径的结构[141,387,388]。对于在拓扑图上运动的点来说，文献[5]描述了 C_{free} 的拓扑。近年来，人们对团体机器人或者群体机器人之间的协调合作的研究产生了浓厚的兴趣[167,230,276,302,307,338,363,668]。

文献[400,518,736,972]研究了装配规划的复杂性。问题是一般 NP-复杂度的；但是对于某些特殊情况，则开发了多项式时间算法[9,432,973,974]。其他工作包括文献

[188,430,456,461,545]。

在过去的几十年间，混合系统受到广泛的关注。大多数这类工作都考虑如何为分段平滑系统设计控制律[139,636]。早期的组合控制的文献出现在文献[411]中。第 7.3.2 节的操作规划架构是基于文献[16,17,168]的。本章所提及的操作规划架构忽略了抓握这种情况。关于抓握问题的分析和算法，参见文献[258,490,684,784,802,803,829,830,921]。文献[128]考虑了在微小尺度上进行的操作。

第 7.4 节以外的关于闭运动链的基于采样的规划，参见文献[246,248,435,979]。文献[699]提出了对某些闭链的一个完整规划器。与逆运动学有关的工作，参见文献[311,696]。文献[160]显示了机器人系统中冗余自由度的操作。

第 7.5 节是对多个应用的分析。运动规划技术应用于计算生物学是一个飞速发展的领域；一些有代表性的文章见文献[24,25,33,247,517,592,604,657,1001]。节点规划覆盖是基于文献[575]的。折叠箱子的内容是基于文献[664]的。文献[66]提出了进行折纸的一个机器人系统和规划技术。

第 7.6 节中的覆盖规划方法是基于文献[224]和文献[374,375]的。对覆盖规划的概述出现在文献[219]中。其他参考文献包括文献[6,7,166,376,447,471,976]。对离散环境中最优地访问一个目标集合（定向越野问题）中的所有状态这个问题，文献[117,190]提出并分析了许多近似算法。

超过二维的最优运动规划非常复杂。基于动态规划的近似方法见第 8.5.2 节。$\mathbb{R}^3$ 中近似最短路径见文献[217,766]。文献[712,713]考虑了加权区域问题。文献[214,388,609]考虑了 Pareto 最优协调问题。

习　题

1. 考虑时变运动规划的状态空间中的障碍区域，见式（7.1）。

（a）为了确保 X_{obs} 是多面体的，那么应该允许什么类型的路径？阐述用于定义 $\mathcal{O}$ 的模型本原 H_i 在一般情况下是如何表示的，用 t 作为参数。

（b）重复（a），只是为了确保 X_{obs} 是半代数的。

2. 提出方法来适应第 5.6 节的基于采样的路线图算法，使之能够解决有速度限制的时变运动规划问题。

3. 两个正在平移的多边形机器人，每个机器人都有一个线性路径，设计一个高效的算法来计算它们的障碍区域。

4. 沿着图 7.43 所示的固定路径运动的两个机器人，简述它们的协调空间。

5. 假设有两个机器人，每个机器人在它自己的三条路径的路线图上运动。每个路线图中的路径在一个三角形中首尾相连。

（a）刻画所产生的固定路线图协调空间的特征，包括对其拓扑的一个描述。

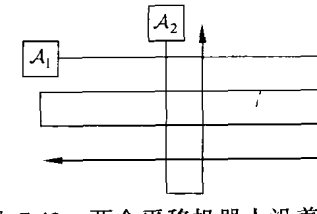


图 7.43　两个平移机器人沿着分段线性的路径运动

（b）现在假设有 n 个机器人，每个三角形路线图上有一个机器人，刻画固定路线图协调空间的特征。

6. 考虑 n 个相同机器人的 C-空间的笛卡儿积所产生的状态空间。假设每个机器人用唯一整数来标注。试说明 X 可以分拆为 $n!$ 个区域，其中 X_{obs} 都是相同的，只有标号不同（标号指示处在冲突中的特定机器人）。

7. 假设有两个机器人，每个机器人在它自己的三条路径的路线图上运动。每个路线图中的路径在一个三角形中首尾相连，其他路线图中的路径被安排为一个 Y 形。刻画所产生的固定路线图协调空间的特征，包括对其拓扑的一个描述。

8. 设计一个高效的算法，它将连杆组的连通性的一个图作为输入并计算一个主动-被动分解。假设所有连杆由旋转关节相连。算法应该能够应用于二维或者三维连杆组（维数也是一个输入）。确定你的算法的渐近运行时间。

9. 考虑在有可预测的运动障碍物的环境中，两个沿着预先计算的路径运动的机器人的协调问题。为该问题设计一个规划算法。

10. 考虑 $\mathcal{W}=\mathbb{R}^2$ 中的一个操作臂，它是由旋转关节连接成链的 4 个连杆组成。关节之间的距离是单位距离，第一个关节固定在 $\mathcal{W}=\mathbb{R}^2$ 中的(0,0)处。假设最后一个连杆在其体坐标系中安置在(1,0)，其末端固定在 $(0,2)\in\mathcal{W}$。

（a）用运动学表达一个位形 $q\in C$ 的封闭约束。

（b）将封闭约束转化为多项式形式。

（c）用微分法来确定对在一个位形 $q\in C$ 上保持封闭的可容许速度的约束。

实现

11. 实现垂直剖分算法来解决路径调节问题，如图 7.5 所示。

12. 用基于栅格的采样和一个搜索算法来计算 3 个沿着预先确定的路径运动的机器人的无碰撞运动。

13. 在习题 12 的条件下，计算最优化每个机器人到达其目标所用时间（阶段数）的 Pareto 最优协调策略。设计一个波前传播算法，算法跟踪每个所到达的状态上最小 Pareto 最优协调策略。要避免存储每个离散化的状态上的全部规划。

14. 为了得到对闭运动链规划的困难程度的评估，尝试仅用式（6.40）隐式环面约束来进行环面上一个点在障碍物中的运动规划。为了简化碰撞检测，障碍物可以是在与环面相交的 $\mathbb{R}^3$ 中球的一个族。适应一个诸如双向 RRT 的采样规划技术来穿越环面并求解规划问题。

15. 实现第 7.6 节的生成树覆盖规划算法。

16. 设计一个 RRT 规划算法，它使机器人在一个包含障碍物的平面环境中追逐一个不可预测的移动目标。算法的速度应该足够快以使在执行过程中重新进行规划。机器人应该执行最近计算的路径的第一部分，同时为下一个时间增量计算更好的规划。

17. 修改习题 16，以使机器人跟踪一个可预测的、常值速度轨迹直到出现某偏差。

18. 说明如何使用一个足够快的规划算法来处理未预料到的障碍物。为简单起见，假设机器人是一个在一个多边形障碍区域中运动的点。机器人首先计算一条路径并开始执行

之。如果障碍区域发生变化，那么从机器人目前的位置计算一条新的路径。用垂直剖分或者另外一种你所选择的算法（只要它足够快）。在规划执行过程中，用户应该能够交互式地放置或者移动障碍物。

19. 用第 7.3.2 节的操作规划架构来设计一个算法，载有圆环的机器人用它来解决著名的 Hanoi 古塔问题[168]。为简单起见，假设一个多边形机器人在 $\mathcal{W}=\mathbb{R}^2$ 中移动多边形零件，并且不允许旋转。制造 3 个桩，初始时所有的零件都放在一个挂钩上，按照从大到小进行排序。目标是将所有零件移动到另外一个挂钩上，同时保持排序不变。

20. 用基于栅格的近似方法来解决平面中一个点机器人的最优规划问题。用不同邻域和度量进行试验。在不同组合情况下能够得到好的近似和坏的近似，刻画不同组合的特点。

第 8 章　反馈运动规划

到目前为止的第 II 部分中都假设一条连续路径足够解决一个运动规划问题。在诸如计算机生成的动画和虚拟造型等许多应用中，没有必要挑战这些假设，因为虚拟环境中的模型通常的行为是设计好的。在涉及与物理世界相互作用的应用中，将来的位形可能是不可预测的。在机器人学中考虑这种情况的传统的方法是使用如图 1.19 所示的改进策略来设计一个反馈控制律，它试图跟踪所计算的路径。这样做有时是符合要求的，但重要的是要认识到这个方法是分离的。反馈和动力学在原始路径的构造中是被忽略的；计算的路径因此可能是没有用的。

第 8.1 节启发了在运动规划中对反馈的思考。第 8.2 节提出了本章主要的概念，但是仅为离散状态空间情况。这对数学的要求比连续空间要少，也更容易提出反馈概念。第 8.3 节提出将反馈的概念扩展到连续状态空间（它包括 C-空间）所需要的数学背景。反馈运动规划方法分为第 8.4 节的完备方法和第 8.5 节的基于采样的方法。

8.1　目　　的

对涉及物理世界的大多数问题，都需要某种形式的反馈。这意味着一个规划的行动应该以某种方式依赖于在执行过程中采集的信息。对反馈的需求来自于未来状态的不可预测性。本章的状态空间或者是离散的，或者 $X = C$，它是第 4 章所考虑的一个位形空间。

在未来状态的可预测性中建模不确定性的两种一般方法是：

（1）显式的：建立模型显式地考虑实际未来状态偏离规划未来状态的可能方式。规划算法必须直接考虑这种不确定性。第III部分将引入这种不确定性的显式模型，并将其融入规划模型中。

（2）隐式的：状态转移模型指出可能不存在不确定性；但是构造反馈规划以确保在执行过程中正好处在某种不期望的状态时它知道该采取什么行动。本章将采用这种方法。

处理这种不确定性的隐式方法开始似乎有些奇怪；因此需要一些解释。在第III部分读者将会看到，显式建立不确定性模型是非常复杂和具有挑战性的。表达可靠模型的要求是很强的；问题的复杂性增加会使得算法设计更加困难，并导致大量产生建模误差的机会。处理不确定性的隐式方法来自于控制论[110,124,689]。我们知道，得到可靠性能需要反馈控制律，但奇怪的是大多数所使用的对动力学的表述都没有显式考虑不确定性。经典控制论总是假设反馈是至关紧要的；但是仅在诸如随机控制和鲁棒控制等现代控制论中才显式地对不确定性进行建模。因此人们进行了广泛而且有效地设计反馈控制律的工作，反馈控制采用状态反馈来隐式地考虑这样一个事实：将来的状态是不可预测的。在过去的一个世纪里的大量应用中，这种控制方法非常成功，在运动规划中利用这种原理也非常有价值。

由于历史原因，在反馈控制发展中反馈和动力学似乎是不可分的。这主要因为控制论

的发展是为了可靠地改变动力系统的行为。在传统的运动规划中，没有考虑反馈和动力学。一条解路径是开环的，即在执行过程中没有反馈信息来构成环路。没有处理动力学是因为这会导致出现微分约束和高维相空间（见第Ⅳ部分）。

将历史放在一边，并且将反馈从动力学中分离出来，这样可以产生四个独立的课题，如图 8.1 所示。涉及动力学的开环规划在近年来受到越来越多的关注。这是第Ⅳ部分的焦点。那些喜欢经典控制论的人可能会抱怨开环方法没有考虑反馈；但是这种开环轨迹（相空间中的路径）在涉及仿真的应用中十分有用，并且，在解耦方法中考虑了动力学的轨迹比忽略动力学的路径更有价值，在过去几十年中一直都忽略动力学。这些问题将在第Ⅳ部分中进一步详细描述。

	开环	反馈
自由运动	传统运动规划	第 8 章
动力学	第 14 章和第 15 章	经典控制理论

图 8.1　将动力学问题从反馈中分开，会出现两个尚未得到充分研究的课题

在图 8.1 中出现的其他有趣课题是为那些没有对动力学和其他微分约束进行显式建模的问题而发展反馈规划。如果采取忽略微分约束的方式来解决经典运动规划中的问题是合理的，那么我们仍然通过忽略微分约束来设计反馈运动规划就不会感到心虚[1]。这里用到不确定性的隐式模型，而不改变任何其他在基本运动规划中所到用的假设。

即使没有不可预测性的问题，反馈规划还可以应用于初始状态未知的问题中。一个反馈规划指示从每个状态应该采取什么行动。因此初始条件并不重要。在图算法中与此相似的是单源最短路径问题，该问题指示如何从任何其他顶点最优地到达一个特定顶点。由于这种关系，在将该思路扩展到运动规划所需要的连续空间之前，下一节将为离散状态空间提出反馈的概念。

由于这些原因，本章考虑反馈运动规划。作为机器人学中使用的解耦方法的一个模块，反馈运动规划至少与用以前方法计算出来的一条路径一样有用。当反馈解在用传统运动规划算法计算的开环路径的地方使用时，我们预料在一般情况下反馈解会更加可靠。

8.2　离散状态空间

本节有助于说明以后出现的相似的概念。这里仅限于离散空间，离散空间易于表述和理解。以此为基础可以扩展到位形空间和其他连续状态空间。这里的讨论也为即将在第 8.4.1 节介绍的反馈规划概念提供相关背景。在这种情况下，将显式建模不确定性。所产生的表述和概念可以认为是对本节的扩展。

8.2.1　反馈规划

考虑一个与表述 2.1 和表述 2.3 相似的离散规划问题，除了初始状态未知以外。因此代

1　第 8.8.4 节将考虑某些简单的微分约束，如加速度界；我们将在第Ⅳ部分详尽讨论微分约束。

价泛函不能仅表示为规划的函数，而是定义在历史状态和历史行动的基础之上。在 k 阶段，相应地定义为：

$$\tilde{x}_k = (x_1, x_2, \cdots, x_k) \tag{8.1}$$

$$\tilde{u}_k = (u_1, u_2, \cdots, u_k) \tag{8.2}$$

有时将 $\tilde{x}_k$ 称为状态轨迹会更方便。

所产生的表述如下所示。

表述 8.1（离散最优反馈规划）

1. 一个有限、非空状态空间 X。

2. 对每个状态 $x \in X$，有一个有限行动空间 $U(x)$。

3. 一个状态转移函数 f，对每个 $x \in X$ 和 $u \in U(x)$，该函数产生一个状态 $f(x,u) \in X$。令 U 表示 $U(x)$ 的并集，对所有 $x \in X$。

4. 阶段的一个集合，用 k 表示，k 从 $k=1$ 开始，无限进行下去。

5. 目标集合 $X_G \in X$。

6. 令 L 表示一个加性的代价泛函：

$$L(\tilde{x}_F, \tilde{u}_K) = \sum_{k=1}^{K} l(x_k, u_k) + l(x_F) \tag{8.3}$$

其中 $F = K+1$。

与第 2 章的表述相比较，有一个不同之处。在这里假设状态空间是有限的。这有利于构造反馈规划，但是一般情况下这是不必要的。

考虑定义一条解决表述 8.1 的路径。如果给定初始条件，那么可以指定一个行动序列，如第 2 章所述。如果没有初始条件，那么一个可能的方法是为每个可能的初始状态 $x_1 \in X$ 确定一个行动序列。一旦给出初始状态，那么相应的行动序列就可以知道。但是这种方法耗费内存。假设某状态 x 作为初始状态而给定，并且采取第一个行动，导致下一个状态 x'。那么对状态 x' 应该采取什么行动？当然可以采取 x 的序列中的第二个行动；但是我们可以假设 x' 是当前的初始状态，对该状态应当采取它的第一个行动。这指出为每个状态保持一个行动序列是非常多余的。对每个状态仅保持行动序列中的第一个行动就足够了。应用该行动产生下一个状态，保存该状态的下一个相应的行动。从一个如下初始状态可以假设存在一个执行序列。从某状态开始，应用所存储的行动，到达另外一个状态，应用该状态的行动，到达下一个状态，如此进行直到到达目标。

因此将一个反馈规划表示成一个将每个状态映射到一个行动的函数似乎是合适的。因此一个反馈规划 π 被定义为一个函数 $\pi: X \to U$。从每个状态 $x \in X$，规划指出采取哪个行动。如果到达目标，那么应该采取终止行动。终止行动是规划的一部分：$\pi(x) = u_T$，如果 $x \in X_G$。一个反馈规划将成为问题的解，如果它使得从每个可达状态都能够到达目标。

如果给定一个初始状态 x_I 和一个反馈规划 π，那么可以确定历史状态和行动。这样也就可以确定式（8.3）的执行代价。因此代价也可以用 $L(\pi, x_1)$ 来代替 $L(\tilde{x}_F, \tilde{u}_K)$。这要求未来的状态总是可预测的。在第 10 章中，由于存在不确定性，建立这种直接对应关系将是不可能的。

可行性与最优性 在反馈规划中需要重新考虑可行规划和最优规划的概念，因为初始

条件未知。一个规划 π 成为可行规划问题的解，如果从每个使 X_G 是可达的 $x \in X$、执行 π 都确实可以到达目标集合。这意味着忽略了代价泛函（可以定义表述 8.1 的另外一种形式，其中去掉了代价泛函）。为方便起见，π 被称为可行反馈规划。

现在考虑最优性。从一个给定的状态 x，很显然存在使用第 2.3 节的概念所定义的一个最优规划。是否可能：每个可以到达 X_G 的 $x \in X$ 都需要一个不同的最优规划？实际情况是仅需要一个规划来对从每个初始状态到达 X_G 的最优路径进行编码。为什么？假设在 X 上用第 2.3 节的 Dijkstra 算法或者值迭代法计算最优尚需代价。某 $x \in X$ 的每个尚需代价值表示了在进行 x 的最优开环规划时所付出的代价。这种最优规划中的第一步可以用式（2.19）确定，第一步产生一个新的状态 $x' = f(x,u)$。从 x'，再次使用式（2.19）确定下一个最优行动。那么 x' 的代价既表示如果 x' 是初始状态时的最优尚需代价，又表示如果继续 x 的最优路径时的最优尚需代价。因为动态规划原理的缘故，它们两者必须是等价的。因为所有这种代价必须是一致的，所以用一个反馈规划就可以得到每个初始条件下的最优尚需代价。

一个反馈规划 π 因此被定义为最优的，如果从每个 $x \in X$，在所有可能的规划中，执行 π 所得到的总的代价 $L(\pi,x)$ 是最低的。这要求对每个初始条件都必须成立，这对反馈规划来说是非常重要的。

例 8.1（二维栅格上的反馈规划） 这个例子使用例 2.1 的二维栅格模型。一个机器人在栅格上运动，它可能的行动有向上（↑）、向下（↓）、向左（←）、向右（→）以及终止（u_T）；某些方向对某些状态来说是不可行的。一个反馈规划解如图 8.2 所示。也存在一些其他可能的路径解。这里显示的恰好是到达目标所用步数最优的一个。还有一些反馈规划也是最优的。用任何初始状态的规划，只需顺着箭头方向就可以到达目标。在每个阶段，用箭头所代表的行动就可以产生下一个状态。当在目标处采取 u_T 行动时，过程就会终止。 ■

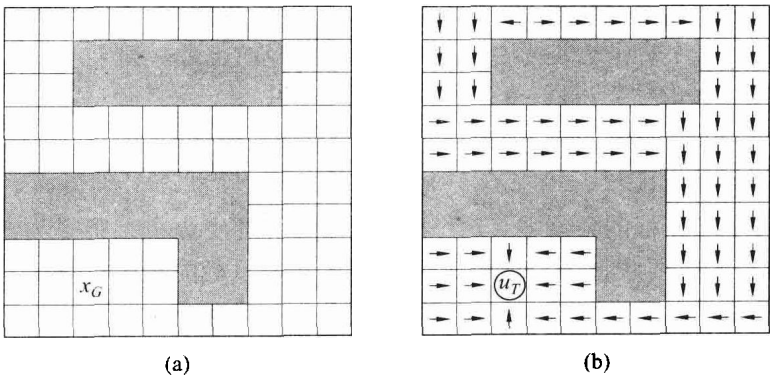


图 8.2 （a）一个二维栅格规划问题；（b）反馈规划的一个解

8.2.2 将反馈规划作为导航函数

第 2 章已经给出了计算反馈规划的工具。诸如 Dijkstra 算法和值迭代等方法产生可以用来表示反馈规划的附加信息。本节将说明这些信息是如何转化为反馈规划的。为此，一个反馈规划将表示为状态空间上的一个势函数（回忆第 5.4.3 节的势函数）。规划算法计算

势函数的值，这些值用来发现执行过程中相应的行动。在某些情况下，最优反馈规划甚至可以用势函数表示。

导航函数 考虑一个（离散）势函数，定义为 $\phi: X \to [0,\infty]$。势函数通过局部算子来定义一个反馈规划，局部算子是一个函数，它选择的行动能够尽可能减小势函数。首先考虑可行规划问题。势函数 ϕ 定义一个反馈规划，它选择 u 是通过局部算子：

$$u^* = \operatorname*{arg\,min}_{u \in U(x)} \{\phi(f(x,u))\} \tag{8.4}$$

它意味着选择 $u^* \in U(x)$ 来尽可能减小 ϕ。局部算子产生势函数的一种“贪婪”下降。需要注意的是行动 u^* 可能不唯一。在与此相似的连续空间中，相应的局部算子实现沿着负梯度方向的下降（通常称为梯度下降）。

在最优规划情况下，局部算子定义为：

$$u^* = \operatorname*{arg\,min}_{u \in U(x)} \{l(x,u) + \phi(f(x,u))\} \tag{8.5}$$

它看起来与动态规划条件式（2.19）相似。如果 ϕ 阐述为最优尚需代价，那么它与式（2.19）相同。如果规划问题是各向同性的，即在每个方向上代价是相同的：对于所有 $u, u' \in U(x) \setminus \{u_T\}$，有 $l(x,u) = l(x,u')$，那么式(8.5)可以被简化。在这种情况下，代价项 $l(x,u)$ 不影响式（8.5）的最小化。应用该假设的一个平常的例子是代价泛函是否对到达目标所需要的步数进行计数。该方法选择特别行动所花费的代价并不重要。应用各向同性的特性，那么式（8.5）可以被简化为式（8.4）。

势函数什么时候有用呢？可以定义不能到达目标，或者引起状态无限循环等一些无用的势函数。最理想的势函数是：如果 X_G 是可达的，那么该势函数应该使任何初始状态都能够到达 X_G。这只要求一些简单的特性。满足这些特性的势函数被称为导航函数[1]。

假设代价泛函是各向同性的。令 $x' = f(x,u^*)$，它是应用式(8.4)所选择的行动 $u^* \in U(x)$ 以后到达的状态。一个势函数 ϕ 被称为是（可行的）导航函数，如果：

（1）对于所有 $x \in X_G$，$\phi(x) = 0$。

（2）$\phi(x) = \infty$，当且仅当 X_G 中的点对 x 都是不可达的。

（3）对于每个可达状态 $x \in X \setminus X_G$，局部算子产生一个状态 x' 使得 $\phi(x') < \phi(x)$。

第一个条件要求目标的势为零（这个条件实际上是不必要的，但是为方便起见将它包括进来）。第二个条件要求将 ∞ 作为从某状态不能到达目标的一个特殊指示器。第三个条件意味着势函数除了目标以外没有其他局部最小值。这意味着执行所产生的反馈规划的过程将不会有循环，最终一定能到达目标区域。

一个最优导航函数被定义为最优尚需代价 G^*。即除了以上三个特性以外，导航函数还必须满足最优性的原理：

$$\phi(x) = \min_{u \in U(x)} \{l(x,u) + \phi(f(x,u))\} \tag{8.6}$$

它仅仅是将式（2.18）中的 G^* 用 ϕ 代替。更多内容参见第 15.2.1 节。

例 8.2（二维栅格上的导航函数） 回到例 8.1 的规划问题。假设采用一个各向同性的

1 这个术语是为文献[543,831]中的连续位形空间而设计的；它被广泛地应用在本书中，但是仍保持其基本含义。

代价模型：如果 $u \neq u_T$，那么 $l(x,u)=1$。图 8.3 显示了一个导航函数。格子上显示的数字代表 ϕ。可以验证 ϕ 满足对导航函数的 3 个要求。

在任何状态下，都应采取能够减小势值的行动。这对应于使用式（8.4）所选择的行动。过程重复直到到达 X_G。本例清楚说明了如何将一个导航函数作为反馈规划的另外一种定义来使用。 ■

例 8.3（机场候机厅） 当到达一个陌生的机场候机厅，你会发现你自己正在使用导航函数来寻找出口。一些候机厅是树形结构的，当与出口的距离增加时，门牌号增加。如果你希望离开候机厅，那么你应该向更小门牌号的方向移动。 ■

22	21	22	21	20	19	18	17	16	17
21	20							15	16
20	19							14	15
19	18	17	16	15	14	13	12	13	14
18	17	16	15	14	13	12	11	12	13
							10	11	12
							9	10	11
3	2	1	2	3			8	9	10
2	1	0	1	2			7	8	9
3	2	1	2	3	4	5	6	7	8

图 8.3　将尚需代价值作为一个导航函数

计算导航函数　有多种方法计算导航函数。Dijkstra 算法从 X_G 向后进行，它所确定的尚需代价函数产生了一个最优导航函数。如果导航函数 ϕ 定义为最优尚需代价 G^*，那么在各向同性情况下，导航函数的第三个条件就是动态规划方程（2.18）。以前曾经提到：仅用尚需代价就可以得到最优行动。这实际上就是一个使用导航函数的例子，所产生的过程已经被认为是一个反馈规划。

如果最优性并不重要，那么就可以使用第 2.2 节的任何后向搜索算法，只要它能够记录每个所到达的状态与目标之间的距离。距离不必是最优的。它仅仅对应于搜索树中从目前顶点向根顶点回溯所得到的代价（或者向 X_G 中的任何顶点回溯，如果有多个目标状态）。

如果规划问题像表述 2.1 那样没有包括代价泛函，那么为了构造一个导航函数，可以引入一个代价泛函。在能够到达 X_G 的每个 $x \in X$，搜索图中从 x 到 X_G 所需要穿越的边的数量可以作为代价加以存储。如果用 Dijkstra 算法来构造导航函数，那么按照到达目标所需要的步数而言，反馈规划执行过程是最短的。

通过局部算子来寻找行动，导航函数本身就可以用来表示反馈规划。因此从一个导航函数 $\phi: X \to [0,\infty]$ 可以发现一个函数 $\pi: X \to U$。同样地，从 π 也可以构造一个导航函数 ϕ。因此 π 和 ϕ 可以认为是反馈规划的两个可互换的表示。

8.2.3　运动规划的基于栅格的导航函数

为了考虑连续空间的反馈规划，需要微分几何的向量场和其他基本定义。这些将在第 8.3 节介绍；但是在此之前，我们首先描述如何将第 8.2 节目前提出的思路作为对反馈运动规划的一个离散近似来使用。

例 8.1 和例 8.2 已经为包含障碍物的二维栅格定义了反馈规划和导航函数。假设该模型用来近似 $C \subset \mathbb{R}^2$ 中的一个运动规划问题。第 5.4.2 节显示了如何产生用采样点栅格来近似运动规划问题的拓扑图。例 8.1 中用到的运动对应于 1-邻域的定义（见式（5.37））。该思路在第 7.7.1 节得到进一步改进后就可以通过在栅格上运动来建模近似最优运动规划（见表述 7.4）。选择例 7.4 定义的 Manhattan 运动模型，产生与例 8.1 中运动相同的一个栅格。

为了构造在移动机器人中非常有用的导航函数，通常要求一个高分辨率的栅格（例如每个轴有 50～100 个点）。在第 5.4.2 节中，因为没有考虑反馈，每个轴仅需要几个点。在

某些情况下，通过每个轴的几个点发现一条无碰撞的路径是可能的。在反馈规划执行过程中，假设机器人未来的状态不一定是可预测的。在机器人所停止的地方，导航函数和局部算子必须产生合适的行动。如果目前的状态（或者位形）用栅格近似，那么尽可能减小近似误差是非常重要的。这可以通过设置高分辨率来实现。在反馈情况下，可以认为栅格“覆盖”整个位形空间，而第 5.4.2 节的栅格仅代表穿过该空间的路径的一个拓扑图[1]。

波前传播算法 一旦进行了近似，第 8.2.2 节讨论的任何方法都可以用于计算导航函数。从目标用 Dijkstra 算法很容易计算一个最优导航函数。如果每个运动都有单位代价，那么可以进行简化。图 8.4 描述了一个计算最优导航函数的波前传播算法。它可以认为是 Dijkstra 算法的特殊形式，它没有显式构造优先级队列。在 Dijkstra 算法执行过程中，队列中最小元素的代价是单调非减的。在每个运动有单位代价的情况下，队列中的一些状态有相同的代价。Dijkstra 算法能够删除所有代价相同且代价最小的元素。假设公共的、最小的代价值是 i。这些状态被组织到一个波前 W_i。初始波前 W_0 表示 X_G 中的状态。对在一个阶段内从 W_0 中的任何状态所能够到达的每个状态，算法赋予一个最优尚需代价值 1。这些一定是最优的，因为没有其他的代价值是最优的。将代价赋值为 1 的状态可以组织到波前 W_1。将 W_1 的没有探索到的邻近点的代价值赋值为 2，它们也必须是最优的。这种过程从 i 到 $i+1$ 重复进行直到到达所有可达状态。最后最优尚需代价将在 $O(n)$ 时间内计算出来，其中 n 是可达栅格状态的个数。对于任何不能到达的状态，赋值 $\phi(x)=\infty$。图 8.3 所示的导航函数实际上可以用波前传播算法进行计算。

波前传播算法

1. 初始化 $W_0=X_G$；$i=0$。
2. 初始化 $W_{i+1}=\phi$。
3. 对于每个 $x\in W_i$，赋值 $\phi(x)=i$ 并将 x 的所有未探索过的相邻点插入 W_{i+1}。
4. 如果 $W_{i+1}=\phi$，那么算法结束；否则，令 $i:=i+1$，并返回到第 2 步。

图 8.4 波前传播算法是一种特殊形式的 Dijkstra 算法，它将优化到达目标所用的阶段数

最大间隙 在可移动机器人中存在的一个典型问题是最优运动规划将机器人带到距离障碍物太近的地方。回忆第 6.2.4 节，在多边形环境中允许运动规划的最短欧氏路径接触障碍物的顶点。这产生了第 6.2.3 节所介绍的最大间隙路线图。这可以产生最大间隙路线图的一个基于栅格的近似版本，并且定义的导航函数可以将机器人引导到路线图上，使之沿着路线图运动，最后使机器人停在一个特定的目标上。在文献[591]中，所产生的导航函数被称为 NF2。

假设只有一个目标状态 $x_G\in X$。计算最大间隙导航函数的过程如下：

（1）W_0 为所有这些状态的集合，它们至少有一个方向的运动被阻挡。这些是离散化的无碰撞空间的边界上的状态。

（2）进行波前迭代，迭代在从障碍物边界向外的波中传播代价。

（3）当波前进行传播时，它们在原始连续问题的最大间隙路线图的近似位置上相遇。将两个波前点从相反方向所到达的任何状态标示为骨架状态（skeleton state）。波前也可能

1 区分这两个的困难多年来使得研究人员相信对于第 5 章的开环路径规划问题，栅格产生很糟的性能。这主要是因为它需要一个高分辨率的栅格。而对于许多问题，栅格在每个轴上只有为数不多的一些格点。

只是互相接触，而并没有到达公共状态；在这种情况下，可选择两个相接触的状态之一作为骨架状态。令 S 表示所有骨架状态的集合。

（4）波前传播结束以后，通过将 x_G 以及沿着通向骨架状态的路径的所有状态都插入到 S，从而将 x_G 连接到骨架状态。这种路径可以用任何搜索算法来寻找。

（5）将所有其他状态当作障碍物并采用波前传播算法，计算 S 上的一个导航函数 ϕ_1。该导航函数将 S 中的所有点导引到目标。

（6）将 S 当作目标区域，用波前传播算法计算导航函数 ϕ_2。该导航函数将状态导引到骨架状态上的最近点。

（7）将 ϕ_1 和 ϕ_2 组合成 ϕ。对于每个 $x \in S$，令 $\phi(x) = \phi_1(x)$。对于每个其余状态，赋值 $\phi(x) = \phi_1(x') + \phi_2(x)$，其中 x' 是距离 x 最近的状态 $x' \in S$。当计算 ϕ_2 时，状态 x' 很容易被记录下来。

如果 C_{free} 被多次连接，那么绕着不同的障碍物运动可能有多条线路通往每个 x_G（路径不是同伦的）。上述方法没有考虑到一条线路的间隙可能比其他线路的间隙更小。所给出的方法只优化沿着骨架状态运动的距离；但是如果有多条线路，该方法并没有最大化与障碍物间最近的距离。

Dial 算法　现在考虑对波前传播思路进行推广。如果对任意 $x \in X$ 和 $u \in U(x)$（除了 $u = u_T$），有 $l(x,u) = 1$，那么波前传播可以应用于任何离散规划问题。当转移图稀疏时最有用（假设用一个邻接矩阵来表示转移图）。栅格问题是一个很有说服力的例子。更一般情况下，如果假设代价项是整数值，那么就产生了 Dial 算法[274]，它是波前传播的推广，也是 Dijkstra 算法的特殊形式。思路是通过将活的顶点分配给对应于不同尚需代价值的漏斗来避免优先级队列。在波前传播中，同一时刻从来不需要两个以上的漏斗。Dial 算法并行处理最小代价漏斗中的所有状态。文献[939]则加强了该思路，产生了一个线性时间的算法。

其他扩展　对本节的几个思路进行推广可以产生其他导航函数。到目前所讨论的方法的一个不足之处在于产生了不期望的阶梯运动（见图 7.40）。如果像式（5.38）那样用 2-邻域定义行动空间，那么运动一般比较短。在这种情况下，Dial 算法可以有效地计算最优导航函数。

对更高维数的位形空间可以采用栅格近似。但是因为需要高分辨率，所以它仅适用于不太高的维数（如 3 维或者 4 维）。如果采用 1-邻域，那么很容易用波前传播来计算导航函数。对于一般 k-邻域情况，Dial 算法可以进行适应性修改来计算导航函数。

在栅格上构造导航函数可以为一些应用提供一些非常实用的解。在其他情况下，产生阶梯运动是无法接受的。在一些应用中，不可能快速计算导航函数。影响该问题的因素有：（1）很高的精度，从而需要一个高分辨率的栅格；（2）位形空间的维数可能很高；（3）环境可能频繁改变，需要实时进行响应。为了处理这些问题，最好抛弃栅格近似。这要求直接在位形空间上定义势函数和速度。第 8.3 节为进行这种过渡而提出基本数学概念。

8.3　向量场与积分曲线

为了考虑在包括位形空间在内的连续状态空间上进行反馈运动规划，我们需要定义一个向量场以及通过从初始点对向量场进行积分而得到的轨迹。对于刻画连续状态空间上反

馈规划的特点，向量场是最合适的。它提供栅格上反馈规划在连续空间上的相似物，如图 8.2（b）所示。

本节提出两个可选的背景数学概念。第 8.3.1 节假设 $X=\mathbb{R}^n$，这样所产生的定义与你在基本积分和微分方程等课程中所学到的内容非常相似。第 8.3.2 节讨论流形上向量场的更一般的情形。这要求更多的理论概念，并建立在第 4.1.2 节流形的定义这个基础之上。

一些读者可能已经有一些微分流形的背景知识。但是如果你是第一次看见这些内容，那么可能很难理解这些内容。除了反复阅读之外，这里还有两个建议。首先，学习在本章结尾提出的这方面的背景材料。其次在接下来的几节中忽略流形的细节而认为 $X=C=\mathbb{R}^2$。没有更多细节的事情也同样是有意义的。假设一个流形被定义成一个有一些相同侧面的方体 $[0,1]^n$，如第 4.1.2 节那样。为 $\mathbb{R}^n$ 而提出的概念可以应用于除方体边界以外的任何地方。例如，如果 $\mathbb{S}^1$ 定义为 $[0,1]/\sim$，一个函数 f 定义在 $\mathbb{S}^1$ 上，那么我们如何定义 $f(0)$ 处的导数呢？第 8.3.2 节的定义能解决这个问题。有时在实际中通过处理等同点就可以避免这些技术细节。

8.3.1 $\mathbb{R}^n$ 上的向量场

本节回顾积分学、线性代数、微分方程等课程中的一些基本概念。你可能已经学习过与 $\mathbb{R}^2$ 和 $\mathbb{R}^3$ 相关的这些内容。我们最终是要描述 $\mathbb{R}^n$ 和流形上的速度，然后使用向量场的概念来表达第 8.4.1 节的反馈规划。

向量空间 在定义向量场之前，明确向量的含义是非常有益的。向量空间（或者线性空间）被定义为一个集合 V，它对于向量加和数乘这两种代数运算是封闭的，且满足稍后给出的公理。这里使用的向量空间是 $\mathbb{R}^n$，其中的标量是实数，向量用实数序列表示。数乘是指标量值与向量的每个分量相乘。向量加将两个向量的每个分量相加而形成了一个新的向量。

向量空间 V 可以定义在任何数域 $\mathbb{F}$ 上（回忆第 4.4.1 节中的定义）。数域 $\mathbb{F}$ 代表标量，V 代表向量。下面提出的概念是对我们所熟悉的向量空间 $\mathbb{R}^n$ 的推广。在这种情况下，$V=\mathbb{R}^n$，$\mathbb{F}=\mathbb{R}$。在下面的定义中，如果需要，你可以进行这种替换。我们不讨论比这更加一般化的向量空间；不过按照 V 和 $\mathbb{F}$ 所给出的定义已经清楚地将标量和向量分开。向量加用“+”表示，数乘用“·”表示。这些运算必须满足如下公理：

1. （向量加下的交换群）集合 V 关于向量加“+”是可交换群。
2. （数乘的结合律）对于任意 $v\in V$ 和任意 $\alpha,\beta\in\mathbb{F}$，有 $\alpha(\beta v)=(\alpha\beta)v$。
3. （标量和的分配律）对于任意 $v\in V$ 和任意 $\alpha,\beta\in\mathbb{F}$，有 $(\alpha+\beta)v=\alpha v+\beta v$。
4. （向量和的分配律）对于任意 $v,w\in V$ 和任意 $\alpha\in\mathbb{F}$，有 $\alpha(v+w)=\alpha v+\alpha w$。
5. （数乘的恒等式）对于任意 $v\in V$，它与幺元 $1\in\mathbb{F}$ 相乘，有 $1v=v$。

第一个公理允许向量以任何顺序相加。其余公理要求数乘以我们对 $\mathbb{R}$ 上的向量空间 $\mathbb{R}^n$ 所期望的方式与向量相互作用。

向量空间的基定义为向量的一个集合 $v_1,\cdots,v_n$，每个 $v\in V$ 都可以记为一个线性组合：

$$v=\alpha_1v_1+\alpha_2v_2+\cdots+\alpha_nv_n \tag{8.7}$$

对某些 $\alpha_1,\cdots,\alpha_n\in\mathbb{F}$。这意味着每个向量可以表示为基元素的一个线性组合。在 $\mathbb{R}^3$ 的情况下，一个熟知的基是[0 0 1]、[0 1 0]和[1 0 0]。所有向量可以表示为这三个向量的线性组合。需要记住的是，基并不是唯一的。在线性代数中，任何三个线性无关的向量都

可以是 $\mathbb{R}^3$ 的基。一般情况下，基只含有线性无关向量。即使基不是唯一的，基中向量的数量与该向量空间上其他可能的基中向量的数量也是相同的。该数量 n 称为向量空间的维数。因此我们称 $\mathbb{R}^n$ 为 $\mathbb{R}$ 上的 n 维向量空间。

例 8.4（$\mathbb{R}$ 上的 n 维向量空间 $\mathbb{R}^n$） $\mathbb{R}^n$ 可以认为是一个向量空间。一个自然的基是这样 n 个向量的集合，其中对于每个 $i \in \{1,\cdots,n\}$，一个单位向量是这样构造的。令 $x_i = 1$，对所有 $j \neq i$，$x_j = 0$。因为有 n 个基向量，$\mathbb{R}^n$ 是一个 n 维向量空间。基不是唯一的。任何 n 个线性无关向量的集合都是可以使用的，这在线性代数中是很常见的，其中非奇异 $n \times n$ 矩阵用来在向量之间进行变换。 ■

为了阐述一般向量空间定义的意义，我们考虑如下的例子。

例 8.5（函数的向量空间） 所有连续的实值函数 $f:[0,1] \to \mathbb{R}$，它的积分：

$$\int_0^1 f(x)\mathrm{d}x \tag{8.8}$$

是有限的，这样的函数形成了 $\mathbb{R}$ 上的一个向量空间。很容易验证它满足向量空间公理。例如，如果两个函数 f_1 和 f_2 相加，其积分仍然是有限的。此外 $f_1 + f_2 = f_2 + f_1$ 以及所有群的公理关于加法都满足。满足式（8.8）的任何函数都可以与 $\mathbb{R}$ 中的标量相乘，其积分仍然是有限的。数乘公理也可以得到验证。

这种向量空间是无限维的。一个方法是将函数限制在所有那些泰勒级数存在并收敛于该函数（称为解析函数）的集合上。每个函数都可以通过一个泰勒级数表示为一个可能有无穷多项的多项式。所有单项式 $x, x^2, x^3, \cdots$ 的集合代表了一个基。每个连续函数可以表示为一个有无穷多个系数的向量；每个系数乘以一个单项式就产生了该函数。这是函数空间的一个简单的例子；再进行一些定义，就导致了 Hilbert 空间，它在泛函分析中是极其重要的，泛函分析是刻画函数空间特性的一门学科[839,841]。 ■

本章的其余部分仅考虑 $\mathbb{R}$ 上的有限维向量空间。但是记住向量空间的基本特性仍是非常重要的。

向量场　向量场看起来像 $\mathbb{R}^n$ 上的一个“指针图”，如图 8.5 所示。思路是在每个点 $p \in \mathbb{R}^n$ 处指定一个方向。当用来表示一个反馈规划时，它指示机器人当发现它自己在 p 点时所需要运动的方向。 •

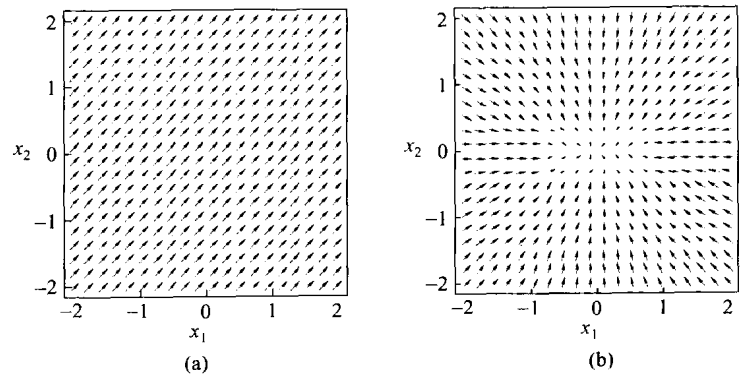


图 8.5　（a）一个常数向量场 $f(x,y) = [1 \quad 1]$；（b）一个向量场 $f(x,y) = [-1 \quad -1]$，其中所有向量都指向原点

对每个 $p\in\mathbb{R}^n$，都可以与一个 n 维向量空间相联系，该向量空间称为 p 点处的切空间（tangent space），记为 $T_p(\mathbb{R}^n)$。为什么不只是称之为 p 点处的向量空间呢？使用“相切的”这个词似乎有些奇特；它是受到流形的启发，在那里切空间与流形上的点是“相切的”。

$\mathbb{R}^n$ 上的向量场[1] $\vec{V}$ 是一个函数，它给每个 $p\in\mathbb{R}^n$ 赋予一个向量 $v\in T_p(\mathbb{R}^n)$。该函数的值域是什么？每个 $p\in\mathbb{R}^n$ 处的向量 $\vec{V}(p)$ 实际上是属于一个不同的切空间。因此函数的值域是：

$$T(\mathbb{R}^n)=\bigcup_{p\in\mathbb{R}^n}T_p(\mathbb{R}^n) \tag{8.9}$$

它被称为 $\mathbb{R}^n$ 上的切丛。即使用 $T_p(\mathbb{R}^n)$ 描述向量的方式与描述任意 $p\in\mathbb{R}^n$ 的方式相同，但是每个切空间产生不同的向量。为了保持这种差别，通过同时指定 p 和 v，切丛中的一个点可以用 $2n$ 个坐标表示。这对于第Ⅳ部分中定义相空间是非常重要的。在目前情况下，认为 $\vec{V}$ 的值域是 $\mathbb{R}^n$ 就足够了，因为对于每个 $p\in\mathbb{R}^n$，$T_p(\mathbb{R}^n)=\mathbb{R}^n$。

因此向量场可以用 $\mathbb{R}^n$ 上的 n 个实值函数表示。令 $f_i(x_1,\cdots,x_n)$ 表示这样的函数，i 从 1 到 n。这样一个向量场被确定为：

$$f(x)=\begin{bmatrix}f_1(x_1,\cdots,x_n) & f_2(x_1,\cdots,x_n) & \cdots & f_n(x_1,\cdots,x_n)\end{bmatrix} \tag{8.10}$$

在这种情况下，向量场是一个从 $\mathbb{R}^n$ 到 $\mathbb{R}^n$ 的函数。因此可以使用标准函数符号来表示一个向量场。

现在考虑 $\mathbb{R}^2$ 上的一些向量场的例子。令 $p=(x,y)$ 表示 $\mathbb{R}^2$ 上的一个点。在标准向量积分中，一个向量场通常表示为 $\begin{bmatrix}f_1(x,y) & f_2(x,y)\end{bmatrix}$，其中 f_1 和 f_2 是 $\mathbb{R}^2$ 上的函数。

例 8.6（常数向量场） 图 8.5（a）显示了一个常数向量场，它给每个 $(x,y)\in\mathbb{R}^2$ 分配向量 $[1\quad 2]$。■

例 8.7（向内流动场） 图 8.5（b）显示了一个向量场，它将给每个 $(x,y)\in\mathbb{R}^2$ 分配 $[-x\quad -y]$。这使得所有向量指向原点。■

例 8.8（旋涡） 图 8.6 的向量场将 $[(y-x)\quad (-x-y)]$ 赋予给每个 $(x,y)\in\mathbb{R}^2$。■

由于规划问题中存在障碍物，所以有时将一个向量场的定义域限制在 $\mathbb{R}^n$ 的一个开子集中是很方便的。因此对任何开子集 $O\subset\mathbb{R}^n$，可以定义一个向量场 $f:O\to\mathbb{R}^n$。

平滑 从 $\mathbb{R}^n$ 的子集到 $\mathbb{R}$ 的函数 f_i 称为平滑函数，如果关于任意变量、在 f_i 的定义域中任意点上存在任意阶导数。向量场被称为平滑的，如果它的 n 个函数 $f_1,\cdots,f_n$ 都是平滑的。平滑函数另外一个名字是 C^∞ 函数。上标表示可以进行的微分的阶数。对于一个 C^k 函数，它至少有 k 阶导数。一个 C^0 函数就是一个连续函数。同胚的概念可以扩展到微分同胚，微分同胚是一个同胚，它也是一

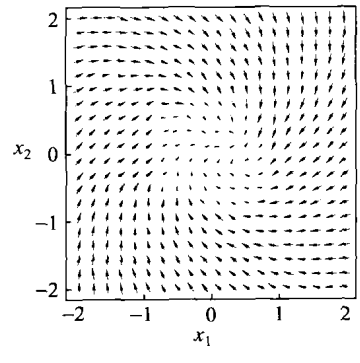


图 8.6 一个旋涡向量场 $f(x,y)=[(y-x)(-x-y)]$

1 “场”这个术语出现在两个不相关的地方：在向量空间的定义中和向量场中。记住这是一个意外的冲突。

个平滑函数。两个拓扑空间称为微分同胚的，如果在它们之间存在一个微分同胚。

向量场作为速度场　我们现在对向量场进行特殊说明。用式（8.11）表示的一个向量场可以用来定义一阶微分方程的一个集合：

$$\begin{aligned}\frac{\mathrm{d}x_1}{\mathrm{d}t} &= f_1(x_1,\cdots,x_n)\\ \frac{\mathrm{d}x_2}{\mathrm{d}t} &= f_2(x_1,\cdots,x_n)\\ &\vdots\\ \frac{\mathrm{d}x_n}{\mathrm{d}t} &= f_n(x_1,\cdots,x_n)\end{aligned} \tag{8.11}$$

每个方程表示一个坐标对时间的导数。对于任意点 $x\in\mathbb{R}^n$，一个速度向量定义为：

$$\frac{\mathrm{d}x}{\mathrm{d}t}=\begin{bmatrix}\frac{\mathrm{d}x_1}{\mathrm{d}t} & \frac{\mathrm{d}x_2}{\mathrm{d}t} & \cdots & \frac{\mathrm{d}x_n}{\mathrm{d}t}\end{bmatrix} \tag{8.12}$$

这使得向量场 f 可以作为一个速度场来说明。

通常采用一个简化符号 $\dot{x}=\mathrm{d}x/\mathrm{d}t$ 来表示。每个速度分量可以表示为 $\dot{x}_i=\mathrm{d}x_i/\mathrm{d}t$。用 f 来表示函数 $f_1,\cdots,f_n$ 的向量，式（8.11）可以表示为：

$$\dot{x}=f(x) \tag{8.13}$$

这里用 f 是为了与状态转移方程中的 f 统一起来。在第Ⅳ部分，我们将用行动来对向量场进行参数化。这产生的一个连续时间状态转移方程看起来像 $\dot{x}=f(x,u)$，并且与第 2 章中的离散转移方程非常相似。

式（8.11）所表示的微分方程通常称为是自治的或者静态的，因为 f 不依赖于时间。也可以定义一个时变向量场 $\dot{x}=f(x(t),t)$。但是在本章并不讨论这种情况。

积分曲线　如果给定一个向量场 f，那么用式（8.10）可以定义每个点处的速度向量。假设一个点在 $t=0$ 时刻从某点 $x_0\in\mathbb{R}^n$ 开始按照 f 表示的速度进行运动。那么它将运动到哪里？它的从 x_0 开始的轨迹可以表示为一个函数 $\tau:[0,\infty)\to\mathbb{R}^n$，其定义域是时间区间 $[0,\infty)$。轨迹表示初始条件为 $\tau(0)=x_0$ 的微分方程的积分曲线（或者解轨迹），如果对于每个时刻 $t\in[0,\infty)$，有：

$$\frac{\mathrm{d}\tau}{\mathrm{d}t}(t)=f(\tau(t)) \tag{8.14}$$

有时以积分的形式进行表示：

$$\tau(t)=x_0+\int_0^t f(\tau(s))\mathrm{d}s \tag{8.15}$$

这在 Caratheodory 意义下被称为微分方程的解。从直觉上讲，积分曲线从 x_0 开始，沿着速度向量所指示的方向流动。这类似于图 8.2（b）中离散情况下跟随箭头方向，只不过这里是连续空间。

例 8.9（常数速度场的积分曲线）　最简单的情况是一个定常速度场。假设在 $\mathbb{R}^2$ 上定义一个一个常数场 $x_1=1$ 和 $x_2=1$。从 $(0,0)$ 开始的积分曲线是 $\tau(t)=(t,2t)$。可以看见对于所有 $t\geqslant 0$，式（8.14）都成立。■

例 8.10（线性速度场的积分曲线）　考虑 $\mathbb{R}^2$ 上的速度场。令 $\dot{x}_1=-2x_1$ 和 $\dot{x}_2=-x_2$。函数

$\tau(t)=(e^{-2t},e^{-t})$ 表示从(1,1)开始的积分曲线。在 $t=0$，$\tau(0)=(1,1)$，这是初始状态。可以验证对于所有 $t\geqslant 0$，式（8.14）都成立。这是线性速度场的一个简单的例子。一般情况下，如果 f_i 是坐标变量 $x_1,\cdots,x_n$ 的一个线性函数，那么可以得到一个线性速度场。当速度场表示为 $\dot{x}=Ax$，那么通过确定矩阵 A 的特征值就可以得到积分曲线。更多例子可参见文献[194]。 ■

微分方程的一个基本结论是，如果 f 是平滑的，那么 $\dot{x}=f(x)$ 存在唯一一条积分曲线。也可以表述为如果 f 满足 Lipschitz 条件，那么存在唯一解。这意味着存在某常数 $c\in(0,\infty)$ 使得对于所有 $x,x'\in X$，有：

$$\|f(x)-f(x')\|\leqslant c\|x-x'\| \tag{8.16}$$

其中 $\|\cdot\|$ 表示欧氏范数。常数 c 通常被称为 Lipschitz 常数。注意到如果 f 满足 Lipschitz 条件，那么它是连续的。如果在所有 X 上 f 的所有偏导数都以一个常数为界，那么 f 是 Lipschitz 的。但是我们更愿意采用式（8.16），因为它更一般（它甚至不意味着 f 是处处可微的）。

分段平滑的向量场 可以允许向量场只在有限个小区域上是平滑的。在两个小区域之间切换边界上会发生不连续跳跃。例如，假设一个 $(n-1)$ 维的切换边界 $S\subset\mathbb{R}^n$ 被定义为：

$$S=\{x\in\mathbb{R}^n\mid s(x)=0\} \tag{8.17}$$

其中 s 是一个函数 $s:\mathbb{R}^n\to\mathbb{R}$。如果 $\mathbb{R}^n$ 的维数为 n 且 s 非奇，那么 S 的维数为 $n-1$。定义：

$$S_+=\{x\in\mathbb{R}^n\mid s(x)>0\} \tag{8.18}$$

$$S_-=\{x\in\mathbb{R}^n\mid s(x)<0\} \tag{8.19}$$

这些定义类似于第 3.1.2 节中用几何本原构造隐式模型。假设 $f(x)$ 在 S_+ 和 S_- 上是平滑的，但是在 S 有一个不连续跳跃。这种微分方程用于描述控制论中的混合系统[139,411,637]。那里的任务是设计一个混合控制系统。这种情况下我们是否还能够确定一条解轨迹？在特殊情况下，我们可以得到 Filipov 意义下微分方程的一个解[340,848]。

令 $B(x,\delta)$ 表示半径为 δ、中心在 x 的一个开球。令 $f(B(x,\delta))$ 表示集合：

$$f(B(x,\delta))=\{x'\in X\mid\exists x''\in B(x,\delta),x'=f(x'')\} \tag{8.20}$$

令 X_0 表示 $\mathbb{R}^n$ 的任何零度量子集（例如：$\mu(X_0)=0$）。令 hull(A) 表示 $\mathbb{R}^n$ 中一个点集 A 的凸包。一条路径 $\tau:[0,t_f]\to\mathbb{R}^n$ 被称为 Filipov 意义下的一个解，如果对于所有 $t\in[0,t_f]$：

$$\frac{\mathrm{d}\tau}{\mathrm{d}t}(t)\in\bigcap_{\delta>0}\left\{\bigcap_{X_0}\mathrm{hull}(f(B(\tau(t),\delta)\setminus X_0))\right\} \tag{8.21}$$

其中交集是在所有可能的 $\delta>0$ 和零测度集 X_0 上进行的。式（8.21）被称为一个微分包含(differential inclusion)[54]，因为对 $\dot{x}$ 来说，可能存在一个选择集。“几乎全部”这种要求意味着在 $[0,t_f]$ 中的一个零测度集上条件有可能不成立。从直觉上讲，几乎所有由 τ 产生的速度向量必须指向 f 在 $\tau(x(t))$ 附近所给出的速度向量“之间”。“之间”来自于对凸包的使用。Filipov 意义下的解是对 Caratheodory 意义下解的一般化。在这种情况下，由 τ 产生的每个速度向量必须与式（8.14）给出的 $f(x(t))$ 相符。式（8.21）中的条件允许在解中出现各种各样的越界，甚至允许 f 是不连续的。

一些奇异的向量场可以产生 Filipov 意义下的解。在 Filipov 条件允许的情况下，切换

边界模型是相对比较简单的。图 8.7 显示了在切换边界 S 上可能发生的各种情况。对于**一致流动**（consistent flow）的情况，解会像你所直觉预料的那样发生。Filipov 条件式（8.21）要求在 S 处 τ 的速度向量指向穿过 S 前后的向量之间（例如它可以指向下，这是两个方向的平均）。数量必须在两个数量之间。对于**向内流动**（inward flow）的情况，积分曲线沿着 S 运动，S 内的向量与边界两侧的向量指向相同方向（在凸包内）。在涉及实际系统的应用中，这可能导致围绕 S 振荡。通过正则化加厚边界[848]可以减小这种振荡（滑动控制用于讨论这种问题[305]）。如果初始状态在 S 中，**向外流动**（outward flow）情况可能导致不唯一性。但是从 S 外开始的轨迹不会经过 S，不存在这种问题。如果流动与边界的两侧相切，那么可能发生其他形式的不唯一性。本章将避免**切向流动**（tangent flow）。

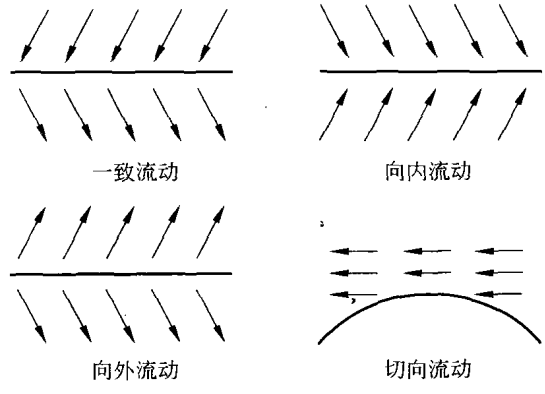


图 8.7　在切换边界周围不同类型的流动

8.3.2　平滑流形

第 4.1.2 节给出的流形的定义通常被称为拓扑流形。用这种方法定义的流形不一定有足够的公理来保证诸如微分和积分等微积分运算的进行。我们愿意讨论位形空间 C 上或者连续状态空间 X 上的速度。由第 4 章我们知道，位形空间可以是一个诸如 $\mathbb{RP}^3$ 的流形。因此我们需要进一步定义流形应该具有的一些性质来保证微积分的进行。这导致了平滑流形的概念。

假设 M 是一个拓扑流形，如第 4.1.2 节中定义的那样。例如 M 可以代表 $\mathbb{R}^3$ 的所有旋转矩阵的集合 $SO(3)$。一个非常简单的例子是 $M=\mathbb{S}^2$，它是 $\mathbb{R}^3$ 中的一个球。我们将第 8.3.1 节中的概念从 $\mathbb{R}^n$ 扩展到流形。第一个定义是点 $p\in M$ 处的切空间 $T_p(M)$。就像你直观想象的那样，切向量与一个曲面相切，如图 8.8 所示。它们指示沿着 p 的流形可能的运动速度。在流形上比 $\mathbb{R}^n$ 上更加难以定义，因为很容易用 n 个坐标表达 $\mathbb{R}^n$ 中的任意点，并且每个点

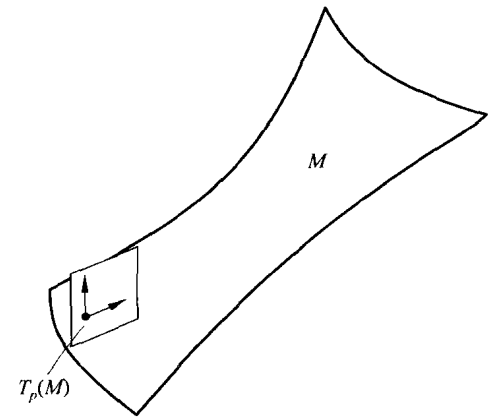


图 8.8　从直观上讲，切空间是在 p 的一个邻域内对流形的一个线性近似

$p \in \mathbb{R}^n$ 处切空间的所有局部坐标系是相互一致的。对于诸如 $\mathbb{S}^2$ 的一个流形，我们定义切空间所采用的方式必须对坐标不敏感，且能够处理当我们在 $\mathbb{S}^2$ 周围运动时切平面发生旋转这种事实。

首先仔细考虑给一个流形赋予坐标意味着什么。假设 M 的维数为 n，它嵌入在 $\mathbb{R}^m$ 中。对于 $M = SO(3)$，有 $n=3$ 和 $m=9$。对于 $M=\mathbb{S}^2$，有 $n=2$ 和 $m=3$。坐标的数量应该是 M 的维数 n；但是嵌入在 $\mathbb{R}^m$ 中的流形通常表示为 $\mathbb{R}^m$ 的一个子集，它必须遵守一些等式约束。我们愿意用 $\mathbb{R}^n$ 中的坐标来表达 M 的某部分。

坐标与参数化 对于任何开集合 $U \subseteq M$ 以及函数 $\phi: U \to \mathbb{R}^n$，使得 ϕ 是 $\mathbb{R}^n$ 的一个子集的一个同胚，有序对 (U,ϕ) 被称为坐标邻域（有些文献称之为图）。某 $p \in U$ 的值 $\phi(p)$ 被称为 p 的坐标。

例 8.11（$\mathbb{S}^1$ 上的坐标邻域） 对于圆 $M=\mathbb{S}^1$，可以得到一个简单的例子。假设 M 表示成一个嵌入 $\mathbb{R}^2$ 的单位圆（$x^2+y^2=1$ 的解集）。令 (x,y) 表示 $\mathbb{R}^2$ 的一个点。令 U 表示 $\mathbb{S}^1$ 的 $x>0$ 的子集。一个坐标函数 $\phi: U \to (-\pi/2, \pi/2)$ 可以定义为 $\phi(x,y) = \tan^{-1}(y/x)$。

对于某坐标邻域 (U,ϕ)，令 $W = \phi(U)$（ϕ 的值域）。因为通过 ϕ，U 和 W 是同胚的，所以也可以定义逆函数 ϕ^{-1}。求逆是我们熟知的参数化思路。继续本例，ϕ^{-1} 产生映射 $\theta \mapsto (\cos\theta, \sin\theta)$，它是圆的常用的参数表示，但是限制在 $\theta \in (-\pi/2, \pi/2)$。 ■

为了进行点 $p \in M$ 处的微分，在 M 的包含 p 的开子集上定义坐标邻域是非常重要的。这主要是因为定义一个函数在一个点处的导数要求在该点的周围存在一个开集。如果坐标没有边界，那么这将是可能的。遗憾的是，用一个坐标邻域不可能覆盖 M 的全部，除非 $M=\mathbb{R}^n$（或者至少 M 与 $\mathbb{R}^n$ 同胚）。因此我们必须定义多个邻域，它们的定义域覆盖全部 M。因为每个定义域都是一个开集，所以有些定义域必定相互重叠。在这种情况下会发生什么？对同一个点我们可能有两个或者两个以上的可选坐标。从一个坐标集运动到另外一个坐标集就是微积分中常见的坐标变换。现在对此进行描述。

假设 (ϕ,U) 和 (ψ,V) 是同一个流形 M 的坐标邻域，$U \cap V \neq \phi$。图 8.9 指示从 ϕ 到 ψ 如何进行坐标变换。坐标变换用复合函数表示为 $\psi \circ \phi^{-1}: \mathbb{R}^n \to \mathbb{R}^n$（$\phi^{-1}$ 将 $\mathbb{R}^n$ 映射到 $M\psi$ 将 M 一个子集映射到 $\mathbb{R}^n$）。

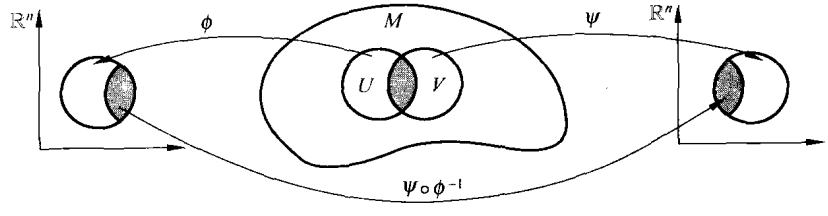


图 8.9 对坐标的变换的一种描述

例 8.12（坐标变换） 考虑 $M = SO(3)$ 从欧拉角变换到四元数。因为 $SO(3)$ 是一个三维流形，所以 $n=3$。这意味着任何坐标邻域必须将 $SO(3)$ 中的一个点映射到 $\mathbb{R}^3$ 中的一个点。通过一个给定的旋转矩阵来计算欧拉角，那么可以构造一个坐标函数 $\phi: SO(3) \to \mathbb{R}^3$。该函数实际上在式（3.47）、式（3.48）、式（3.49）中已经进行了定义。为了产生一个坐标邻域，必须指定 M 一个开子集 U。

用四元数可以构造另外一个坐标函数 $\psi : SO(3) \to \mathbb{R}^3$。这看起来存在一个问题，因为四元数有 4 个分量；但是第四个分量可以由其他三个来确定。采用式（4.24）～式（4.26），可以确定 a、b、c 坐标。

现在假设我们想在重叠区域 $U \cap V$ 中将欧拉角变换到四元数，其中 V 是一个开集，四元数的坐标邻域就定义在该开集上。任务是构造一个坐标变换 $\psi \circ \phi^{-1}$。我们首先必须在 U 转换 ϕ。即我们需要 M 一个欧拉角参数表示。这由式（3.42）给出，它为 α、β、γ 产生一个旋转矩阵 $\phi^{-1}(\alpha,\beta,\gamma) \in SO(3)$。一旦确定该矩阵，那么用 ψ 就可以确定四元数参数 a、b、c。这意味着我们已经构造了三个函数 f_1、f_2、f_3，它们产生 $a = f_1(\alpha,\beta,\gamma)$、$b = f_2(\alpha,\beta,\gamma)$、$c = f_3(\alpha,\beta,\gamma)$。这样就产生了 $\psi \circ \phi^{-1}$。 ■

在不同的环境中进行坐标变换有几个原因。例 8.12 是由运动规划中经常出现的变换而引起的。例如，假设图形组件用四元数显示物体，但是碰撞检测算法采用欧拉角。在这种情况下，就有必要频繁进行坐标变换。在微积分的学习中，你可能回忆到用坐标变换来简化积分。在平滑流形的定义中，还有另外一个意图。因为坐标邻域是基于开集的同胚，所以可能需要几个坐标邻域来覆盖全部 M。例如，即使我们决定 $SO(3)$ 采用四元数，那么还可能需要几个映射到四元数的坐标邻域。在它们定义域的交集上，需要进行坐标变换。

现在我们定义一个平滑流形。坐标变换将出现在流形的定义中，它们必须满足平滑条件。一个（拓扑）流形 M 上的一个平滑结构[1]是坐标邻域的一个族[2] $\mathcal{U} = \{U_\alpha, \phi_\alpha\}$ 以使：

（1）所有 U_α 的并集包含 M。因此有可能为 M 中的任意一个点得到 $\mathbb{R}^n$ 中的坐标。

（2）对于 $\mathcal{U}$ 中任何 (U,ϕ) 和 (V,ϕ)，如果 $U \cap V \neq \phi$，那么坐标变换 $\psi \circ \phi^{-1}$ 和 $\phi \circ \psi^{-1}$ 是 $U \cap V$ 上的平滑函数。在交集上，坐标变换必须产生微分同胚。在这种情况下，坐标邻域称为是相容的。

（3）如果某 (U,ϕ) 与 u 中的每个坐标邻域都是相容的，那么 $\mathcal{U}$ 必须包括 (U,ϕ)，在这个意义下族 $\mathcal{U}$ 是最大的。

一个著名的定理（见文献[135]第 54 页）论述到：如果相容的邻域的一个集合覆盖全部 M，那么存在唯一的平滑结构包含它们[3]。这意味着一个微分结构通常可以用少量的邻域来指定，其余的邻域是隐含的。

如第 4.1.2 节所定义的，一个有平滑结构的流形称为平滑流形[4]。

例 8.13（$\mathbb{R}^n$ 作为一个平滑流形） 我们期望目前的概念能够应用于 $\mathbb{R}^n$，$\mathbb{R}^n$ 是最简单的流形族。可以用一个坐标邻域 $\mathbb{R}^n \to \mathbb{R}^n$，它是恒等映射。对于所有整数 $n \in \{1,2,3\}$ 和 $n > 4$，这是 $\mathbb{R}^n$ 上可能仅有的平滑结构。令人惊奇的是对于 $\mathbb{R}^4$，有无数多个不相容的平滑结构，它们称为奇异 $\mathbb{R}^4$ [293]。但是不必担心，因为只使用恒等映射给出的其中一个。 ■

例 8.14（$\mathbb{S}^n$ 作为一个平滑流形） 将 $\mathbb{S}^n$ 定义成一个平滑流形的一个方法是使用 $2(n+1)$ 个坐标邻域，这产生简单的表达。令 $\mathbb{S}^n$ 定义为：

$$\mathbb{S}^n = \left\{ (x_1, \cdots, x_{n+1}) \in \mathbb{R}^{n+1} \mid x_1^2 + \cdots + x_{n+1}^2 = 1 \right\} \tag{8.22}$$

1　另外的一个名称是微分结构和 C^∞ 结构。

2　在坐标领域被称为图（charts）的文献中，这个族被称为一个图册（atlas）。

3　这是基于假设：M 是 Hausdorff 的且有可数个开集的基。

4　另外的一个名称是微分流形和 C^∞ 流形。

每个坐标邻域的定义域定义如下。对于每个 i 从 1 到 $n+1$，有两个邻域：

$$U_i^+ = \{(x_1,\cdots,x_{n+1}) \in \mathbb{R}^{n+1} \mid x_i > 0\} \tag{8.23}$$

$$U_i^- = \{(x_1,\cdots,x_{n+1}) \in \mathbb{R}^{n+1} \mid x_i < 0\} \tag{8.24}$$

每个邻域是一个开集，它覆盖一半 $\mathbb{S}^n$，但没有 $x_i = 0$ 处的大圆。通过投影到包含大圆的 $(n-1)$ 维超平面就可以定义坐标函数。对于每个 i，在 U_i^+ 上：

$$\phi_i^+(x_1,\cdots,x_{n+1}) = (x_1,\cdots,x_{i-1},x_{i+1},\cdots,x_n) \tag{8.25}$$

用同样的方法在 U_i^- 上定义每个 ϕ_i^-。如所要求的那样，每个坐标函数是从 $\mathbb{S}^n$ 的一个开子集到 $\mathbb{R}^n$ 的一个开子集的一个同胚。在邻域重叠的子集上，坐标函数的变换上平滑的。例如对于某 $i \neq j$，考虑从 ϕ_i^+ 到 ϕ_j^- 的变换。坐标变换是一个函数 $\phi_j^- \circ (\phi_i^+)^{-1}$。$\phi_i^+$ 的逆表示为：

$$\begin{aligned}&(\phi_i^+)^{-1}(x_1,\cdots,x_{i-1},x_{i+1},\cdots,x_n) = \\ &(x_1,\cdots,x_{i-1},1-\sqrt{1-x_1^2-\cdots-x_{i-1}^2-x_{i+1}^2-\cdots-x_n^2},x_{i+1},\cdots,x_n)\end{aligned} \tag{8.26}$$

当与 ϕ_j^- 组合，那么第 j 个坐标被去掉。这产生：

$$\begin{aligned}&\phi_k^- \circ (\phi_i^+)^{-1}(x_1,\cdots,x_{i-1},x_{i+1},\cdots,x_n) = \\ &(x_1,\cdots,x_{i-1},1-\sqrt{1-x_1^2-\cdots-x_{i-1}^2-x_{i+1}^2-\cdots-x_n^2},x_{i+1},\cdots,x_{j-1},x_{j+1},\cdots,x_n)\end{aligned} \tag{8.27}$$

它是定义域 U_i^+ 上的一个平滑函数。圆 $\mathbb{S}^1$ 和球 $\mathbb{S}^2$ 的坐标变换可以形象描述。

采用球极投影（stereographic projection），平滑结构也可以只用两个坐标邻域来定义。对于 $\mathbb{S}^2$，引一条从北极到 x 的线，然后将它映射到与它相交的 $x_3 = 0$ 平面中的点，这样一个坐标函数就可以将几乎每个点 $x \in \mathbb{S}^2$ 映射到 $\mathbb{R}^2$。只排除北极这个点。从南极也可以构造一个相似的映射。■

例 8.15（$\mathbb{RP}^n$ **作为一个平滑流形**） 这个例子特别重要，因为 $\mathbb{RP}^3$ 与第 4.2.2 节建立的 $SO(3)$ 是同一个流形。第 4.1.2 节中 $\mathbb{RP}^n$ 被定义为 $\mathbb{R}^{n+1}$ 中所有经过原点的直线的集合。这意味着对于任何 $\alpha \neq 0$ 的 $\alpha \in \mathbb{R}$ 和任何 $x \in \mathbb{R}^{n+1}$，x 和 αx 是相同的。在射影空间中，尺度的大小无关紧要。

一个平滑结构可以仅用 $n+1$ 个坐标邻域来指定。对于每个 i 从 1 到 $n+1$，令：

$$\phi_i(x_1,\cdots,x_{n+1}) = (x_1/x_i,\cdots,x_{i-1}/x_i,x_{i+1}/x_i,\cdots,x_n/x_i) \tag{8.28}$$

在 $\mathbb{R}^{n+1}$ 中所有点的开集上，$x_i \neq 0$。坐标函数的逆函数由下式给出：

$$\phi_i^{-1}(z_1,\cdots,z_n) = (z_1,\cdots,z_{i-1},1,z_i,\cdots,z_{n+1}) \tag{8.29}$$

不难验证这些简单的变换在重叠的邻域上是平滑的。

$SO(3)$ 上一个平滑结构可以作为一个特殊情况而产生，因为 $SO(3)$ 拓扑等价于 $\mathbb{RP}^3$。假设 $SO(3)$ 的元素用单位四元数表示。每个 (a,b,c,d) 被认为是 $\mathbb{S}^3$ 的一个点。有四个坐标邻域。例如其中之一是：

$$\phi_b(a,b,c,d) = (a/b,c/b,d/b) \tag{8.30}$$

它定义在 $\mathbb{R}^4$ 的 $b \neq 0$ 的子集上。需要定义 $\phi_b(a,b,c,d)$ 的逆以使 $SO(3)$ 上的点映射到 $\mathbb{R}^4$ 中一个有单位数量的点。■

流形上的切空间 现在考虑在流形上定义切空间。从直观上讲，n 维流形 M 上点 p 处的切空间 $T_p(M)$ 是 $\mathbb{R}^m$ 中的一个 n 维超平面，当超平面的原点平移到 p 点时，它在 p 周围

非常好地逼近 M，如图 8.8 所示。第 7.4.1 节通常采用切面来描述闭运动链的运动规划的局部运动（见图 7.22）。

为了定义流形的切空间，我们首先考虑 $\mathbb{R}^n$ 中一个点处的切空间，它比第 8.3.1 节给出的定义更加复杂。假设 $M=\mathbb{R}^2$ 以及考虑一个平滑函数 $f:\mathbb{R}^2\to\mathbb{R}$ 在点 $p\in\mathbb{R}^2$ 处的方向导数。对某（非归一化的）方向向量 $v\in\mathbb{R}^2$，f 在 p 点的方向导数被定义为：

$$\nabla_v(f)\big|_p=v_1\frac{\partial f}{\partial x_1}\Big|_p+v_2\frac{\partial f}{\partial x_2}\Big|_p \tag{8.31}$$

这里的方向导数没有归一化方向向量（这与基本微积分不同）。因此 $\nabla_v(f)=\nabla(f)\cdot v$，其中“$\cdot$”表示内积或者点积，$\nabla(f)$ 表示 f 的梯度。在该构造中使用的所有可能的方向向量的集合形成的一个二维向量空间，它恰好是以前所定义的切空间 $T_p(\mathbb{R}^2)$。这可以推广到 n 维情况：

$$\nabla_v(f)\big|_p=\sum_{i=1}^{n}v_i\frac{\partial f}{\partial x_i}\Big|_p \tag{8.32}$$

所有可能的方向向量表示切空间 $T_p(\mathbb{R}^n)$。为运动规划之目的，所有方向的集合可以阐述为可能的速度向量的集合。

现在考虑取一个流形上平滑函数 $f:M\to\mathbb{R}$ 的（非归一化的）方向导数。对一个 n 维流形，点 $p\in M$ 处的切空间 $T_p(M)$ 再次被认为是所有非归一化的方向的集合。这些方向必须与流形相切，如图 8.8 所示。那么就存在一个简便方法来定义它们，而不需要特定的坐标邻域。这样的关于 $T_p(M)$ 的定义是流形的固有性质的反映。

在这一点上，你可能已经接受这样一个事实：$T_p(M)$ 是附黏于 M 中 p 点的一个 n 维向量空间，其指向如图 8.8 所示。但是为了完整起见，将给出来自于微分几何的 $T_p(M)$ 的定义；详细内容见文献[135,874]。构造过程是基于刻画所有可能方向导数算子的集合的特征。令 $C^\infty(p)$ 表示定义域包含 p 的所有平滑函数的集合。现在进行如下等同。任意两个函数 $f,g\notin C^\infty(p)$ 定义为等价的，如果存在一个开集 $U\subset M$ 使得对于任何 $p\in U$，$f(p)=g(p)$。不需要区分等价函数，因为它们在 p 点处的导数必须相同。令 $\tilde{C}^\infty(p)$ 表示这种等同下的 C^∞。p 点上的一个方向导数算子可以认为是在某方向将 $\tilde{C}^\infty(p)$ 映射到 $\mathbb{R}$ 的一个函数。在 $\mathbb{R}^n$ 情况下，对每个方向 v，算子为 ∇_v。考虑所有方向导数算子的集合。必须为 $\tilde{C}^\infty(p)$ 中的每个函数指派一个实值，它必须遵守关于方向导数的微积分的两个公理。令 ∇_v 表示某 $p\in M$ 处一个方向导数算子（需要小心，因为这里的 v 没有显式表示，原因是它没有坐标）。方向导数算子必须满足两个公理：

1. 线性：对于任意 $\alpha,\beta\in\mathbb{R}$ 和 $f,g\in\tilde{C}^\infty(p)$

$$\nabla_v(\alpha f+\beta g)=\alpha\nabla_v f+\beta\nabla_v g \tag{8.33}$$

2. 求导：对于任意 $f,g\in\tilde{C}^\infty(p)$

$$\nabla_v(fg)=\nabla_v fg(p)+f(p)\nabla_v g \tag{8.34}$$

你可以将这些来自于基本向量微积分的公理作为方向导数的性质。可以看到所有可能的满足这些公理的算子形成了一个 n 维向量空间[135]。这个向量空间称为 p 点处的切空间 $T_p(M)$。这样就完成了对切空间的定义，而没有用到坐标。

但是有一个明确的表达 $T_p(M)$ 中向量的方法是非常重要的。通过使用坐标邻域，可以得到切空间的一个基。微分几何中一个重要的定理论述到：如果 $F:M\to N$ 是一个开集 $U\subset N$ 的一个微分同胚，那么切空间 $T_p(M)$ 与 $T_{F(p)}(N)$ 是同构的。这意味着通过使用一个参数表示（一个坐标邻域的逆），在 $T_p(M)$ 中的速度向量和 $T_{F(p)}(N)$ 中的速度向量之间存在一个双射。小的参数扰动引起在流形 N 上切向的运动。例如假设给表示 $SO(3)$ 的四元数中的 3 个参数加入一个小的扰动。如果扰动足够小，那么所发生的运动与 $SO(3)$ 相切。那就是说，受扰矩阵离 $SO(3)$ 非常近（它们没有在 $SO(3)$ 中，因为 $SO(3)$ 用 $\mathbb{R}^9$ 上的非线性约束来定义，如第 4.1.2 节所讨论）。

例 8.16（$\mathbb{S}^2$ **的切空间**） 确定 $\mathbb{S}^2$ 的切空间可以使我们的讨论更加具体，该空间作为满足 $x^2+y^2+z^2=1$ 的所有点 $(x,y,z)\in\mathbb{R}^3$ 的集合而嵌入 $\mathbb{R}^3$。通过使用标准球坐标，可以定义一个覆盖 $\mathbb{S}^2$ 的绝大部分的坐标邻域。令 f 表示将 (x,y,z) 映射到角 (θ,ϕ) 的坐标函数。f 的定义域是由 $\theta\in(0,2\pi)$ 和 $\phi\in(0,\pi)$ 所定义的开集（排除极点）。标准公式是 $\theta=\text{atan2}(y,x)$ 和 $\phi=\cos^{-1}z$。逆函数 f^{-1} 产生 $x=\cos\theta\sin\phi$、$y=\sin\theta\sin\phi$ 以及 $z=\cos\phi$。

现在考虑表达极点以外的某点 $p\in\mathbb{S}^2$ 处切空间的不同方法（需要坐标的变换来处理这些）。使用坐标 (θ,ϕ)，速度可以定义为 $\mathbb{R}^2$ 中的向量。我们可以假设在由 θ 和 ϕ 定义的平面内的运动，只要遵守 $\theta\in(0,2\pi)$ 和 $\phi\in(0,\pi)$。

我们也可以采用参数表示来产生切空间的基向量，它们是 $\mathbb{R}^3$ 中的向量。因为切空间仅有两维，我们必须得到一个在 p 点与球“相切”的平面。通过求导可以得到它们。令 f^{-1} 表示为 $x(\theta,\phi)$、$y(\theta,\phi)$、$z(\theta,\phi)$。p 点处切平面的两个基向量是：

$$\left[\frac{\mathrm{d}x(\theta,\phi)}{\mathrm{d}\theta}\quad\frac{\mathrm{d}y(\theta,\phi)}{\mathrm{d}\theta}\quad\frac{\mathrm{d}z(\theta,\phi)}{\mathrm{d}\theta}\right] \tag{8.35}$$

$$\left[\frac{\mathrm{d}x(\theta,\phi)}{\mathrm{d}\phi}\quad\frac{\mathrm{d}y(\theta,\phi)}{\mathrm{d}\phi}\quad\frac{\mathrm{d}z(\theta,\phi)}{\mathrm{d}\phi}\right] \tag{8.36}$$

计算并归一化这些导数就产生向量 $[-\sin\theta\ \cos\theta\ 0]$ 和 $[\cos\theta\cos\phi\ \ \sin\theta\cos\phi\ \ -\sin\phi]$。它们可以被假设为在 p 点对 θ 和 ϕ 的小的扰动所产生的结果。这些向量的线性组合所得到的向量空间是 $\mathbb{R}^2$ 的切空间。正如所料，基向量的方向依赖于 $p\in\mathbb{S}^2$。

切向量处在与曲面相切的平面内，如图 8.8 所示。由 $g(x,y,z)=0$ 所指定的一个曲面的法向量是 ∇g，在归一化后它产生 $[x,y,z]$。也可以通过上述两个向量的叉积，并用 f^{-1} 的参数中的 x、y、z 来表达它。对一个点 $p=(x_0,y_0,z_0)$，平面方程是：

$$x_0(x-x_0)+y_0(y-y_0)+z_0(z-z_0)=0 \tag{8.37}$$

■

流形上的向量场和速度场 流形上的切空间概念看起来和 $\mathbb{R}^n$ 上的相同。这使向量场的定义和概念可以很自然地从 $\mathbb{R}^n$ 扩展到平滑流形。流形 M 上的向量场为每个 $p\in M$ 分配了 $T_p(M)$ 中的一个向量。可以将它想象成一个指针图，但是现在指针图散布在流形上，而不是在 $\mathbb{R}^n$ 中。

一个向量场的速度场描述也可以扩展到平滑流形。这意味着现在 $\dot{x}=f(x)$ 在 M 上定义了 n 个微分方程的一个集合，并且通常用平滑结构的一个坐标邻域来表达。如果 f 是一个

平滑向量场，那么从任意 $x_0 \in M$ 都可以定义解轨迹 $\tau:[0,\infty] \to M$。对分段平滑向量场的情况，也可以定义 Filipov 意义下的解轨迹。

8.4　连续空间中的完备方法

如果解存在，一个完备的反馈规划算法必须计算一个反馈解；否则，它必须报告无解。第 8.4.1 节与第 8.2 节是并行的，它定义了连续状态空间情况下的反馈规划和导航函数。第 8.4.2 节说明从采用胞腔剖分技术计算而来的一个胞腔复形如何来定义一个可行反馈规划。第 8.4.3 节提出一种组合方法来计算最优导航函数以及相应的 $\mathbb{R}^2$ 中的反馈规划。第 8.4.2 节和第 8.4.3 节允许反馈规划是不连续的向量场。在一些应用中，特别是在动力学占主导的应用中，需要加强关于导航函数以及它们所产生的向量场的某些条件。第 8.4.4 节因此考虑了关于对所允许的向量场和导航函数的约束。这些包括 Rimon-Koditschek 意义下的导航函数（见文献[832]），导航函数这个术语就是该文献提出的。

8.4.1　反馈运动规划

使用第 8.3 节的概念，我们现在准备定义位形空间或者其他连续状态空间上的反馈运动规划。回顾表述 4.1，它根据位形空间定义了基本运动规划问题。在这里的不同之处是没有初始条件和对解路径的要求被解向量场所代替。这里的表述可以认为是在连续时间下对表述 8.1 的适应性修改。

表述 8.2（反馈运动规划）

1. 有一个状态空间 X，它是一个平滑流形。状态空间通常是 C_{free}，如第 4.3.1 节所定义的[1]。

2. 对每个状态 $x \in X$，有一个行动空间 $U(x) = T_x(X)$。指定零速度 $0 \in T_x(X)$ 为终止行动 u_T。使用这种模型，机器人在任何状态下都能够选择它的速度[2]。

3. 一个无界的时间区间 $T = [0,\infty)$。

4. 一个状态转移（微分）方程：

$$\dot{x} = u \tag{8.38}$$

它用坐标邻域来表达，产生直接由行动 u 确定的速度 $\dot{x}$。u_T 产生的速度是 $0 \in T_x(X)$（意味着“停止”）。

5. 一个目标集合 $X_G \in X$。

表述 8.2 中的反馈规划 π 被定义为一个函数 π，它为每个 $x \in X$ 产生一个行动 $u \in U(x)$。反馈规划可以等价地认为是 X 上的一个向量场，因为每个 $u \in U(x)$ 指定一个速度向量（u_T 指定零速度）。因为初始条件没有确定，那么定义作为问题解的规划的含义就更加复杂一些。令 $X_r \subset X$ 表示所有使得 X_G 是可达的那些状态的集合。更精确地，一个状态 x_I 属于 X_r，

1　注意到 X 已经排除掉障碍区域。对于第Ⅳ部分中的某些问题，状态空间将是 $X = C$，这包括障碍区域。

2　这允许速度的不连续变化，在许多应用中这是不切实际的。也将会讨论诸如加速度界等这些额外的约束。对于微分约束的完整考虑，参见第Ⅳ部分。

当且仅当存在一条连续路径 $\tau:[0,1]\rightarrow X$，对于该路径，存在某 $x_G \in X_G$ 使得 $\tau(0)=x_I$ 和 $\tau(1)=x_G$。这意味着对于第 4 章考虑的“开环”运动规划问题存在从 x_I 开始的解路径。

8.4.1.1 解的概念

一个反馈规划 π 称为表述 8.2 中的问题的一个解，如果从所有 $x_I \in X_I$，π 的积分曲线（认为是一个向量场）都到达 X_G，并在所到达的点处采取终止行动。必须给出一些关于“到达” X_G 的含义的词语。控制论[526,848]中的稳定性的概念对区分不同情况非常有用，参见第 15.1 节。如果 X_G 是一个以 x_G 为中心的小球，那么使用如图 8.5（b）所示的向内向量场在有限时间以后到达这个球。现在假设 X_G 是单点 x_G。向内向量场产生的速度引导状态越来越接近原点，但是什么时候到达 x_G 呢？在这种情况下收敛于原点仅仅是渐近的；当时间趋近无穷时，极限是原点。这种稳定性问题在控制论中经常出现。我们可能允许这种渐近收敛到目标（如果向量场是平滑的，并且目标是一个点，那么这是不可避免的）。如果任何积分曲线仅渐近收敛于目标，那么解路径称为渐近解路径。一般情况下，不可能要求 π 是平滑的（或者是连续的）非零向量场。例如，由 hairy ball 定理[837]可知，对于任何偶数 n 的 $\mathbb{S}^n$，这种向量场是不存在的。因此最强的可能要求是 π 在除了零测度集以外都是平滑的，参见第 8.4.4 节。我们允许这样的解 π：它几乎所有的积分曲线都到达 X_G。

但是在本章，默认情况下都假设一个解规划在有限时间内收敛于 x_G。例如，如果归一化向内场来在除原点以外的任何地方产生单位速度，那么在有限时间内到达原点。可以对所允许的向量场施加一个约束而不影响解规划的存在性。像基本运动规划问题那样，沿着路径的速度并不重要。令一个归一化向量场是指对于所有 $x \in X$，满足 $\|f(x)\|=1$ 或者 $f(x)=0$ 的任何向量场。这意味着所有速度向量要么是单位向量，要么是零向量，并且速度不再是一个影响结果的因素。一个归一化的向量场提供运动的方向或者不运动。对于每个 $x \in X$，通过将速度向量 $f(x)$ 除以它的量值（除非量值为零），任何向量场 f 都可以转换为一个归一化向量场。

在一些情况下，单位速度并不一定就是真实物理意义下的恒定速度。例如，如果机器人是一个浮动的刚体，那么有多种方法来参数表示位置和方向。物体的速度对这些参数很敏感。因此其他约束可能比 $\|f(x)\|=1$ 更可取；但是施加的约束使得 $f(x)$ 给出 x 处的方向，记住这一点是非常重要的。特定的量值并不重要。

至此，我们仅考虑的是一个可行反馈运动规划问题。引入一个代价泛函，可以定义一个最优反馈运动规划问题。令 $\tilde{x}_t$ 表示函数 $\tilde{x}_t:[0,t]\rightarrow X$，它称为状态轨迹（或者状态历史），它是以前为离散情况所定义的状态历史的连续时间版本。相似地，令 $\tilde{u}_t$ 表示行动轨迹（或行动历史）$\tilde{u}_t:[0,t]\rightarrow U$。令 L 表示一个代价泛函，从任意 x_I，它产生：

$$L(\tilde{x}_{t_F},\tilde{u}_{t_K})=\int_0^{t_F} l(x(t),u(t))\mathrm{d}t+l_F(x(t_F)) \tag{8.39}$$

其中 t_F 是应用终止行动的时刻。用状态转移方程（8.38），$l(x(t),u(t))$ 项也可以表示为 $l(x(t),\dot{x}(t))$。一个归一化的向量场被认为是一个最优反馈运动规划，如果它能够对从能够到达目标的所有初始状态开始的式（8.39）进行最优化。

应注意到，用行动历史和初始状态可以确定状态轨迹。实际上我们已经使用了行动轨迹来定义第 4 章的运动规划问题的解路径。所不同的是，那里定义的解是路径 $\tau:[0,1]\rightarrow C_{free}$

来避免引入平滑流形上的速度场。那是本书中唯一的一个行动空间似乎消失了的地方，现在你可以看见：它仅仅是隐藏起来以避免无关紧要的符号。

8.4.1.2　导航函数

像第 8.2.2 节那样，我们假定得到连续状态空间中的一个局部算子，那么就可以用势函数来表示反馈规划。在离散情况下，局部算子选择能够减小势值的行动。在连续情况下，局部算子必须将势函数转换为一个向量场。也就是说，在每个状态下都必须定义一个速度向量。默认情况下，这里假设导航函数产生的向量场不一定归一化。

假设对所有 $x \in X_G$ 定义 $\pi(x) = u_T$，不管势函数如何。假设定义了一个势函数 $\phi : X \to \mathbb{R}$，它的梯度：

$$\nabla\phi = \left[\frac{\partial\phi}{\partial x_1} \quad \frac{\partial\phi}{\partial x_2} \quad \cdots \quad \frac{\partial\phi}{\partial x_n} \right] \tag{8.40}$$

在所有 $X \setminus X_G$ 上存在。相应的反馈规划可以定义为 $\pi(x) = -\nabla\phi|_x$。这定义了一个局部算子，它意味着速度选取的是 ϕ 的最速下降的方向。用这种方式使用势函数的思路是 Khatib[528,529]提出的，这种思路也可以认为是一种形式的梯度下降法，而梯度下降法是一种一般的优化技术。

也可能用到梯度处处不存在的势函数。在这种情况下，对一个小的固定的 Δt，式（8.4）的一个连续空间版本可以被定义为：

$$u^* = \operatorname*{arg\,min}_{u \in U(x)} \{\phi(x')\} \tag{8.41}$$

其中 x' 是从 x 在 Δt 时间内对速度 u 进行积分得到的状态。一个问题是选择的 Δt 应该使用 ϕ 的最小的可能邻域。最好只允许这样的势函数：对于它来说，在不影响式（8.41）的结果的情况下，每个 x 的 Δt 尽可能小。更精确地，这意味着 u^* 值的一个无穷序列可以由一个收敛于 0 的 Δt 值的序列来确定。一个势函数的选择应该保证序列 u^* 中的某点存在，并且当 Δt 趋近于 0 时，选择相同的 u^* 来满足式（8.41）。一个特殊情形是 ϕ 的梯度是否存在；在这种情况下，无穷序列收敛于负梯度。

一个势函数 ϕ 被称为一个导航函数，如果它产生的向量场是一个解规划。最优尚需代价用作一个最优导航函数。如果从同一个 ϕ 可以产生多个向量场，那么每个可能的向量场都必须产生一个反馈规划解。如果设计合理，势函数可以作为一种将状态引向 X_G 的“滑雪斜坡”。如果有局部极小点，那么将不能到达 X_G。导航函数不允许这种 X_G 外的局部极小点，并且可能还有其他附加的要求来保证所产生的向量场满足诸如有界加速度等附加的约束。

例 8.17（**二次型势函数**）作为一个简单的例子，假设 $X = \mathbb{R}^2$，没有障碍物，$q_{goal} = (0,0)$。一个二次函数 $\phi(x,y) = \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2$ 就是一个非常好的势函数来引导状态向目标移动。反馈运动策略定义为 $f = -\nabla\phi = \begin{bmatrix} -x_1 & -x_2 \end{bmatrix}$，它是图 8.5（b）所示的向内向量场。

如果目标是某 $(x_1', x_2') \in \mathbb{R}^2$，那么引导状态向目标移动的一个势函数是：

$$\phi(x_1, x_2) = (x_1 - x_1')^2 + (x_2 - x_2')^2$$

■

假设状态空间表示一个包含点障碍物的位形空间。前面的函数 ϕ 可以认为是一个吸引

势，因为位形被吸引到目标。还可以构造一个排斥势，它将位形从障碍物排斥开来避免碰撞。令 ϕ_a 表示吸引分量，ϕ_r 表示在所有障碍点上求和的排斥势。定义形式为 $\phi=\phi_a+\phi_r$ 的势函数来组合这两种效果。机器人应该被引导到目标，同时避开障碍物。问题是通常很难保证势函数不包含多个局部极小点。位形可能陷入一个局部极小点，而该局部极小点并不是目标区域内的点。这是第 5.4.3 节的规划器的问题。

8.4.2 胞腔复形上的向量场

本节描述如何在一个胞腔复形上构造一个分段平滑向量场。这里只考虑归一化的向量场。假设复形中的每个胞腔的形状很简单，在胞腔上很容易定义一块向量场。在一些情况下，第 6 章的胞腔剖分技术可以用来构造一个反馈规划。

假定一个 n 维状态空间 X 已经剖分为一个胞腔复形，诸如单纯复形或者奇异复形，如第 6.3.1 节所定义。假设目标集合是一个单点 x_G。在 X 上定义一个反馈规划 π 要求在 X 上放置一个向量场，它的所有积分曲线都导向 x_G（如果 x_G 是可达的）。这可以通过为每个 n-胞腔定义一个平滑向量场来完成。每个(n-1)-胞腔都是第 8.3.1 节所考虑的一个切换边界。这直接产生 Filipov 意义下的解轨迹。如果允许 π 可以是不连续的，那么在维数为 $n-1$ 或者更小维数的任何胞腔上指定值实际上并不重要了。

用层次方法来构造 π：

（1）在 n-胞腔上定义一个离散规划问题。包含 x_G 的胞腔指定为目标，离散导航函数定义在胞腔上。

（2）在每个 n-胞腔上定义一个向量场。该向量场应该使得胞腔中的所有状态流入下一个胞腔，就像离散导航函数那样。

在应用中一个非常重要的考虑是试图尽可能减小经过边界时由不连续所产生的影响。消除不连续是可能的；甚至可以在 n-胞腔之间构造一个平滑过渡。这里不考虑这个问题，但是它非常重要[237,646]。

现在正式描述该方法。假设在一个连续状态空间 X 上已经定义了一个胞腔复形。令 $\breve{X}$ 表示 n-胞腔的集合，它可以认为是一个有限状态空间。在 $\breve{X}$ 上定义一个离散规划问题。为了避免与原始的连续问题相混淆，给离散规划分量加一个前缀“超”。每个超状态 $\breve{x}\in\breve{X}$ 对应于一个 n-胞腔。从每个 $\breve{x}$，对每个邻近的 n-胞腔存在一个超行动 $\breve{u}\in\breve{U}(\breve{x})$（邻近是指两个胞腔必须分享一个($n$-1)-维边界）。令目标超状态 $\breve{x}_g$ 为包含 x_G 的 n-胞腔。假设给离散问题定义代价泛函以使每个行动（除了 u_T）产生一个单位代价。现在第 8.2 节的概念就可以应用于离散问题。一个离散导航函数 $\breve{\phi}:\breve{X}\to\mathbb{R}$ 可以用 Dijkstra 算法（或者另外一个算法，特别是如果最优性并不重要时）进行计算。使用第 8.2.2 节的离散局部算子产生离散反馈规划 $\breve{\pi}:\breve{X}\to\breve{U}$。

基于离散反馈规划，有两种类型的 n-胞腔。第一种是目标胞腔 $\breve{x}_g$，需要定义一个向量场以使所有积分曲线在有限时间内都导向 X_g[1]。当到达 x_G 时，采取终止行动。其余的 n-胞腔是第二种类型。对每个胞腔 $\breve{x}$，与应用 $\breve{u}=\breve{\pi}(\breve{x})$ 所到达的胞腔共享的边界称为出口面（exit

1 在有限时间内这是可能的，即使 X_g 是一个单点，因为向量场不是连续的。否则，只可能渐近收敛。

face)。必须定义 n-胞腔 $\bar{x}$ 上的向量场以使所有积分曲线导向出口面。当到达出口面，那么就转移到下一个 n-胞腔。如果 n-胞腔是凸的，那么很容易定义该转移（除非对场有附加的要求，如要求在边界上平滑等）。对于更复杂的胞腔，可以定义一个向量场，该向量场将所有点收缩到胞腔中的一条曲线上。

该方法的一个简单的例子是为 $X = C_{free} \subset \mathbb{R}^2$，其中 C_{free} 的边界是多边形。第 6.2 节考虑了这种运动规划问题，但是没有反馈。假设已经计算 X 的一个三角剖分，如第 6.3.2 节所描述。图 6.16 显示了一个例子。图 8.10 显示了为一个特殊目标状态而进行的离散反馈规划。每个 2-胞腔（三角形）用指向下一个胞腔的箭头来标注。

对于包含 x_G 的胞腔，图 8.5（b）所示的向内向量场除以它的量值就可以进行归一化。然后将它的原点平移到 x_G。对于其余的 2-胞腔，必须构造一个流入相应相邻胞腔的向量场。图 8.11 阐述了一个简单的实现方法。将图 8.5（b）所示的场取反产生一个向外的向量场 $f = [x, y]$。对它进行归一化和平移来将原点移动到与出口边无关联的三角形顶点。在图 8.11 中它被称为排斥顶点（repulsive vertex）。这产生了一个将三角形中所有点排斥到出口边的向量场。如果用这种方式为每个三角形都构造这样的场，那么全局向量场就表示了问题的一个反馈规划解。积分曲线（在 Filipov 意义下）在有限时间内导向 x_G。

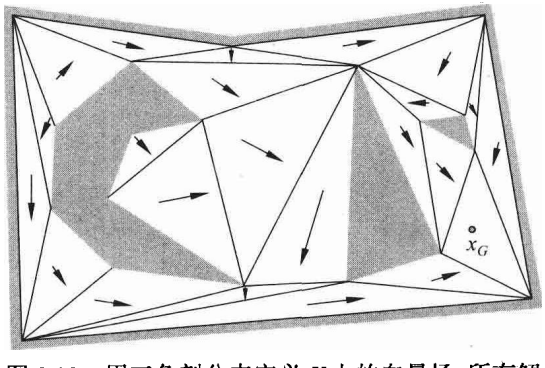


图 8.10　用三角剖分来定义 X 上的向量场；所有解轨迹都导向目标

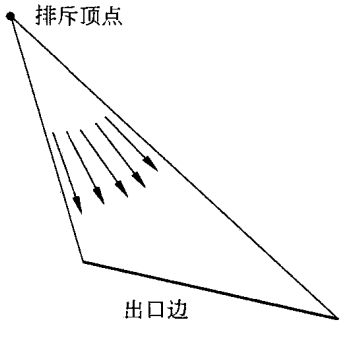


图 8.11　每个三角形的向量场可以用排斥与出口边相对的顶点来定义

8.4.3　最优导航函数

第 8.4.2 节的向量场产生可行轨迹，但不一定是最优轨迹，除非初始状态和目标状态在同一个凸 n-胞腔中。如果 $X = \mathbb{R}^2$，那么可能产生 Dijkstra 算法的一个连续版本[711]。基于到达目标 x_G 的欧氏最短路径，这产生了 X 上的一个精确的尚需代价函数。尚需代价函数用作导航函数，对该导航函数采用局部最速下降法就可以定义反馈规划。

假设 X 以一个简单的多边形为界（无孔）。假设仅允许归一化的向量场。假设代价泛函就是沿着状态轨迹运动的欧氏距离。回忆第 6.2.4 节中最优路径规划，必须采用 $X = cl(C_{free})$。假设 C_{free} 和 $cl(C_{free})$ 有相同的连通性[1]。这会影响第 8.3 节的切空间的定义，

1　这排除了 C_{free} 的一个选择，对于该选择来说，增加一个边界点使得一条不同同伦的路径经过该边界点。这种情况的一个例子是当 $\mathbb{R}^2$ 中的正方形障碍物仅在一个点对上发生接触。

因为 X 的每个点都必须包含在一个开邻域中。尽管如此，我们允许沿着边界的向量，只要它们“指向”与边界相切的方向。通过将边界区域当作单独的流形就可以正式定义它。

考虑为诸如图 8.12（a）所示的问题计算到一个点 x_G 的最优尚需代价。对于任意 $x \in X$，令**可视多边形**（visibility polygon）$V(x)$ 表示从 x 可视的所有点的集合，如图 8.12（b）所描述。一个点 x' 处在 $V(x)$ 中，当且仅当从 x' 到 x 的线段包含在 X 中。这意味着从 x' 到 x 的尚需代价就是从 x' 到 x 的欧氏距离。因此最优导航函数在 $V(x_G)$ 上可以定义为：

$$\phi(x) = \|x - x_G\| \tag{8.42}$$

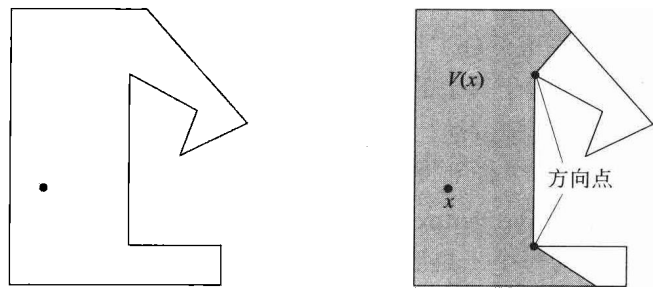


图 8.12 （a）一个简单多边形中的点 x；（b）可视多边形 $V(x)$

该导航函数在等间隔的值上的水平集如图 8.13（a）所示。

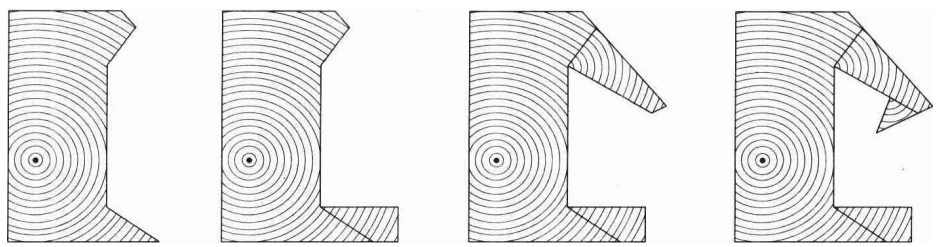


图 8.13 4 次迭代计算得到最优导航函数；在每次迭代中，导航函数从一个新的方向点进行扩展

如何计算 $X \setminus V(x_G)$ 中的点的最优尚需代价值？对于任意 $x \in X$ 的 $V(x_G)$ 的边界上的线段，一些边在 X 的边界上，而另外一些则穿入 X 的内部。对于图 8.12（b）中的例子，有两条边穿入内部。对于每条穿入内部的线段，令两个顶点中距离 x 较近的一个称为方向点（way point）。图 8.12（b）显示了两个方向点。$V(x_G)$ 的方向点是某些最优路径必须经过的地方。令 $W(x)$ 表示 $V(x)$ 的方向点的集合，$x \in X$。

一个显而易见的算法过程如下。令 Z_i 表示在算法的第 i 次迭代中已经定义了 ϕ 的那些点的集合。在第一次迭代中，$Z_1 = V(x_G)$，如图 8.13（a）所示的情形。$V(x_G)$ 的方向点放置在一个队列 Q 中。在每次迭代中，从 Q 中删除一个方向点 x。令 Z_i 表示 ϕ 的定义域。扩展 ϕ 的定义域使之包括所有从 x 可视的新的点。这些新的点是 $V(x) \setminus Z_i$。这产生了 ϕ 的扩展定义域 $Z_{i+1} = Z_i \cup V(x)$。对 $x' \in Z_{i+1} \setminus Z_i$，$\phi(x')$ 的值定义为：

$$\phi(x') = \phi(x) + \|x' - x\| \tag{8.43}$$

其中 x 是从 Q 中删除的方向点（已经计算了 x 的最优尚需代价值）。不在 Z_{i+1} 中的 $V(x)$ 的

方向点加入到 Q 中。这些点产生尚未看见的 X 的新的部分。当 Q 为空时，算法结束，即对于某 k，$Z_k = X$。算法的执行如图 8.13 所述。

如果 X 用 n 条边来描述，那么可视多边形可以在 $O(n \lg n)$ 时间内计算出来。通过一种被称为径向扫描（radial sweep）的方法来实现，径向扫描是对第 6.2.2 节为垂直胞腔剖分所采用的方法的适应性修改。计算 $V(x)$ 的不同之处在于 x 处的射线是径向扫描的（就像雷达扫描）。与射线相交的线段以它们到 x 的距离进行排序。对于构造导航函数的算法，计算的可视多边形不超过 $O(n)$ 个，因为每个可视多边形从唯一一个方向点计算而来。那么整个算法的运行时间是 $O(n^2 \lg n)$。遗憾的是，不能扩展到高维，回忆第 7.7.1 节中在三维环境下计算最短路径是 NP 复杂度的[174]。

这里给出的算法很容易描述，但是它不是最一般的，也不是最有效率的。如果 X 有孔，那么在一个障碍物附近运动时水平集曲线会因为从不同方向到达而发生冲突。上面所描述的队列 Q 可以像 Dijkstra 算法那样进行排序，当尚需代价向外传播而发生关键事件时还需要特殊数据结构来对它们进行识别。从文献[446]可以看到这将在 $O(n \lg n)$ 时间和 $O(n \lg n)$ 空间内完成。

8.4.4　考虑动力学的向量场

如果动力学是一个重要的因素，那么就不希望有目前所考虑的不连续向量场。由于动量的缘故，一个机械系统不能瞬间改变它的速度（见第 13.3 节）。这时的向量场应该满足诸如平滑或者有界加速度等附加的约束。这只代表考虑动力学的其中一步。完整考虑动力学因素则由第Ⅳ部分给出，在那里动力系统的精确运动方程将作为模型的一部分予以描述。本节的方法是构造动力学的向量场，而不是仔细考虑特定的运动方程。

通过定义一个在诸如圆盘等简单区域上满足某些所期望的约束的导航函数，就可得到一个框架[832]。然后设计一个变换的集合，当将导航函数应用于更加复杂的环境而需要进行适应性修改时，这些变换被证明能够保护这些约束。对于一个给定的问题，从某种情形开始并应用合适的变换序列就可以得到构造导航函数的完整算法。

本节关注于在这种基于变换的框架下所保持的约束。第 8.4.2 节和第 8.4.3 节采用的是归一化的向量场。在这种约束条件下，实际上可以定义任何向量场，只要算法所构造的场的积分曲线在 Filipov 意义下存在。在本节中，我们删除向量场必须归一化这个约束，而考虑其他约束。当将向量场用作反馈规划时，现在假设向量场所给出的速度代表必须执行的真实速度。

给向量场增加约束的一个含义是最优解可能不满足这些约束。例如第 8.4.3 节的最优导航函数导致不连续向量场，这违背了这里所考虑的约束条件。所要求的约束限制了可允许的向量场的集合。因此最优性必须定义在受限向量场集合上。在某些情况下，可能不存在最优解（见第 9.1.1 节关于最优性和开集合的讨论）。因此本节只关注可行解。

8.4.4.1　一个基于加速度的控制模型

为了引入约束，考虑文献[237,833]提出的一个控制模型。在表述 8.2 中定义的行动空间 $U(x) = T_x(X)$ 为每个行动 $u \in U(x)$ 产生一个速度。因此 $\dot{x} = u$。假设每个行动产生一个加

速度。这可以表示为 $\ddot{x}=u$，其中 $\ddot{x}$ 是一个加速度向量：

$$\ddot{x}=\frac{\mathrm{d}\dot{x}}{\mathrm{d}t}=\left[\frac{\mathrm{d}^2x_1}{\mathrm{d}t^2}\quad\frac{\mathrm{d}^2x_2}{\mathrm{d}t^2}\quad\cdots\quad\frac{\mathrm{d}^2x_n}{\mathrm{d}t^2}\right] \tag{8.44}$$

通过积分可以得到速度。对式（8.44）积分两次就可得到状态轨迹 $\tilde{x}:T\to X$。

假设向量场以 $\dot{x}=f(x)$ 的形式给出。那么如何产生一个反馈规划？考虑当 x 变化时，由 $f(x)$ 指定的速度向量是如何改变的。假设 $f(x)$ 是平滑的（或者至少是 C^1），并且令：

$$\nabla_{\dot{x}}f(x)=\left[\nabla_{\dot{x}}f_1(x)\quad\nabla_{\dot{x}}f_2(x)\quad\cdots\quad\nabla_{\dot{x}}f_n(x)\right] \tag{8.45}$$

其中 $\nabla_{\dot{x}}$ 表示在 $\dot{x}$ 方向上非归一化的方向导数：$\nabla f_i\cdot\dot{x}$。假设给定初始条件 x_I 和初始速度 $\dot{x}=f(x_I)$。反馈规划可以定义为：

$$u=\nabla_{\dot{x}}f(x) \tag{8.46}$$

这等价于第 8.4.1 节的关于反馈规划的定义；仅有的不同之处是现在需要两个积分（要求 x 和 $\dot{x}=f(x_I)$ 作为初始条件），并且向量场必须满足可微性条件。

现在重新定义 $\dot{x}$ 和 $f(x)$ 之间的关系。假设 $\dot{x}$ 是在执行过程中的真实测量速度，并且 $f(x)$ 是从向量场 f 得到的速度。在执行过程中，假设 $\dot{x}$ 和 $f(x)$ 不一定相同，但是任务是保持它们之间尽可能近。由于存在未建模动态，所以它们之间有差异。例如，如果场 $f(x)$ 要求速度必须突然改变，一个机器人由于有动量而不可能急转弯。

采用新的方法进行阐述，差分 $f(x)-\dot{x}$ 可以被认为是一个差异或者误差。假设已经计算了一个向量场。一个反馈规划就变成基于加速度的控制模型：

$$u=K(f(x)-\dot{x})+\nabla_{\dot{x}}f(x) \tag{8.47}$$

其中 K 是个标量增益常数。一个大的 K 将使控制系统更加强烈地减小误差。如果 K 太大，那么可能违背加速度或者能量约束。如果 $\dot{x}=f(x)$，那么 $u=\nabla_{\dot{x}}f(x)$，它等价于以前的表述。

8.4.4.2 速度和加速度约束

考虑基于加速度的控制模型，对可允许向量场的集合可以进行一些约束。一个有界速度模型意味着 $\|\dot{x}\|<v_{\max}$，正实数 $v_{\max}$ 称为最大速度。这表示由于安全原因，机器人有一个最大速度。也可能限制速度向量的某个分量。例如对飞行器的最大角速度和线速度有不同的限制。直觉地，速度约束表示定义向量场的函数 f_i 不能取最大值。

一个有界加速度模型意味着 $\|\ddot{x}\|\leqslant a_{\max}$，正实数 $a_{\max}$ 被称为最大加速度。从直觉上看，这意味着当沿着积分曲线运动时，速度不能变化太快。采用控制模型 $\ddot{x}=u$，意味着 $\|u\|\leqslant a_{\max}$。这施加了这样一个约束：对所有 $\dot{x}$ 和 $x\in X$，向量场必须满足 $\|\nabla_{\dot{x}}f(x)\|\leqslant a_{\max}$。条件 $\|u\|\leqslant a_{\max}$ 在实际应用中非常重要，因为更高的加速度通常非常昂贵（要求更大的发动机和消耗更多的燃料等）。行动 u 可能直接对应于发动机的扭矩。在这种情况下，每个发动机通常有一个上限。

对速度设置一个上界通常不影响解的存在性。假设一个机器人总能够决定用更慢的速度运动。如果还有加速度的上界，那么机器人可以用更慢速的运动来满足这个界限。假设汽车为转弯而减速。如果你希望速度更快，那么满足加速度约束就更加困难。尽管如此，在大多数情况下，人们都喜欢更快的速度。

一个不连续的向量场不能满足任何加速度界，因为在不连续处它实质上是要求加速度无穷大来产生速度向量的不连续跳跃。如果向量场满足某常数 C 下的 Lipschitz 条件式（8.16），那么如果 $C < a_{\max}$，它满足加速度界。

在第 13 章中，我们将精确确定每个 $x \in X$ 的 $U(x)$，它比施加简单的速度和加速度界更加具有一般性。这也使得可以对任何物理系统进行建模。

8.4.4.3　Rimon-Koditschek 意义下的导航函数

现在考虑构造一个导航函数，该导航函数所生成的向量场满足基于加速度的控制模型式（8.47）所产生的约束。通常地，导航函数的定义从一个势函数 $\phi : X \to \mathbb{R}$ 开始。势函数应该具有什么性质才能成为一个第 8.4.1 节定义的导航函数，并且产生一个满足加速度界的向量场？即将给出的充分条件意味着势函数就是一个满足界限的导航函数。

为了给出条件，首要的是刻画多变量函数极值的特性。从基本微积分可知：函数 $f : \mathbb{R} \to \mathbb{R}$ 当一阶导数为零时有一个临界点。在该点上，二阶导数的符号指定该临界点是最大值还是最小值。这些思路可以一般化到更高维数。ϕ 的一个临界点是使 $\nabla\phi = 0$ 的点。ϕ 的 Hessian 矩阵定义为：

$$H(\phi) = \begin{pmatrix} \dfrac{\partial^2\phi}{\partial^2 x_1^2} & \dfrac{\partial^2\phi}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2\phi}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2\phi}{\partial x_2 \partial x_1} & \dfrac{\partial^2\phi}{\partial^2 x_2^2} & \cdots & \dfrac{\partial^2\phi}{\partial x_2 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \dfrac{\partial^2\phi}{\partial x_n \partial x_1} & \dfrac{\partial^2\phi}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2\phi}{\partial^2 x_n^2} \end{pmatrix} \tag{8.48}$$

在每个临界点，Hessian 都给出关于极值的一些信息。如果在 x 处的 $H(\phi)$ 的秩为 n，那么 Hessian 指示极值的类型。如果式（8.48）是正定的[1]，那么在 x 处 ϕ 有局部最小值。如果式（8.48）是负定的[2]，那么在 x 处 ϕ 有局部最大值。在所有其他情况下，x 是一个鞍点（saddle point）。如果在 x 处的 $H(\phi)$ 的秩小于 n，那么 Hessian 是退化的。在这种情况下，Hessian 不能区分极值的类型。当 x 处在稳定状态（ϕ 既没有增加的方向，也没有减少的方向。这种行为对势函数来说显然是不好的，因为局部算子无法选择方向）时，会发生这种情况。

假设要求导航函数的平滑性来保证每个点上梯度的存在性。这使得能够进行梯度下降。如果 X 不是可缩的，那么一定有一些 x_G 以外的临界点使得 $\nabla\phi(x) = 0$。这些临界点可以用来推断 X 的拓扑，这是 Morse 理论的基本思路[704,236]。遗憾的是，这暗示对于这种空间，不存在导航函数解，因为第 8.4.1 节的定义要求：通过使用导航函数产生的向量场，从能够到达 x_G 的任何状态开始的积分曲线必须要到达目标。如果初始状态是一个临界点，那么积分曲线是常数（状态保持在临界点上）。因此在平滑约束下，应该修改导航函数的定义使得只在很少地方有临界点（仅在零测度集合上）。从一些状态开始的积分曲线能够到达每个临

1　$n \times n$ 矩阵 A 正定意味着对于所有 $x \in \mathbb{R}^n$，有 $x^T Ax > 0$。如果 A 是对称的（应用于 $H(\phi)$），那么这等价于 A 的特征值都是正的。

2　负定意味着对于所有 $x \in \mathbb{R}^n$，有 $x^T Ax < 0$。如果 A 是对称的，那么这等价于 A 的特征值都是负的。

界点，因此还要求这样的状态的集合（每个临界点的吸引域）有零测度。从所有可能的初始状态，除了零测度集，如果 x_G 是可达的，积分曲线都必须到达 x_G。在如下定义中，这将得到保证。

一个函数 $\phi: X \to \mathbb{R}$ 称为 Riman-Koditschek 意义下的导航函数，如果[832]：

（1）它是平滑的（或者至少是 C^2）。

（2）在包含 x_G 的 $\mathcal{C}_{free}$ 的连通分支上的所有值中，只有一个局部最小值点 x_G[1]。

（3）在 $\mathcal{C}_{free}$ 的边界 $\partial\mathcal{C}_{free}$ 上，函数值最大且不变。

（4）它是 Morse 函数[704]，这意味着在每个临界点 x（例如 $\nabla\phi|_x = 0$），ϕ 的 Hessian 不是退化的[2]。这种函数存在于任何平滑流形上。

如果 ϕ 在 C^∞ 意义下是平滑的，那么根据 Sard 定理[236]可知：临界点的集合有零测度。

文献[832]为一般问题族概述了构造导航函数的方法，在这些问题中 $\mathcal{C}_{free}$ 采用半代数描述。基本思路是从容易定义导航函数的简单形状开始。一个这样的例子是 $\mathbb{R}^n$ 的一个球形子集，它包含球形障碍物。一个变形变换的集合将导航函数进行适应性修改成其他形状，同时确保上面所说的 4 个性质得以保持。这种变换将一个球扩展为任何可视区域（在第 8.4.3 节所定义的意义下）。这通过将球平滑地展开成可视区域的形状来完成（这种区域有时称为星形的）。将文献[832]中给出的变换组合起来就可以为一个大的位形空间族定义导航函数。主要问题是位形空间障碍物和 $\mathcal{C}_{free}$ 的连通性是隐式表示的，这使得很难将该方法正确应用于复杂高维问题。该方法的一个优势是简化了对收敛于目标的证明。在许多情况下，可以进行李亚普诺夫稳定性分析（见第 15.1.1 节）。

8.4.4.4 调和势函数

导航函数的另外一个重要的族是从调和函数构造而来[238,241,242,484,532]。一个函数 ϕ 称为调和函数，如果它满足微分方程：

$$\nabla^2\phi = \sum_{i=1}^{n} \frac{\partial^2\phi}{\partial x_i^2} = 0 \tag{8.49}$$

该方程有许多可能的解，这取决于沿着 ϕ 的定义域的边界的条件。文献[237]给出一个简单的基于圆盘的例子，它有一个解析解。复杂的导航函数一般是通过沿着 $\mathcal{C}_{free}$ 的边界给 ϕ 施加约束来定义的。一个 Dirichlet 边界条件意味着边界必须保持常数值。使用该条件，一个调和导航函数能够将状态从一个单连通的状态空间的任何位置引导到目标区域。如果有内部障碍物，那么一个 Neumann 边界条件迫使速度向量与障碍边界相切。在两个边界条件的组合条件下求解式（8.49），可以构造一个调和导航函数，该函数用与障碍物的边界平行运动的方式来避开障碍物，最后到达目标。在一般情况下都可以产生导航函数[242,241]；但是主要问题是 $\mathcal{C}_{free}$ 的边界通常不能显式构造（回忆第 5 章为什么避免这种问题）以及式（8.49）的数值解的计算开销很大。例如这可以通过采用 Gauss-Seidel（如文献[242]中所指出）迭代来完成，它与值迭代有关（不同点见文献[98]）。文献[126]提出了一个基于采样的方法通过调和函数来构造导航函数。第 8.5.2 节将采用值迭代方法来产生近似最优的导航函数。

---

1 有些作者认为全局最小值不包括在局部最小值内。在这种情况下，可以说没有局部最小值。

2 为了成为 Morse 函数，函数在每个临界点处的值必须不同。

8.5　连续空间中基于采样的方法

第 8.4 节中的方法可以认为是第 6 章的组合方法在反馈下的相似版本。尽管这些方法为问题提供了非常好的解决方案，但是它们要么局限于低维问题，要么是问题具有某种特殊结构。这启发人们引入基于采样的方法。本节介绍第 5 章方法的反馈版本。

8.5.1　计算一个漏斗组合

Mason 引入漏斗的概念作为比喻来形容收敛于状态空间中同一个小区域的运动，不管初始位置在哪里[682]，就像漏斗中的沙粒，它们沿着漏斗的斜面到达底部的开口。导航函数可以被想象成引导状态进入目标的一个漏斗。例如图 8.13（d）所描述的尚需代价函数可以认为是一个复杂的漏斗，它沿着一条最优路径将每个沙粒送到目标。

并不是设计单独一个漏斗，而是考虑将状态空间分解为一族简单的、相互重叠的区域。在每个区域上，设计一个漏斗将状态引导到另外一个漏斗（见图 8.14）。作为一个例子，文献[164]中的方法给每个漏斗都构造了一个李亚普诺夫函数（这种函数将在第 15.1.2 节介绍）来保证收敛到下一个漏斗。几个漏斗的组合就可以构造一个反馈规划。从 X 中的某初始状态开始，访问一系列漏斗直到到达目标。每个漏斗的子目标是到达下一个漏斗。最后到达包含目标的漏斗，该漏斗上的导航函数引导到达目标。漏斗比喻是从对不确定性的感测中产生的，在感测不确定性的情形中，漏斗组合就变成原像规划框架[662]，这将在第 12.5.1 节介绍。但是本节假设目前的状态总是已知的。

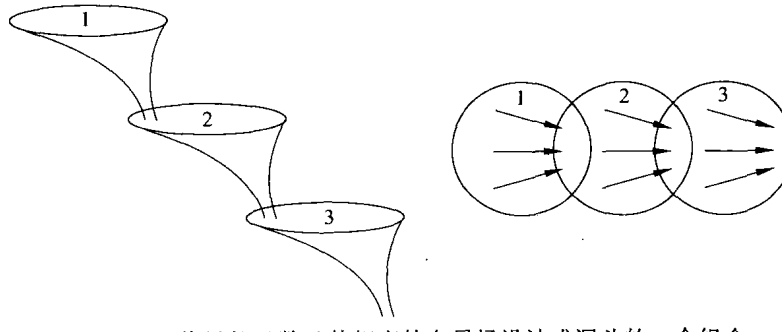


图 8.14　可将导航函数及其相应的向量场设计成漏斗的一个组合

8.5.1.1　一个近似覆盖

图 8.15 阐述近似覆盖的概念，它将用来表示漏斗的定义域。令 $\tilde{X}$ 表示状态空间 X 的一个子集。$\tilde{X}$ 的一个覆盖是集合的一个族 $\mathcal{O}$，该集合：

（1）对每个 $O \in \mathcal{O}$，有 $O \subseteq X$。

（2）$\tilde{X}$ 是覆盖中所有集合的并集的一个子集：

$$\tilde{X} \subseteq \bigcup_{O \in \mathcal{O}} O \tag{8.50}$$

令每个 $O \in \mathcal{O}$ 称为一个邻域。覆盖的概念在第 8.3.2 节用来定义一个平滑流形，所使用的是

坐标邻域的一个覆盖。

一般情况下，覆盖允许：

（1）任意数量的邻域可能相互重叠（有非空的交集）。

（2）任何邻域可能包含 $\tilde{X}$ 以外的点。

第 6.3.1 节介绍的胞腔剖分是一种特殊类型的覆盖，它的邻域形成 $\tilde{X}$ 的一个分拆，邻域必须匹配良好（回顾图 6.15）。

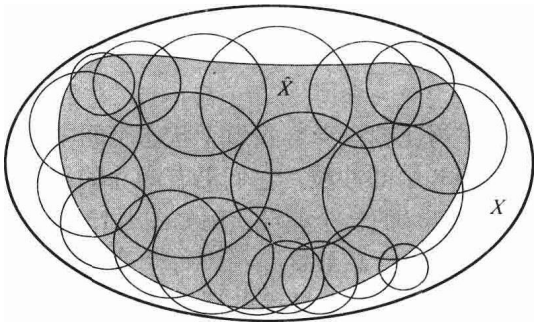


图 8.15 显示一个近似覆盖；$\tilde{X}$ 的每个点至少包含在一个邻域中，$\tilde{X}$ 是 X 的一个子集

到目前为止还没有对邻域进行约束。在实际应用中，选择的邻域应该简化邻域上导航函数的设计。对于原始运动规划问题，所设计的胞腔剖分应该使得在一个胞腔中很容易确定一条无碰撞路径。这里应用同样的思路，只不过我们现在想构造的是一个反馈规划而已。因此通常假设胞腔的形状很简单。

一个覆盖称为是近似的，如果 $\tilde{X}$ 是 X 的一个真子集。在理想情况下，我们想得到一个精确覆盖，它是指 $\tilde{X} = X$ 以及每个邻域都有诸如凸的等这样的良好特性。在实际应用中，对于低维或者具有某种特殊结构的状态空间来说，得到这种覆盖是可能的。这可以从第 6 章介绍的胞腔剖分中观察到。

考虑构造 X 的一个近似覆盖。目标应该是尽可能多地覆盖 X。这意味着 $\mu(X \setminus \tilde{X})$ 应该尽可能得小，其中 μ 表示第 5.1.3 节定义的 Lebesgue 测度。也期望确保 $\tilde{X}$ 能够保持 X 的连通性。也就是说，如果在 X 中存在一条两个点之间的路径，那么在 $\tilde{X}$ 中也应该存在一条相同的路径。

8.5.1.2 在覆盖上定义反馈规划

对第 8.4.2 节的思路可以进行适应性修改，目的是在 $\tilde{X}$ 上用一个覆盖来定义反馈规划。令 $\breve{X}$ 表示离散状态空间，在该空间中每个超状态是一个邻域。相关的离散规划问题的大多数组成部分与第 8.4.2 节中的相同。仅有的不同之处是对超行动的定义，因为在一个覆盖中邻域可以相互重叠。对每个邻域 $O \in \mathcal{O}$，其他的每个邻域 $O' \in \mathcal{O}$ 都存在一个超行动使得 $O \cap O' \neq \phi$（通常它们的内部相互重叠产生 $\text{int}(O) \cap \text{int}(O') \neq \phi$）。

在胞腔剖分情况下，这不会产生超行动，因为它是一个分拆。采用合成漏斗的比喻，某些漏斗的定义域应该重叠，如图 8.14 所示。通过在 O 上定义一个将所有状态从 $O \setminus O'$ 送入 $O \cap O'$ 的向量场，可以得到从一个邻域 O 到另外一个邻域 O' 的转换，如图 8.16 所示。一旦到达 O'，就不再采用 O 的向量场，而是采用 O' 的向量场。采用 O' 的向量场，应用转

换可以到达另一个邻域。从 O 的向量场到 O' 的向量场的跳跃可能使得反馈规划成为 $\breve{X}$ 上的一个不连续向量场。如果所设计的覆盖使得 $O \cap O'$ 很大（如果它们相交），那么通过将 O 和 O' 的向量场进行组合，可能实现逐渐过渡。

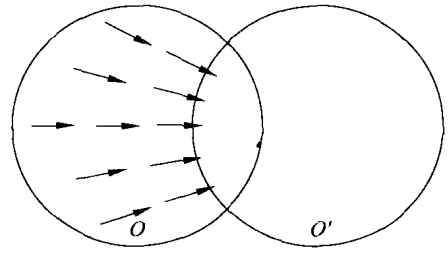


图 8.16　O 上的一个向量场引起从 O 到 O' 的转移，对于该转移来说，所有积分曲线都导向 $O \cap O'$

一旦定义离散问题，那么在 $\breve{X}$ 上就可以计算第 8.2 节所定义的离散反馈规划。通过在每个邻域上定义一个产生相应转换的向量场，$\breve{X}$ 上的反馈规划就可以变换为 X 上的反馈规划。每个 $\breve{x} \in \breve{X}$ 可以作为一个超状态和一个邻域进行阐述。对于每个 $\breve{x}$，离散反馈规划产生一个超行动 $\breve{u} = \pi(\breve{x})$，该行动产生一个新的邻域 $\breve{x}'$。那么设计的 $\breve{x} = O$ 上的向量场使得所有状态进入 $\breve{x}' = O'$。

如果需要，X 上的导航函数 ϕ 甚至可以从 $\breve{X}$ 上的导航函数 $\breve{\phi}$ 中产生。假设构造的 $\breve{\phi}$ 使得对于每个 $\breve{x} \in \breve{X}$ 都有不同的 $\breve{\phi}(\breve{x})$。任何导航函数很容易进行变换来满足这个约束（$\breve{X}$ 是有限的）。令 ϕ_O 表示某 $O \in \mathcal{O}$ 上的一个导航函数。假设 X_G 是一个点 x_G（也可以扩展到更一般的情形）。对于使得 $x_G \notin O$ 的每个邻域 $O \in \mathcal{O}$，定义 ϕ_O 使得进行梯度下降可以进入使 $\breve{\phi}(\breve{x})$ 最小的重叠邻域。如果 O 包含 x_G，那么导航函数 ϕ_O 将状态导引到 x_G。

每个 $O \in \mathcal{O}$ 上的导航函数连接在一起产生所有 X 上的导航函数。在有多个邻域重叠的地方，ϕ 定义为与使得 $\breve{\phi}(\breve{x})$ 最小的这个邻域有关的导航函数。这可以通过给每个 ϕ_O 加一个大的常数来实现。令 c 表示一个常数，使得在所有 $O \in \mathcal{O}$ 和 $x \in O$ 上，$\phi_O(x) < c$（假设每个 ϕ_O 是有界的）。假设 $\breve{\phi}$ 仅采用整数值。令 $\mathcal{O}(x)$ 表示所有使得 $x \in O$ 的 $O \in \mathcal{O}$ 的集合。X 上的导航函数定义为：

$$\phi(x) = \min_{O \in \mathcal{O}(x)} \left\{ \phi_O(x) + c\breve{\phi}(O) \right\} \tag{8.51}$$

8.5.1.3　一个基于采样的方法

有多种方法构造覆盖。为了阐述该思路，这里提出的方法称为基于采样的邻域图（sampling-based neighborhood graph）[983]。假设 $X = \mathcal{C}_{free}$，它是某位形空间的一个子集。如第 5.4 节所介绍的，令 α 为 X 中采样点的一个稠密、无穷序列。假设返回机器人和障碍物之间距离（见式（5.28））的一个碰撞检测算法是可行的。这种算法已在第 5.3 节描述。

图 8.17 给出一个递增算法。初始情况下，$\mathcal{O}$ 为空。在每次迭代中，如果 $\alpha(i) \in \mathcal{C}_{free}$，并且它并不包含在某邻域中，那么一个新的邻域将被加入到 $\mathcal{O}$ 中。两个主要的问题是：（1）如何定义一个新的邻域 O 使得 $O \subset \mathcal{C}_{free}$；（2）什么时候终止。在任意给定时刻，覆盖是近似的。所有邻域的并集是 $\breve{X}$，它是 X 的一个真子集。与图 8.15 相比较，覆盖是一种特殊情况，其中的邻域不能扩展到 $\breve{X}$ 以外。

定义新的邻域　为了定义新的邻域，保持它们简单是非常重要的，因为在执行过程中，必须快速确定包含状态 x 的邻域。假设所有邻域是开球：

$$B(x, r) = \{ x' \in X \mid \rho(x, x') < r \} \tag{8.52}$$

其中 ρ 是 $\mathcal{C}$ 上的度量。对于假设某 $O \in \mathcal{O}$ 的所有邻域是球状的[703]，存在多种高效的算法

构造递增覆盖

1. 初始化 $\mathcal{O}=\phi$ 和 $i=1$。
2. 令 $x=\alpha(i)$，并令 d 为在 x 应用碰撞检测算法所返回的距离。
3. 如果 $d>0$（这意味着 $x\in\mathcal{C}_{free}$）并对于所有 $O\in\mathcal{O}$ 来说 $x\notin O$，那么将一个新的邻域 O_n 插入 $\mathcal{O}$。邻域的大小和形状由 x 和 d 确定。
4. 如果不满足终止条件，那么令 $i:=i+1$，返回第 1 步。
5. 从 $\mathcal{O}$ 中删除那些整个都包含在另外一个邻域的所有邻域。

图 8.17 通过增加新的确保无碰撞的邻域，达到覆盖递增地扩展

来确定是否 $x\in O$。实际上，基于 Kd-树的方法的性能非常好[48,53]（回顾第 5.5.2 节）。对于 $x=\alpha(i)$，一个新球 $B(x,r)$ 可以在第 3 步进行构造，但半径是多大？对于一个在 $\mathbb{R}^2$ 或者 $\mathbb{R}^3$ 中平移的点机器人来说，$\mathcal{W}$ 中机器人和障碍物之间的 Hausdorff 距离 d 就是从 $\alpha(i)$ 到 $\mathcal{C}_{obs}$ 的距离。这意味着我们可以设置 $r=d$，并且确保 $B(x,r)$ 是无碰撞的。

在一般位形空间中，可能找到 r 的一个值使得 $B(x,r)\subseteq\mathcal{C}_{free}$，但是一般情况下 $r<d$。这个问题在第 5.3.4 节检测路径段时出现过。在确定 r 时，第 3.2 节和第 3.3 节的变换变得非常重要。为便于说明，假设 $\mathcal{C}=\mathbb{R}^2\times\mathbb{S}^1$，它对应于在 $\mathcal{W}==\mathbb{R}^2$ 中能够平移和旋转的一个刚体机器人 $\mathcal{A}$。每个点 $a\in\mathcal{A}$ 用式（3.35）进行变换。现在假设从某位形 $q=(x,y,\theta)$ 开始，对每个坐标进行扰动 $\Delta x,\Delta y$ 和 $\Delta\theta$。$\mathcal{A}$ 上的点能够运动的最长距离是多少？平移以相同的方式影响所有点，但是旋转对不同点的影响是不同的。回忆第 5.3.4 节中的图 5.12。令 $a_r\in\mathcal{A}$ 表示距离原点(0,0)最远的点。令 r 表示从 a_r 到原点的距离。如果对旋转进行小量扰动 $\Delta\theta$，那么任意点 $a\in\mathcal{A}$ 的位移不超过 $r\Delta\theta$。如果所有 3 个位形参数都受到扰动，那么：

$$(\Delta x)^2+(\Delta y)^2+(r\Delta\theta)^2<d^2 \tag{8.53}$$

是必须满足的约束来保证所产生的球包含在 $\mathcal{C}_{free}$ 中。这实际上是椭球体的方程，如果 $r=1$ 它就变成球体。通过参数表示为 $SE(2)$，它可以做成一个球，使得 $\Delta\theta$ 的影响和 $\Delta x,\Delta y$ 相同。变换 $h:\theta\mapsto r\theta$ 将 θ 映射到一个新的域 $Z=[0,2\pi r)$。在这个新的空间中，球的方程是：

$$(\Delta x)^2+(\Delta y)^2+(\Delta z)^2<d^2 \tag{8.54}$$

其中 Δz 表示 $z\in Z$ 中的变化。式（3.35）的参数表示是：

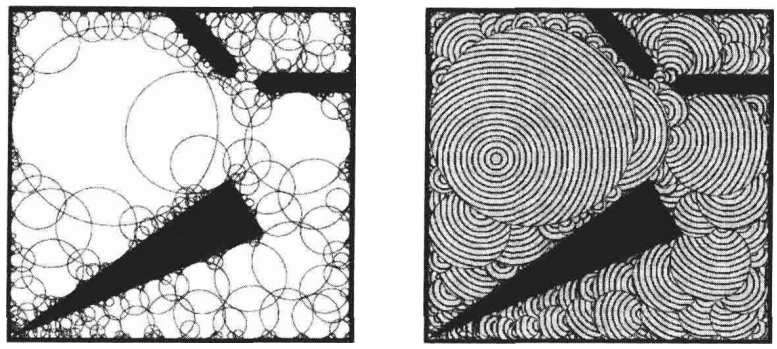
$$T=\begin{pmatrix}\cos(\theta/r) & -\sin(\theta/r) & x_t\\ \sin(\theta/r) & \cos(\theta/r) & y_t\\ 0 & 0 & 1\end{pmatrix} \tag{8.55}$$

对于三维刚体，对欧拉角或者四元数进行相似的参数表示可以产生六维球（注：三维位置、三维角度）。也可以扩展到物体链的情况[983]。主要问题在于用包围球来表示这种扰动距离并不是最合理的。在高维情况下，问题变得更加糟糕，因为需要大量的球，并且上述构造的半径趋向于比可能的半径更小。如果采用轴向圆柱，那么可以减小球的数量，但是在多于六维的 $\mathcal{C}_{free}$ 上依然很难构造覆盖。

终止 如果使用第 5 章中基于采样的规划算法，那么当发现路径解时工作将终止。在目前情况下，由于我们对所有初始位形所产生的解都感兴趣，这样的情况使得终止复杂化。因为 α 是稠密的，$\mathcal{C}_{free}$ 中没有被覆盖的点的体积趋近于零。在有限次数的迭代之后对近似

的精度进行测量，当得到我们所期望的精度时过程终止。也可能用第 5.6.2 节中的基于可视性采样的路线图方法。使用随机采样点，通过 C_{free} 中覆盖以外的一个采样点的过程中记录失败的百分比可以得到对 C_{free} 的其中一部分的估计。例如，如果在 1000 次迭代中仅产生一次新的邻域，那么可以估计到 C_{free} 的 99.9%已被覆盖。也可以确定高概率的界限。文献[983]给出了终止条件，该条件以概率大于 P_c 的可能性保证至少 C_{free} 的 $\alpha \in (0,1)$ 部分被覆盖。常数 P_c 和 α 将作为参数在算法中给出，当用严格统计检验发现条件已经满足时，过程终止。如果采用确定性采样，那么可以构造基于离散度的终止条件，离散度指示 C_{free} 中最大的、不包含另外一个邻域中心的球。这里基于体积的指标有一个问题：无法保证覆盖能够保持 C_{free} 的连通性。如果 C_{free} 的两个部分通过一个窄通道相连，那么覆盖可能失去一个体积非常小，但是需要用来连接两部分的邻域。

例 8.18（计算漏斗的一个二维例子） 图 8.18 显示了一个由采用随机采样点和图 8.17 中的算法计算而来的二维例子。一旦计算出一个覆盖，那么它就可以用来为各种目标快速计算不同的导航函数和向量场。该例子主要目的是为了进行阐述。对多边形环境的情况，构造基于凸多边形的覆盖是更有效的。 ■



(a)　(b)

图 8.18　（a）二维位形空间的一个近似覆盖；（b）一个导航函数的水平集

8.5.2　带插值的动态规划

本节通过用第 2.3 节所介绍的值迭代方法解决运动规划问题来结束第 II 部分的内容。它已经用在第 8.2 节来得到离散反馈规划。现在通过首先在一个连续状态空间进行插值，然后在一个连续行动空间进行插值，对它进行这种适应性修改使之应用到连续空间。这为运动规划产生一个数值方法来计算最优导航函数和反馈规划。关注的仍然是后向值迭代；但是插值的概念也可能应用于前向迭代。这里的方法将最优反馈运动规划看作是一个离散时间最优控制问题[28,86,153,586]。

8.5.2.1　在连续空间上进行插值

这里所考虑的问题与表述 8.1 相同，除了 X 是连续的以外。假设 X 是有界的，并且假设对于所有 $x \in X$，行动空间 $U(x)$ 是有限的。可以应用后向值迭代。动态规划的自变量和导数与第 2.3 节的相同。所产生的递归与式（2.11）相同，并且为方便起见，递归将重复

进行：

$$G_k^*(x_k)=\min_{u_k\in U(x_k)}\left\{l(x_k,u_k)+G_{k+1}^*(x_{k+1})\right\} \tag{8.56}$$

存在的困难是不能为每个 $x_k\in X$ 存储 $G_k^*(x_k)$，因为 X 是连续的。有两种方法解决该问题。一种方法是用曲面的一个参数族来近似 G_k^*，如多项式或者从神经网络产生的非线性基函数[99]。另外一种方法是仅在有限采样点集上存储 G_k^*，并且用插值来得到其他点的值[585,586]。

假设用采样点的一个有限集合 $S\subset X$ 来表示 X 上的尚需代价函数。图 8.19 显示了用插值对式（8.56）所进行的计算。一般情况下，选择的采样点应该尽可能减小离散度（见第 5.2.3 节的定义）。这可防止试图在大面积不包含采样点的区域上近似尚需代价函数。收敛速率从根本上是依赖于离散度[94]（结合状态转移方程上的 Lipschitz 条件和代价泛函）。为了简化符号和其他一些问题，假设 S 是 $\mathbb{R}^n$ 中等间隔点的一个栅格。

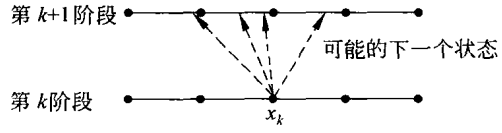


图 8.19 即使 x_k 是一个采样点，下一个状态 x_{k+1} 也可能落在采样点之间；对于每个 $u_k\in U(x_k)$，可能需要对所产生的下一个状态 $x_{k+1}=f(x_k,u_k)$ 进行插值

首先，考虑 $X=[0,1]\subset\mathbb{R}$ 的情况。令 $S=\{s_0,s_1,\cdots,s_r\}$，其中 $s_i=i/r$。例如，如果 $r=3$，那么 $S=\{0,1/3,2/3,1\}$。这总在 X 的边界上产生点，这确保对于(0,1)中的任意点，在该点的上方和下方都有采样点。令 i 是使得 $s_i<x$ 的最大整数。这暗示 $s_{i+1}>x$。采样点 s_i 和 s_{i+1} 称为 x 的插值邻点。

通过线性插值可以得到式（8.56）中 G_{k+1}^* 在任意 $x\in[0,1]$ 处的值：

$$G_{k+1}^*(x)\approx\alpha G_{k+1}^*(s_i)+(1-\alpha)G_{k+1}^*(s_{i+1}) \tag{8.57}$$

其中对系数 $\alpha\in[0,1]$ 的计算方法是：

$$\alpha=1-\frac{x-s_i}{r} \tag{8.58}$$

如果 $x=s_i$，那么 $\alpha=1$，并且式（8.57）将变成 $G_{k+1}^*(s_i)$。如果 $x=s_{i+1}$，那么 $\alpha=0$，并且将式（8.57）变成 $G_{k+1}^*(s_{i+1})$。在它们之间的所有点，式（8.57）用 α 作为权重来组合 s_i 和 s_{i+1} 处的尚需代价值。

插值的思路可以很自然地扩展到多维情况。令 X 为 $\mathbb{R}^n$ 上一个有界子集。令 S 表示 $\mathbb{R}^n$ 中点的一个 n 维栅格。S 中的每个采样点用 $s(i_1,i_2,\cdots,i_n)$ 表示。对于某 $x\in X$，有 2^n 个插值邻点“包围”着它。这些是包含 x 的 n 维方体的角。令 $x=(x_1,\cdots,x_n)$。令 i_j 表示 $s(i_1,i_2,\cdots,i_n)$ 的第 j 个坐标不超过 x_j 时所对应的最大整数。对于每个 $j\in\{1,\cdots,n\}$，2^n 个采样点是所有那些在表达式 $s(\cdot,\cdot,\cdots,\cdot)$ 中出现 i_j 或者 i_{j+1} 的点。这要求对于所有这些情况，S 中存在采样点。X 可能是 $\mathbb{R}^n$ 的一个复杂子集，只要对于任意 $x\in X$，所要求的 2^n 个插值邻点都在 S 中。使用 2^n 个插值邻点，通过多线性插值可以得到在任意 $x\in X$ 上式（8.56）中 G_{k+1}^* 的值。在 $n=2$ 情况下，这表示为：

$$G^*_{k+1}(x) \approx \alpha_1\alpha_2 G^*_{k+1}(s(i_1,i_2)) + \alpha_1(1-\alpha_2)G^*_{k+1}(s(i_1,i_2+1)) + (1-\alpha_1)\alpha_2 G^*_{k+1}(s(i_1+1,i_2)) + (1-\alpha_1)(1-\alpha_2)G^*_{k+1}(s(i_1+1,i_2+1)) \tag{8.59}$$

其中 α_1 和 α_2 的定义与式（8.58）中 α 的相似，只是分别基于沿着 x_1 和 x_2 方向的距离。高维多线性插值的表达与此相似，但是更加难以表达。也可以采用诸如二次型插值等高阶插值[586]。

遗憾的是，插值邻点的数量随维数 n 呈指数增长。一种改进是将由 2^n 个采样点定义的方体剖分为单形，而不是使用所有 2^n 个插值邻点。每个单形仅有 $n+1$ 采样点作为它的顶点。仅有包含 x 的单形的顶点被声明为 x 的插值邻点；这将计算 $G^*_{k+1}(x)$ 的代价降低到 $O(n)$ 时间。但问题是确定包含 x 的单形可能是一个具有挑战性的点定位问题（在计算几何中这是一个很平常的问题[266]）。如果重心重分采用所有面的中间点来剖分方体，那么点定位问题可以在 $O(n\lg n)$ 时间内解决[265,610,724]，它是对上面所描述的 $O(2^n)$ 时间的一个改进。图 8.20 显示了这种剖分的二维和三维情况下的例子。这有时被称为 Coxeter-Freudenthal-Kuhn 三角剖分。即使对于 n 并不太大的实际情况（典型情况下，$n \leqslant 6$），在实现过程中也能够节省大量时间（甚至对于 $n=3$）。

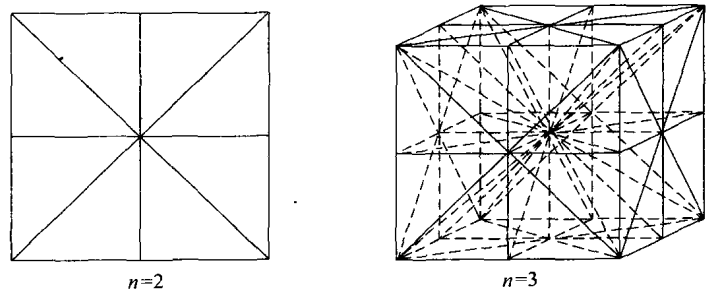


图 8.20　可以用重心重分来将每个方体分拆成单形，它允许在 $O(n\lg n)$ 时间内完成插值，而不是 $O(2^n)$ 时间

直接用 X 中所有点的集合是很方便的，对这些点来说所要求的所有插值邻点都存在。对于采样点的任意有限集合 $S \subseteq X$，令插值区域 $R(S)$ 为所有 $x \in X \setminus S$ 的集合，对该集合来说 $G^*(x)$ 可以用插值来计算。这意味着 $x \in R(S)$，当且仅当 x 的所有插值邻点都在 S 中。图 8.21（a）显示了一个例子。某些采样点可能没有对 R 贡献任何点。如果用采样点的一个栅格来近似 G^*，那么当采样的分辨率增加时，$X \setminus R(S)$ 的体积将接近于零。

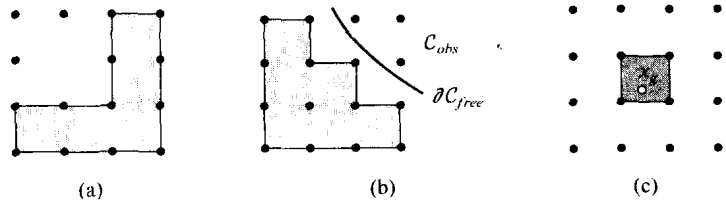


图 8.21　（a）显示了采样点 S 的一个集合的插值区域 $R(S)$；（b）由于障碍物而出现的插值区域；（c）目标点的插值区域一定为空

连续行动空间 除了 X 是连续的以外，现在还假设 $U(x)$ 是连续的。假设 $U(x)$ 是 $\mathbb{R}^n$ 的一个封闭、有界子集。式（8.56）的动态规划递归还是相同的。困难在于 min 表示的是不可数无穷多个选择上的优化问题。一种可能的解决方法是利用非线性优化技术来选择最优 $u \in U(x)$。其效果非常依赖于 $U(x)$、X 和代价泛函。

另外一种方法是从 $U(x)$ 所得到的采样点的一个有限集合上计算式（8.56）。最好选择能够尽可能减小离散度的采样点。在某些情况下，利用尚需代价函数的性质及其插值表示，有可能从所考虑的行动中删除某些行动。

8.5.2.2 与反馈运动规划的联系

现在已经有用值迭代解决运动规划的工具。位形空间是连续状态空间；令 $X = C_{free}$。行动空间也是连续的，$U(x) = T_x(X)$。对于运动规划问题，当应用 u_T 时，只能得到 $0 \in T_x(X)$。因此不需要分别地进行表示。为了计算运动规划的最优尚需代价函数，主要关心：

（1）行动空间必须是有界的。

（2）必须进行离散时间近似来生产一个状态转移方程。

（3）代价泛函必须进行离散化。

（4）必须考虑障碍区域 C_{obs}。

（5）至少有某插值区域必须产生 $G^*(x) = 0$，它表示目标区域。

现在对它们进行讨论。

对行动空间进行限制 回忆使用归一化向量场并不改变解的存在性。这是很方便的，因为需要对 $U(x)$ 进行限制以用采样点的一个有限集合来近似它。限制行动集合可得到：

$$U(x) = \{u \in \mathbb{R}^n \mid \|u\| \leqslant 1\} \tag{8.60}$$

是非常有用的。为了提高性能，有时可能仅使用那些 $\|u\| = 1$ 或者 $u = 0$ 的 u；但是可能会出现数值不稳定性的问题。$U(x)$ 的一个有限采样集合应该是低离散度的，并且总是包括 $u = 0$。

得到一个状态转移方程 在离散阶段会出现值迭代，但是在连续时间出现反馈规划积分曲线。因此需要对时间区间 T 进行采样。令 Δt 表示一个小的正常数，它代表一个固定时间区间。令阶段索引 k 所指的是时间 $(k-1)\Delta t$。现在考虑在 $\mathbb{R}^n$ 上表示一个速度场 $\dot{x}$。根据定义：

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \lim_{\Delta t \to 0} \frac{x(t+\Delta t) - x(t)}{\Delta t} \tag{8.61}$$

在第 8.3.1 节，通过给每个 $x \in X$ 指派某 $u \in U(x)$ 来定义一个速度场。如果速度向量 u 从 $x(t)$ 在 Δt 上进行积分。那么产生一个新的状态 $x(t+\Delta t)$。如果 u 保持常数，那么：

$$x(t+\Delta t) = x(t) + u\Delta t \tag{8.62}$$

这称为一个欧拉近似。如果执行一个反馈规划，那么从 x 通过 $u = \pi(x(t))$ 来确定 u。一般情况下，这意味着当状态向前积分时，u 可以发生变化。在这种情况下，式（8.62）仅仅是近似：

$$x(t+\Delta t) \approx x(t) + \Delta t \cdot \pi(x(t)) \tag{8.63}$$

式（8.62）可以被认为是阶段上的一个状态转移方程。令 $x_{k+1} = x(t+\Delta t)$ 和 $x_k = x(t)$。现在状态转移可以表示为：

$$x_{k+1} = f(x_k, u) = x_k + u\Delta t \tag{8.64}$$

当 Δt 减小时近似的效果会提高。随时间使用更多采样点可以得到更好的近似。最著名的近似是 Runge-Kutta 族。对于最优运动规划，沿着积分曲线的方向向量几乎总是保持常数。例如在图 8.12 中，通过进行最优导航函数的梯度下降可以得到分段线性的路径。在大多数所产生的积分曲线上方向向量是常数（仅在接触障碍物时发生改变）。因此近似的问题在运动规划问题中并不经常出现。当近似诸如第 13 章提出的那些动力系统时，需要更好的近似，见第 14.3.2 节。一个重要的问题是所选择的 Δt 应该与栅格分辨率相兼容。如果 Δt 太小，使得行动对状态的改变不足以产生新的插值邻点，那么插值产生的尚需代价值将保持常数。这意味着 Δt 的选择必须保证 $x(t+\Delta t)$ 的插值邻点集合与 $x(t)$ 的不同。

可以与第 7.7 节的近似运动规划问题联系起来。表述 7.4 对应于这里所定义的近似，除了用 ϵ 代替 Δt，因为没有考虑速度（因为没有反馈，所以需要指定初始条件）。图 7.41 显示了不同的行动空间。就像第 7.7 节所论述，如果采用 Manhattan 或者独立关节模型，那么位形仍然在点的一个栅格上。这使得能够进行离散值迭代。如第 8.2.3 节所述，可以计算一个离散反馈规划和导航函数。如果采用欧氏运动模型，那么转移允许可能位形的一个连续统。这种情况最后可以通过在位形空间上进行插值来处理，如本节所述。

近似代价泛函　必须从连续代价泛函式（8.39）中产生一个离散代价泛函。最后一项指定为 $l_F(x_F)=l_F(x(t_f))$。每个阶段的代价是：

$$l_d(x_k,u_k)=\int_0^{\Delta t} l(x(t),u(t))\mathrm{d}t \tag{8.65}$$

$l_d(x_k,u_k)$ 用在式（8.56）中的 $l(x_k,u_k)$ 的位置上。对于许多问题，不需要重复计算积分。为了得到欧氏最短路径，可以为所有 $x_k \in X$ 和 $u_k \in U(x_k)$ 指定 $l_d(x_k,u_k)=\|u_k\|$。当 Δt 很小时，式（8.65）的一个合理的近似是 $l(x(t),u(t))\Delta t$。

处理障碍物　处理障碍物的一个简单方法是确定每个 $x \in S$ 是否 $x \in \mathcal{C}_{obs}$。在迭代进行之前这些可以计算和存储在一个阵列中。对于刚体机器人来说，这可以用快速傅立叶变换来计算[516]。对于每个 $x \in \mathcal{C}_{obs}$，有 $G^*(x)=\infty$。在这些状态上不进行值迭代，它们的值必须保持无穷大。在式（8.59）的计算过程中（或者一个更高维版本），试图得到不同的行动。对于每个行动，要求 x_{k+1} 的所有插值邻点都在 $\mathcal{C}_{free}$ 中。如果它们中的一个在 $\mathcal{C}_{obs}$ 上，那么行动产生无穷大代价。这对所有顶点都在 $\mathcal{C}_{free}$ 中的所有方体都有自动减小插值区域 $R(S)$ 的效果，如图 8.21（b）所示。在本节的剩余部分都假设 $\mathcal{C}_{obs}$ 中的所有采样点都被删除；但是插值仍然使用整个栅格以使无穷大值表示障碍区域。

就像所表达的那样，$\mathcal{C}_{obs}$ 中的点可能在 $R(S)$中，因为碰撞检测仅在采样点上进行。在实际中，要么产生的栅格分辨率必须足以能够最小化错误发生的机会，或者必须使用从碰撞检测算法所得到的距离信息来推断每个采样点周围一个足够大的球是无碰撞的。如果一个插值区域不能保证在 $\mathcal{C}_{free}$ 中，那么可能不得不至少在局部上增加分辨率。

处理目标区域　后向值迭代从最终尚需代价函数开始向后迭代。初始时，目标区域以外的所有状态的最终尚需代价被指定为无穷大。为了正确初始化最终代价函数，必须存在 X 的某子集，在该子集上通过插值可以得到零值。令 $G=S \cap X_G$，要求插值区域 $R(G)$ 必须非空。如果不能满足条件，那么需要增加栅格分辨率，或者需要扩大目标集合。如果 X_g 是一个点，那么它就需要扩大，无论分辨率怎样（除非得到在目标点附近进行插值的方法）。

在如图 8.21（c）所示的插值区域中，X_G 附近的所有状态产生一个插入的零值尚需代价。如果这种区域不存在，那么从任意状态对式（8.59）进行的计算过程中，所有代价将保持无穷大。Δt 的选择必须足够大以确保新的采样点能够到达 G。

将 G^* 作为导航函数 在尚需代价值稳定以后，所产生的尚需代价函数 G^* 可以当作一个导航函数来使用。即使 G^* 仅定义在 $S \subset X$ 上，在 $R(S)$中任意点上进行插值都可以得到导航函数的值。选择的最优行动需要满足式(8.6)最小化。这意味着状态轨迹不必像 Manhattan 模型那样访问栅格点。轨迹可以访问 $R(S)$中的任意点，这使得当 Δt 和栅格间隔趋近于零时轨迹可以收敛于真实最优解。

拓扑考虑 到现在为止，没有显式考虑 C 的拓扑。假设 C 是一个流形，那么前面所讨论的概念可以应用于定义坐标的任意开集上。实际上，使用第 4.1.2 节的流形表示通常非常方便。流形可以表示为一个方体 $[0,1]^n$，等同它的某些面得到 $[0,1]^n / \sim$。在方体的内部，本节所介绍的所有概念都不需要修改。在边界上，用来插值的采样点必须考虑等同，并且行动 u_k 和下一个状态 x_{k+1} 在边界上必须正常运行。但是在声明某个解是最优的时必须小心，因为欧氏最短距离路径依赖于流形的参数表示。按照诸如时间或者能量等物理指标来表述代价通常可以解决这种模糊。这经常要求对动力学进行建模，这将在第Ⅳ部分进行介绍。

带插值的值迭代是极其一般的。它是近似最优控制问题解的一般算法。通过限制 $U(x)$ 来考虑复杂的微分约束，它就可以用于解决第Ⅳ部分中的许多问题。方法还可以扩展到涉及明确的不确定性的问题。这种值迭代方法将在第 10.6 节进行介绍。

8.5.2.3 获取类似 Dijkstra 的算法

对于用式（8.62）计算的运动规划问题，可以预料相对于 X 的大小来说 $X(t+\Delta t)$ 总是离 $X(t)$很近。这提示可以使用类似 Dijkstra 的算法来更有效地计算最优反馈规划。像第 2.3.3 节中所讨论的有限情形，一些值在值迭代过程中保持不变，如例 2.5 所示。Dijkstra 算法保持一个数据结构，该数据结构将计算集中于值正在发生变化的那部分状态空间。通过仔细考虑采样点，对连续空间也可以采用相同的做法[610]。

在值迭代过程中，和离散情况一样有 3 种采样点（回顾第 2.3.3 节）：

（1）不活动的：尚需代价已经稳定到它的最优值。

（2）活动的：目前的尚需代价是有限的，但是还不知道值是否是最优的。

（3）未访问的：尚需代价值保持无穷大，因为还没有到达采样点。

对连续状态空间，集合的保持存在一定困难，因为在采样点集 S 和插值区域 $R(S)$之间有交集。

想象首次值迭代的情况。初始情况下，G 中的所有点都被设置为零。在采样点的族 S 中，有多少点能够一步到达 $R(G)$？我们预料离 G 非常远的采样点将不能到达 $R(G)$，它们的值保持无穷大。离 G 非常近的采样点能够到达 $R(G)$。所以去掉那些离 G 太远而无法降低它们的值的采样点，这样做是非常方便的。在每次迭代中，对那些远离已经到达的点的采样点不进行迭代。对那些不活动的采样点进行迭代也是不必要的；因为它们的值不再变化。

为了保持对可达采样点的跟踪，引入后向投影（backprojection）的概念，在第 10.1 节将对这个概念做进一步的介绍。对于一个状态 $x \in X$，它的后向投影被定义为：

$$B(x)=\{x'\in X \mid \exists u'\in U(x'),\ x=f(x',u')\} \tag{8.66}$$

点集 $X'\subseteq X$ 的后向投影只是每个点的后向投影的并集：

$$B(X')=\bigcup_{x\in X'} B(x) \tag{8.67}$$

现在考虑的值迭代用后向投影来去除那些值不变的状态。令 i 表示当前值迭代的阶段数。在首次迭代时 i=1，这意味着考虑所有一步轨迹。令 S 表示采样点的集合（假设都不在 C_{obs} 中）。令 D_i 和 A_i 分别表示不活动的和活动的采样点。初始情况下，$D_1=G$，即不活动的采样点为目标集合中采样点的集合。活动的采样点的第一个集合 A_1 用边界的概念来指派。采样点的一个集合 $S'\subseteq S$ 的边界（frontier）为：

$$\text{Front}(S')=(B(R(S'))\setminus S')\cap S \tag{8.68}$$

这是能够一步到达 $R(S')$ 的采样点的集合，排除那些已经在 S' 中的点。图 8.22 阐述了这种边界。通过式（8.68），A_1 被定义为：$A_1=Front(D_1)$。

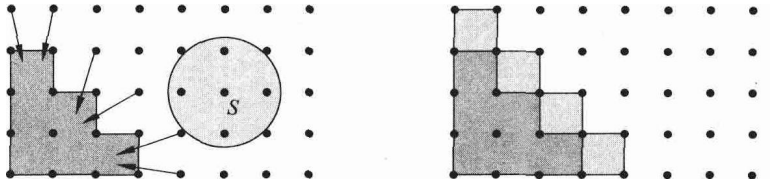


图 8.22　阐述边界的概念：（a）黑点代表从左边的采样点一步能够到达的所有点的集合；采样点在一步之内不能到达右边的阴影区域，阴影区域表示 $R(S')$；（b）边界是能够到达 $R(S')$ 的采样点的集合；边界的包含在 $R(S')$ 之外增加插值区域

现在描述第 i 次迭代。在 A_i 中的所有点上计算尚需代价（更新式(8.56)）。如果对某 $s\in A_i$ 有 $G^*_{k+1}(s)=G^*_k(s)$，那么就等于声明 s 是不活动的，将其移到 D_{i+1}。从来不会从不活动的集合中删除采样点，因此 D_i 中的所有点也都被加到 D_{i+1}。下一个活动集合 A_{i+1} 包括 A_i 中的所有采样点，除了那些被移进不活动集合的采样点以外，并且 Front(A_i) 中的采样点被加到 A_{i+1}，因为这些将在下一次迭代中产生一个有限尚需代价值。迭代继续直到某步 m 的 A_m 为空，D_m 包括所有能够到达目标的采样点（在本节的近似假设条件下）。

为有效起见，对边界进行近似，只要真实边界是近似边界的一个真子集。例如边界可以加入 S' 中点的特定半径内的所有新的采样点。在这种情况下，某 $s\in A_i$ 的更新的尚需代价值可能保持无穷大。如果这种情况发生，那么当然不能加入到 D_{i+1}，并且在下一个边界的计算中从 A_i 中删除（只为有有限尚需代价值的采样点计算边界）。

通过排除不必要的采样点，该方法能够减少每次值迭代中的计算量。在每次迭代中可以得到相同的尚需代价值，因为在每次迭代中只有那些值不能变化的采样点被排除掉。但是所产生的算法并不与 Dijkstra 算法相类似，因为每步值迭代都是在所有 A_i 中上进行的。

为了构造一个行为与 Dijkstra 算法相似的版本，引入一个队列 Q。算法在每次迭代中删除 Q 的最小元素。插值版本首先为每个 $s\in G$ 指定 $G^*(s)=0$。它也保持不活动采样点的一个集合 D，该集合初始化为 $D=G$。对每个 $s\in$ Front(G) 计算尚需代价（更新式（8.56））。优先级 Q 被初始化为 Front(G)，并且元素用它们目前的尚需代价值（可能不是最优的）进

行排序。算法从 Q 中迭代地删除最小元素（因为知道它的最优尚需代价就是目前的值），并且当 Q 为空时迭代终止。每次删除最小元素 $s_s \in Q$，该元素将被插入到 D。然后进行两个步骤。（1）队列中的某些元素需要用式（8.56）重新计算它们的尚需代价值，因为知道值 $G^*(s_s)$ 是最优的，并且尚需代价值可能依赖于它。这是 Q 中处在 Front(G) 中的采样点（其中 D 仅扩展到包括 s_s）。（2）在用式（8.56）计算它们的尚需代价值以后，B($R(D)$) 中不在 Q 中的任意采样点将被插入到 Q 中。这使得活动的采样点的集合随不活动的采样点的增长而增长。带插值的 Dijkstra 算法不计算那些与值迭代所产生的值相同的值，因为当计算 G_k^* 时，G_{k+1}^* 没有显式存储。每个计算的值都是某尚需代价，但是仅当将采样点放置在 D 中时才知道它是最优的。但是能够表明该方法是收敛的，因为计算的值不比在值迭代中已经计算的值大。这也是用 Gauss-Seidel 迭代的动态规划法的基础[98]。

对于发现能够以最小步数到达目标的解这种特殊情况而言，可以产生算法的一个特定的波前传播版本。算法与图 8.4 所示的相似。它从一个初始波前 $W_0 = G$ 开始，其中对于每个 $s \in G$ 都有 $G^*(s) = 0$。在每次迭代中，最优尚需代价值 i 加 1，波前 W_{i+1} 从 W_i 计算而来，$W_{i+1} = \text{Front}(W_i)$。当波前为空的首次迭代时算法终止。

进一步阅读

与前几章相比较而言，与本章相关的文献比较少。如第 8.1 节所述，反馈与运动规划之间相互割裂是有历史原因的。在将反馈引入运动规划时，导航函数[544,832]是最有影响的思路之一，因此导航函数是本章的主题。采用和设计导航函数的其他文献，见文献[208,276,753]。进展测度[319]、李亚普诺夫函数（将在第 15.1.1 节介绍）、尚需代价函数之间是紧密联系的。对于用李亚普诺夫函数来设计反馈控制律，参见文献[280]。在运动规划范畴内，误差检测和恢复(EDR)框架也包含反馈的思想[286]。

在文献[327]中通过刻画覆盖 $C \times C$ 所需要的区域的最小数量来研究 C-空间的拓扑复杂性，其中在每个区域上定义一条连续路径函数。这指出了对可以构造的导航函数所进行的限制，假设 q_I 和 q_G 都是变量（本章的 q_G 都是固定的）。这个方向上进一步的工作包括文献[328,329]。

为了得到对向量场性质的更直观的认识，文献[44]是非常有帮助的，它有大量的富有深刻见解的例证和说明。平滑流形特别适合说明控制论方面的概念，关于平滑流形的一个非常好的入门书籍是文献[135]。从文献[756]中可以得到二维和三维曲线和曲面的基本直观认识。其他关于平滑流形和微分几何的文献包括文献[4,109,236,281,874,907,960]。分段平滑向量场的讨论参见文献[27,637,848,998]。

第 8.4.2 节和第 8.4.3 节的内容分别来源于文献[237,646]和文献[711]。许多问题都会迎刃而解，因为这些方法允许不连续向量场。通过要求连续性和平滑性，得到第 8.4.4 节的内容。它们主要是基于文献[832,833]的。其他关于导航函数的工作包括文献[251,654,655]。

第 8.5.1 节主要是受到文献[164,682]的启发，基于邻域图的方法是从文献[983]中而来的。

第 8.5.2 节中带插值的值迭代有时在运动规划中将被遗忘，因为在它产生的那个年代计

算机的功能还不够强大[86,87,585,586]。但是目前计算机已经能够计算具有挑战性的多维（例如三维或四维）问题的解。文献[94]最先建立收敛到解决最优连续问题的离散化值迭代方法，它是基于状态转移方程上的 Lipschitz 条件和代价泛函。考虑插值的分析问题和一般离散化问题出现在文献[170,294,402,568,570]中。文献[725]提出了值迭代的一个多分辨率形式。对值迭代的类似 Dijkstra 算法的讨论是基于文献[610,946]的。水平集方法与文献[535,536,537,864]有密切关系。

习　题

1. 假设在一个可移动机器人内的电路板上运行着一个非常快的路径规划算法（例如，如果采用 $\mathbb{R}^2$ 上的梯形剖分，那么它可能在数毫秒内找到答案）。介绍应用该方法来模拟机器人上的一个反馈规划。说明在一个计算开环规划的快速在线算法和一个计算反馈规划的更好的离线算法之间所采取的折中。

2. 用 Dijkstra 算法在带障碍物的二维栅格上构造导航函数。为了到达距离障碍物尽可能近的地方，实验给代价泛函加入一个罚函数。

3. 如果有可选路线，那么 NF2 算法不必沿着有最大的最大间隙的路线输出状态。修改 NF2 算法使之处理该问题。

4. 切空间问题：

（a）对于单位四元数的流形，寻找 $\mathbb{R}^4$ 的任意点上的切空间的基向量。

（b）寻找 $\mathbb{R}^9$ 中的切空间的基向量，假设 $SO(3)$ 中的矩阵用四元数，如式（4.20）所示。

5. 将第 8.4.3 节中描述的算法进行扩展使之能够应用于有孔的多边形。相似的问题见例 8.16。

6. 给出一个完整算法，它用 $\mathbb{R}^2$ 中的一个多边形障碍区域的垂直胞腔剖分来构造一个用作反馈规划的向量场。向量场可能是不连续的。

7. 图 8.23 刻画了 X_{free} 是一个开的环面的一个二维例子。考虑设计一个向量场，对它来说所有积分曲线都流向 x_G，并且向量场在 x_G 外是连续的。给出一个这样的向量场，或者解释为什么它是不可能的。

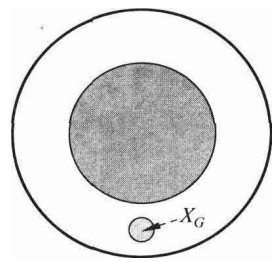


图 8.23　考虑设计一个流进 X_G 的连续向量场

8. 用第 6.2.3 节中的最大间隙路线图思想来定义一个胞腔剖分和最大化间隙的反馈运动规划（向量场）。向量场可能是不连续的。

9. 开发一个计算方法来计算一个多边形位形空间的精确覆盖，确保如果两个邻域相交，那么它们的交集总包含一个开集合（例如重叠区域是二维的）。覆盖中的邻域应该是多边形的。

10. 用一个距离测量和欧拉角，确定一个无碰撞的球的表达式（使球尽可能大）。这应该推广式（8.54）。

11. 用一个距离测量和四元数，确定一个无碰撞的球的表达式（也使之尽可能大）。

12. 将式（8.59）中的多线性插值方案从二维一般化到 n 维。

13. 如果用$\|u\|=1$代替$\|u\|\leqslant 1$来约束可允许的行动的集合，说明这样所产生的值迭代的收敛性问题。

实现

14. 用数值方法来解决在不同边界条件下的二维函数（见式（8.49））。论述方法的有效性和精度。它们应用于高维情况会如何？

15. 为一个能够在$\mathcal{W}=\mathbb{R}^2$中多边形障碍物中平移和旋转的多边形机器人，实现带插值的值迭代（不必使用图 8.20 中的方法）。定义代价泛函使得能够得到关于加权欧氏度量的运动距离（可以任意设定旋转对平移的权值）。

16. 如习题 12 中那样将式（8.59）推广得到多线性插值，计算如图 8.20 所显示的插值方法应用于多线性插值时的效率。不需要实现所有值迭代方法（当然如果这样做，那么就能够对它们的性能进行很好的比较）。

17. 实现第 8.4.2 节在一个三角剖分上计算向量场的方法。对于给定的输入多边形，你的程序是否绘制了所计算的向量场的一个指针图。当目标发生变化时，确定能够多快重新计算向量场。

18. 最优导航函数问题：

（a）实现图 8.13 中所述的算法。显示最优尚需代价函数的水平集。

（b）将算法进行扩展并应用于X_{free}中有多个多边形孔的情况。

19. 对带插值的值迭代方法进行适应性修改，以使一个在平面上运动的点机器人能够跟踪一个被称为目标的一个可预测的动点。如果目标是不可视的，那么代价泛函应该加入一个小的罚值。对此进行最优化应该能够最小化目标不可视的时间。假设给定机器人的初始位形，请计算机器人的最优反馈规划。

20. 通过加入排斥状态离开障碍物、吸引状态向x_G的势函数，可经验性地构造导航函数。为简单起见，可以假设$X=\mathbb{R}^2$和障碍物是圆盘状的。从单个圆盘开始，然后逐步构造更加复杂的障碍区域。保证所产生的势函数在x_G以外没有局部最小值点，这是否很困难？

第III部分　决策论规划

存在不确定性条件下的规划

我们将第III部分称为决策论规划，也可称为存在不确定性下的规划。第 I 部分和第 II 部分中的所有概念都避开模型不确定性。第 8 章考虑能够克服某些不确定性的规划，但是没有显式建模不确定性。

在本部分中，不确定性通常影响规划的两个方面：

（1）可预测性：由于不确定性，当采取行动时不知道将来会发生什么。这意味着将来的状态不一定是可预测的。

（2）感测：由于不确定性，目前的状态不一定知道。关于状态的信息从初始状态、传感器以及对以前行动的记忆中产生。

这两种不确定性在许多方面是相互独立的。它们对规划问题产生不同的效果。

单个决策　在进行单个决策过程中通过提出表示不确定性的方法，第 9 章对第III部分的内容进行了介绍。该章的观点是：不确定性可以建模为来自于另外一个决策者的干扰。将引入一个被称为**大自然**(nature)的特殊决策者。任务是进行好的决策，不管大自然如何行动。可以采用**最坏情况**(worst case)模型或者**概率**(probabilistic)模型来刻画大自然的决策过程。一些规划问题可能涉及多个理性的决策者。这就导致了对策论，这是当局中人的目标有冲突时不能确定其他局中人将如何行动而产生的。第 9 章中所有概念都涉及进行单个决策；因此通常不需要状态空间，因为状态转移方程仅可使用一次。本章的目的是引入和仔细评估在不同形式的决策论中经常所做的假设。这将成为更加复杂问题的基础，特别是序贯决策和控制论。

预测中的不确定性　第 10 章采用第 9 章中的概念，并且在多个阶段上进行迭代。这产生状态和状态转移的概念，并且可以认为是第 2 章的离散规划概念和第 9 章的不确定性的混合体。本章也给出连续状态空间和连续时间的一些内容，这些内容是对第 II 部分的扩展。状态转移方程逐渐扩展到允许将来状态依赖于大自然所采取的未知行动。在对策论中，状态转移甚至可能依赖于多个决策者的行动。

对于第 10 章的所有模型，只存在预测中的不确定性；而目前的状态总是已知的。一个规划被定义为一个函数，它指示对任意目前状态所采取的行动。规划不能表述为一个行动序列，因为将来的状态是不可预测的，并且可能还要求在将来的状态出现时对它们产生响应。因此对一个固定的规划，每次的执行工作可能是不同的：采取的行动不同，到达的状态不同。规划一般是用最坏情况、期望情况或者均衡对策分析来计算。

感测中的不确定性：信息空间　第 11 章将介绍本书最重要的概念：信息空间。如果在感测目前的状态时存在不确定性，那么规划问题将很自然地处于一个信息空间中。可以

构造与位形空间和运动规划相似的内容。在使用位形空间概念来统一运动规划之前[591,660,854]，大多数算法是一种情况一种情况地进行研究的。例如当在世界范畴内进行定义时，机器人机械臂和可移动机器人有不同的特点。但是根据位形空间，很容易考虑诸如第5章和第6章的一般算法。

对涉及感测不确定性的规划问题也应该存在一种相似的统一（例如都不能确定目前状态）。目前，文献中的方法主要围绕单个模型和问题，就像基本运动规划那样。因此很难提供像第II部分那样的统一性。尽管如此，仍然可以用第11章的概念来统一介绍第12章中许多涉及感测不确定性的规划问题。就像在位形空间中的情况一样，这里需要认真学习信息空间概念，辛勤的耕耘一定会有丰厚的回报。

第12章将提出几个不同问题以及感测不确定性下的规划解。问题包括利用机器人探索新的环境、用摄像机进行追逃对策以及在没有或者很少感测的情况下对物体进行操纵。本章给出信息空间的许多有趣的应用，但是仍然会给读者留下这样的感觉：还有许多的事情等待解决。在信息空间中进行规划是具有挑战性的研究问题，它贯串于机器人学、控制论和人工智能的许多方面。

第 9 章　基本决策理论

本章作为建模和解决涉及多决策者的规划问题的基础。关注于在出现其他决策者可能影响结果的情况下所进行的单个决策。第 10 章～第 12 章的规划问题将被看作是决策问题序列。本章的思路可以被看作是进行一步规划。相对于第 2 章，本章将步数减少到一步，然后介绍更加完善的方法来对单步进行建模。在第 10 章则返回到多步情况，可以看到：第 2 章中的许多算法都可以很好地扩展来融合本章的决策论概念。

因为在阶段之间没有信息交流，所以不需要状态空间。本章我们是为决策者(DM)设计一个策略，而不是为机器人设计规划。在后续章节描述序贯决策时，规划这个术语将被再次使用。在本章它似乎是不适宜的，因为单个决策过程由于退化而不能被称为规划。

贯串整个第III部分的一个主题是多个决策者之间的相互影响。除了所指的机器人等基本决策者以外，将还有一个或者多个无法由机器人预测或控制的其他决策者。一个被称为大自然的特殊决策者将被当作一个通用方法来建模不确定性。大自然不是一个真实的实体来为其利益而进行智能的、理性的决策，从这个意义上来说，它是我们假想的。大自然的引入仅仅是作为一个方便的建模工具，通过它我们可以表达一些不同形式的不确定性。但是在某些情况下，决策者实际上可能是智能对手，它们为了各自的利益而进行决策。这就产生了对策论（game theory），其中所有的决策者（包括机器人）被称为局中人。

第 9.1 节提出了一些有助于理解和联系后续概念的基本的回顾和展望。第 9.2 节介绍了不确定性下的单个决策，这被称为决策论。第 9.3 节和第 9.4 节讨论了对策论，其中两个或者多个决策者同时进行决策，它们之间存在利益冲突。在第 9.3 节讨论的零和对策论中，两个决策者的利益决然相反。在第 9.4 节讨论的非零和对策论中，任何数量的决策者一起形成非合作对策，在它们之间任何冲突和竞争都是允许的。第 9.5 节用对本章所表述的一般模型的评论来结束本章内容。当试图将决策论模型应用于规划问题时，它是非常有用的。

本章的内容并不非常依赖于第II部分。实际上甚至不需要来自于第 2 章的概念，因为没有阶段或者状态空间。偶尔参考第II部分的内容，但是对理解来说这并不是至关重要的。本章关注的是离散空间。

9.1　基 本 概 念

9.1.1　最优化

9.1.1.1　单目标最优化

在进行复杂的决策模型之前，首先考虑单个决策者必须进行最好决策这种简单情况。这导致了一个我们所熟悉的最优化问题，它是这样表述的。

表述 9.1（最优化）

1. 一个被称为行动空间的非空集合 U。每个 $u \in U$ 都是一个行动。
2. 一个被称为代价函数的函数 $L: U \to \mathbb{R} \cup \{\infty\}$。

将表述 9.1 和表述 2.2 进行比较。这里不再需要状态空间 X 和状态转移的概念，因为仅有一个决策。因为没有状态空间，那么也就没有初始状态和目标状态。一个策略只是选择最好行动。

“最好”行动意味着什么？如果 U 是有限的，那么最好行动 $u^* \in U$ 是：

$$u^* = \arg\min_{u \in U} \{L(u)\} \tag{9.1}$$

如果 U 是无限的，那么存在不同情形。假设 $U = (-1,1)$ 和 $L(u) = u$。什么行动所产生的代价最低呢？我们愿意声明 -1 是最低的代价，但是 $-1 \notin U$。如果我们已经定义 $U = [-1,1]$，那么就可以这样进行了。但是如果 $U = (-1,1)$ 和 $L(u) = u$，那么没有行动产生最小代价。对于任意行动 $u \in U$，总可以选择另一个 $u' \in U$ 使得 $L(u') < L(u)$。但是如果 $U = (-1,1)$ 和 $L(u) = |u|$，那么式（9.1）正确地报告 $u = 0$ 是最好的行动。在这种情况下是没有问题的，因为最小值发生在内部，而不是在 U 的边界上。一般情况下，最优值可能不存在，意识到这一点是非常重要的。

有两种方法来解决这种令人沮丧的情况。一是要求 U 是一个有界闭集。因为闭集包括它的边界，所以该问题可以避免。有界条件避免了诸如优化 $U = \mathbb{R}$ 和 $L(u) = u$ 这样的问题。什么是最好的 $u \in U$？可以选择越来越小的 u 来产生一个更低的代价，即使 $\mathbb{R}$ 是一个闭集。

另外一种解决该问题的方法是定义和使用 inf 表示的下确界。它被定义为代价的最大下界。在 $U = (-1,1)$ 和 $L(u) = u$ 情况下，这是：

$$\inf_{u \in (0,1)} \{L(u)\} = -1 \tag{9.2}$$

但问题是不存在产生这种代价的行动 $u \in U$。下确界本质上是利用 U 的闭包来计算 $u \in U$。如果 U 正好是封闭的，那么 u 将包含在 U 中。这种方法也可以用于处理无界问题。$U = \mathbb{R}$ 和 $L(u) = u$ 情形下的下确界是 $-\infty$。

如果读者不能确信使用哪种方法，那么将 inf 写在将用 min 的地方是比较安全的。当最小值存在时，下确界恰好产生最小值，并且，当最小值不存在时，它给出一个合理的答案。但是在显然不需要 inf 的地方使用它确实令人为难（例如：在一个有限 U 的情况下）。

可通过乘以 -1 来产生最优化问题的相反版本。结果没有发生根本变化，但有时将一个问题表述为一个最大值可能比最小值更自然。例如这将在第 9.5.1 节效用理论的讨论中进行。在这种情况下，定义收益函数 R，而不是定义代价函数。任务是选择一个行动 $u \in U$ 使得收益最大化。通过对所有 $u \in U$ 设置 $L(u) = -R(u)$，最大值问题可以很方便地转化为最小值问题。对于最大值问题，下确界可以用上确界 sup 来代替，对于所有 $u \in U$，它是 $R(u)$ 上的最小上界。

对于本书中的大多数问题，在单个决策阶段中最优 $u \in U$ 的选择很简单；规划问题在一些其他方面却是非常复杂的。如果 U 和 L 很复杂，那么优化问题本身就极具挑战性，意识到这一点非常重要。例如 U 可能有界但非常大，或者 U 可能是 $\mathbb{R}^n$ 的一个高维（例如 1000）子集，并且代价函数可能很难或者不可能表示成一个简单封闭形式。如果函数足够简单，

那么可以应用基于一阶和二阶导数的标准微积分工具。但是在大多数实际应用中，还需要更加成熟的技术。许多技术涉及一种梯度下降，因此只能够保证得到局部最小值。在许多情况下都需要基于采样的技术。实际上第 5.2 节中诸如离散度这样的一些采样思路是在优化问题中发展起来的。对于某些类型的问题，可能存在组合解。例如线性规划涉及寻找线性函数族的最小值或者最大值，并且存在一些组合方法[261,266,667,734]。这种优化问题将出现在第 9.4 节。

如果给定在优化问题中基于采样的方法和组合方法的重要性，那么有两个与运动规划并行的有趣问题。第 5 章和第 6 章将分别讲述这两个问题。最优运动规划实际上对应于路径空间上的一个优化问题，它的特征很难进行刻画。在某些特殊情况下，诸如在第 6.2.4 节中，发现最优解是可能的，但在一般情况下很难发现最优解。变分法是在必须满足微分约束的一个路径空间上讨论优化问题的一个一般方法[844]，这将在第 13.4.1 中进行介绍。

9.1.1.2　多目标最优化

假设有一个代价函数的族，它们每个计算一个行动。这导致表述 9.1 扩展到多目标优化。

表述 9.2（多目标优化）

1. 一个被称为行动空间的非空集合 U。每个 $u \in U$ 都是一个行动。

2. 对于某整数 d，有一个向量代价函数 $L: U \rightarrow \mathbb{R}^d$。如果需要，任何代价分量都允许是 ∞。

第 7.7.2 节考虑的是这个问题的变化形式，在那里所涉及的是多个机器人的最优协调。如果 $L(u) = L(u')$，则两个行动 u 和 u' 被称为是等价的。如果对于所有使得 $1 \leqslant i \leqslant d$ 的 i，它们不等价且 $L_i(u) \leqslant L_i(u')$，则一个行动 u 被认为比 u' 占优。这在行动集合上定义了一个偏序 $\leqslant$。一些行动可能是不相容的。如果一个行动不被其他任何行动占优，则称该行动是 Pareto 最优的。这意味着它关于偏序是最小的。

例 9.1（Pareto 最优性的例子） 假设 $U = \{1,2,3,4,5\}$ 和 $d = 2$。代价赋值 $L(1) = (4,0)$，$L(2) = (3,3)$，$L(3) = (2,2)$，$L(4) = (5,7)$ 和 $L(5) = (9,0)$。可以删除行动 2、4 和 5，因为它们被其他行动占优。例如(3,3)被(2,2)占优；因此 $u = 3$ 比 $u = 2$ 可取。剩下的两个行动 $u = 1$ 和 $u = 3$ 是 Pareto 最优的。■

基于这个简单的例子，Pareto 最优性的概念似乎是针对抛弃被占优的行动。尽管可能有多个 Pareto 最优解，但是它至少将 U 缩减到最好可选行动的一个族上。

例 9.2（宾夕法尼亚州收费公路） 假设驾驶汽车通过宾夕法尼亚州公路时所面临的收费问题，这是一条高速公路，张贴有关于速度限制的警告标牌以及相应超速行驶的罚款。令 $U = \{50,51,\cdots,100\}$ 表示可能的整数速度，用每小时的英里数来表示(mph)。指示超速罚款的标牌是：(1) 驾驶速度在 56 mph～65 mph 的罚款是 50 美元；(2) 驾驶速度在 66 mph～75 mph 的罚款是 100 美元；(3) 驾驶速度在 76 mph～85 mph 的罚款是 200 美元；(4) 驾驶速度在 86 mph～100 mph 的罚款是 500 美元。如果速度超过 100 mph，那么处罚还包括拘留，这种处罚过于严厉，这里不予考虑。

对于一个驾驶员来说的两个指标是：(1) 经过该州的时间；(2) 罚款的费用。假设你违反速度限制，那么你将被抓住。目标是使这两个指标最小。那么 Pareto 最优驾驶速度是

多少？将 56 mph 和 57 mph 进行比较。它们所花费用相同，但是 57 mph 速度所用时间较少。因此 57 mph 比 56 mph 占优。实际上，65 mph 比所有小于 65 mph 而大于 56 mph 的速度占优，因为费用相同，并且降低时间。基于这种考虑，Pareto 最优驾驶速度是 55、65、75、85、100。单个驾驶员可以决定对他们来说最好的行动；但是任何超出 Pareto 最优集合以外的速度都是不明智的。■

下面的例子将阐述 Pareto 最优性的主要不足之处。仅删除非占优解可能没有用。在某些情况下，甚至可能有 Pareto 最优解的一个连续统。因此，Pareto 最优这个概念并不总是有用的。它的值依赖于特定的应用。

例 9.3（Pareto 最优解的连续统） 令 $U=[0,1]$ 和 $d=2$。令 $L(u)=(u,1-u)$。在这种情况下，U 的每个元素都是 Pareto 最优的。一个指标的微量减少会引起其他指标的增加，从这里可以看出这一点。因此任何两个行动是不可比较的。■

9.1.2 回顾概率论

本节将回顾概率论的一些基本概念，引入将在第III部分中使用的符号。

概率空间 一个概率空间是一个三元组 $(S,\mathcal{F},P)$，其中 3 个分量是：

（1）样本空间：一个非空集合 S 被称为样本空间，它表示所有可能的输出。

（2）事件空间：S 的子集的一个族 $\mathcal{F}$ 被称为事件空间。如果 S 是离散的，那么 $\mathcal{F}=\text{pow}(S)$。如果 S 是连续的，那么 $\mathcal{F}$ 通常是 S 上的一个 sigma 代数，如第 5.1.3 节定义。

（3）概率函数：一个函数 $P:\mathcal{F}\to\mathbb{R}$ 给出 $\mathcal{F}$ 中事件的概率。这有时称为 S 上的概率分布。

概率函数 P 必须满足几个基本性质：

（1）对所有 $E\in\mathcal{F}$，$P(E)\geqslant 0$。

（2）$P(S)=1$。

（3）对于所有 $E,F\in\mathcal{F}$，如果 $E\cap F=\phi$，那么 $P(E\cup F)=P(E)+P(F)$。

如果 S 是离散的，那么在所有 $\mathcal{F}$ 上 P 的定义可以从它在 S 的单个元素上的定义通过性质进行推导。在这种情况下，对某 $s\in S$ 记为 $P(s)$ 是很常见的，这种表示稍微有一点不合适，因为 s 不是一个事件。对某 $\{s\}\in\mathcal{F}$，它应该是 $P(\{s\})$。

例 9.4（掷骰子） 考虑一个正六面体或者骰子，它的每个面都有一个 1～6 的数字。当骰子停下来时，它显示一个数字。在这种情况下，$S=\{1,2,3,4,5,6\}$ 是样本空间。事件空间是 $\text{pow}(S)$，这是 S 的 2^6 个子集。假设概率函数表示所有数字是等可能性的。对于任何 $s\in S$，$P(\{s\})=1/6$。事件包括所有子集使得可以表述任何概率命题。例如得到偶数的概率是多少？事件 $E=\{2,4,6\}$ 发生的概率为 $P(E)=1/2$。■

第三个概率性质看起来像第 5.1.3 节中度量空间定义中最后一个性质。实际上 P 就是例 5.12 所提到的一个特殊类型的度量空间。但是如果 S 是连续的，那么这种度量不能由在单元素集合上所定义的概率来确定。单元素集合的概率通常是零。概率密度函数 $p:S\to\mathbb{R}$ 用来定义概率度量。任何事件 $E\in\mathcal{F}$ 的概率函数 P 可以通过积分来确定：

$$P(E)=\int_E p(x)\mathrm{d}x \tag{9.3}$$

其中 $x \in E$ 是积分变量。从直觉上看，P 表示 E 上所积累的总的概率。

条件概率　对于任意两个事件 $E, F \in \mathcal{F}$，条件概率表示为 $P(E \mid F)$，叫做“给定 F 的情况下，E 的概率”。它的定义是：

$$P(E \mid F) = \frac{P(E \cap F)}{P(F)} \tag{9.4}$$

两个事件 E 和 F 称为独立的，当且仅当 $P(E \cap F) = P(E)P(F)$；否则称它们是相关的。一个重要的而且容易误解的概念是条件独立。考虑某第三个事件 $G \in \mathcal{F}$。可能存在这种情况：E 和 F 是相关的，但是当给定 G，它们就变成独立的。因此 $P(E \cap F) \neq P(E)P(F)$；但是 $P(E \cap F \mid G) = P(E \mid G)P(F \mid G)$。这种例子在实际中经常出现。例如 E 可能用来表示人的身高，F 是他们的阅读水平。这些通常是相关的事件，因为儿童通常比较矮并且阅读水平低。如果给定人的年龄作为事件 G，那么身高不再重要。从直觉上可以意识到，一旦给定年龄，那么身高和阅读水平之间应该没有联系。

对条件概率的定义式（9.4）施加约束：

$$P(E \cap F) = P(F)P(E \mid F) = P(E)P(F \mid E) \tag{9.5}$$

它将 $P(E \mid F)$ 和 $P(F \mid E)$ 很好地联系起来。这产生贝叶斯公式，它是一种非常方便的交换 E 和 F 的方法：

$$P(F \mid E) = \frac{P(E \mid F)P(F)}{P(E)} \tag{9.6}$$

概率分布 $P(F)$ 被称为先验分布，$P(F \mid E)$ 是后验分布。这些术语分别表示概率出现在考虑 E 之前和之后。

如果所有概率都是以某事件 $G \in \mathcal{F}$ 为条件的，那么就出现了条件贝叶斯公式，它与式（9.6）仅有的不同之处在于将条件 G 放入概率：

$$P(F \mid E, G) = \frac{P(E \mid F, G)P(F \mid G)}{P(E \mid G)} \tag{9.7}$$

边缘化　令事件 $F_1, F_2, \cdots, F_n$ 为 S 的任意划分。通过边缘化，可以得到一个事件 E 的概率：

$$P(E) = \sum_{i=1}^{n} P(E \mid F_i)P(F_i) \tag{9.8}$$

它的最重要的应用是在贝叶斯公式的分母中，将式（9.8）代入式（9.6）中的分母就产生：

$$P(F \mid E) = \frac{P(E \mid F)P(F)}{\sum_{i=1}^{n} P(E \mid F_i)P(F_i)} \tag{9.9}$$

这种形式有时很实用，因为 $P(E)$ 将被消除掉。

随机变量　假设给定概率空间 (S, F, P)。一个随机变量[1] X 是将 S 映射到 $\mathbb{R}$ 的一个函数。因此，X 给样本空间的每个元素赋一个实数值。这很方便计算概率空间上的统计。如果 S 是 $\mathbb{R}$ 的一个子集，那么 X 在默认情况下表示恒等函数。

1　这个名称经常容易引起混淆。随机变量不是“随机的”，也不是一个“变量”。它是一个函数 $X : S \to \mathbb{R}$。它通常用一个大写字母来表示，而小写字母通常用来表示函数。

期望 随机变量 X 的期望或者期望值用 $E[X]$ 表示。它可以被认为是对 X 的一种加权平均，其中权值从概率分布得到。如果 S 是离散的，那么：

$$E[X]=\sum_{s\in S}X(s)P(s) \tag{9.10}$$

如果 S 是连续的，那么[1]：

$$E[X]=\int_S X(s)p(s)\mathrm{d}s \tag{9.11}$$

还可以定义条件期望，它对概率分布加入一个给定条件。例如，如果 S 是离散的且给定事件 F，那么：

$$E[X\mid F]=\sum_{s\in S}X(s)P(s\mid F) \tag{9.12}$$

例 9.5（掷骰子） 回到例 9.4 中，S 的元素已经是实数。因此随机变量 X 可以用 $X(s)=s$ 来定义。通过使用式（9.11），期望值 $E(X)$ 为 3.5。实际中期望值不一定是一个“被期望”的值。实际上不可能得到值 3.5，即使它不包含在 S 中。假设仅在导致大于 3 的试验中希望得到 X 的期望值。这可以用事件 $F=\{4,5,6\}$ 来描述。采用条件期望式（9.12），那么期望值 $E[X\mid F]=5$。

现在考虑连续掷两个骰子。每个元素 $s\in S$ 表示为 $s=(i,j)$，其中 $i,j\in\{1,2,3,4,5,6\}$。因为 $S\not\subset\mathbb{R}$，所以需要随机变量更加容易引起关注。一个通常的方法是计算骰子的总和，对任意 $s\in S$，它产生 $X(s)=i+j$。在这种情况下，$E[X]=7$。 ■

9.1.3 随机策略

直到现在，在规划中所采取的行动都是确定性的。第 2 章的规划指定完全确定的行动。表述 9.1 可用最好的行动来解决。可将它看作是一个策略，只不过它每次所做的决策都相同。

在某些应用中，决策者可能不希望自己是可预测的。为了实现这种愿望，策略中可以融入随机化。如果 U 是离散的，可用 U 上的概率分布 $P(u)$来指定一个随机策略 w。令 W 表示所有可能的随机策略的集合。当应用策略时，按照概率分布 $P(u)$ 进行采样来选择行动 $u\in U$。我们必须明确地区分“定义策略”和“应用策略”之间的区别。目前它们一直被认为是等价的；但是必须执行随机策略来确定所产生的行动。如果重复执行策略，那么假设每次试验都与以前试验中所得到的行动无关。那就是说，$P(u_k\mid u_i)=P(u_k)$，其中 $P(u_k\mid u_i)$ 表示在给定 u_i 的情况下，在第 k 次试验中策略选择行动 u_k 的概率，u_i 表示在第 i 次试验中所选择的行动，$i<k$。如果 U 是连续的，那么随机策略可以用概率密度函数 $p(u)$来指定。在决策论和对策论的文献中，确定性策略和随机策略通常分别被称为单纯的和混合的。

例 9.6（基于投掷硬币的决策） 令 $U=\{a,b\}$。一个随机策略 w 可以定义为：

（1）投掷一枚硬币，它有两种可能的结果：正面(H)或者反面(T)。

（2）如果结果为 H，那么选择 a；否则，选择 b。

1 用测度论中语言，两个定义只是 Lebesgue 积分的特殊情况。测度论将离散概率论和连续概率论很好地统一起来，因此不需要明确指出这两种情况，参见文献[348,548,838]。

因为投掷硬币是公平的，所以 w 用 $P(a)=P(b)=1/2$ 来定义。每次运用策略时并不知道将选择什么行动。但是许多次试验后，它有一半的次数选择 a。 ■

如果你不被零概率事件所困扰，那么一个确定性的策略总是可以认为是一个特殊情形的随机策略。一个确定性的策略 $u_i \in U$ 可以用随机策略来模拟，只是需要指定概率 $P(u)=1$（如果 $u=u_i$）；否则 $P(u)=0$。选择不同行动的概率仅为零。

假设使用随机策略来解决表述 9.1 所表达的问题。首先出现的困难是不能预测代价。如果多次应用策略，那么我们可以定义平均代价。当次数趋近无穷时，该平均将收敛到期望的代价 $\bar{L}(w)$，如果 L 被当作一个随机变量（除可以被当作代价函数以外）。如果 U 是离散的，那么随机策略 w 的期望代价为：

$$\bar{L}(w)=\sum_{u\in U} L(u)P(u)=\sum_{u\in U} L(u)w \tag{9.13}$$

一个有趣的问题是对于所有 $u\in U$，是否存在某 $w\in W$ 使得 $\bar{L}(w)<L(u)$。也就是说，是否存在比用表述 9.1 表示的所有确定性策略都更好的随机策略？回答是否定的，因为最好的策略给能够最小化 L 的行动 u^* 分配概率 1。这等价于采用确定性策略。如果两个或者两个以上行动都能得到最优代价，那么在它们之间随机策略可以任意分配概率。但是代价不会有进一步的降低。因此在这种情况下，随机选择似乎是无意义的，除非有其他考虑。

能够显示随机策略非常重要的一个例子是在与一个智能对手的竞争中所进行的决策。如果问题重复多次，那么对手能够学到任意确定性策略。那么可以用随机化来弱化对手的预测能力。这个思路将在第 9.3 节用来产生更好的方法来进行零和对策。

下面的例子阐述了当与一个智能对手进行对策时随机选择的优势。

例 9.7（投掷硬币的比赛） 考虑的对策涉及两个局中人，它们重复进行的对策是将硬币放在桌子上。在每次试验中，局中人必须同时将硬币正面朝上(H)或者反面朝上(T)。令两个字母的串表示输出结果。如果输出是 HH 或者 TT（局中人选择相同的朝向），那么局中人 1 付给局中人 2 一个比索；如果输出是 HT 或者 TH，那么局中人 2 付给局中人 1 一个比索。如果局中人 1 采用确定性策略，那么会如何？如果局中人 2 能够确定策略，那么他所选择的策略总能够使他赢。但是如果局中人 1 采用最好的随机策略，那么局中人 2 最多只能希望突破平均数。什么随机策略能够实现呢？

将该问题推广到 3 个行动就成为著名的 Rock-Paper-Scissors 对策[958]。如果你想设计一个与精明的对手重复进行这种对策的计算机程序，那么最好融入随机性。 ■

9.2　与大自然之间的对策

9.2.1　对大自然的建模

本书到这里才第一次对不确定性进行直接建模。两个决策者是：

- 机器人：在本书中主决策者的名称都是机器人。到目前为止都只有一个决策者。现在将有两个决策者，那么名称就更加重要了，因为需要用名称来区分不同的决策者。
- 大自然：这个决策者对机器人来说是一种不可预测的神秘力量。它有它自己的行

动，它选择的行动将干扰机器人的运行。大自然可以被认为是一个综合的决策者，在决策中或者规划过程中构造它的目的是为了建模不确定性。

假设机器人和大自然各自进行决策。每个决策者都有一个行动集合可供选择。假设代价取决于各自所选择的行动。代价仍然表示机器人的输出的效果；但是机器人必须考虑大自然对代价的影响。因为大自然是不可预测的，机器人必须建立大自然行为模型。像以前一样，假设机器人的行动集合 U。现在假设大自然也有一个行动集合。该集合称为大自然行动空间，用 Θ 表示。大自然的行动表示为 $\theta \in \Theta$。那么现在称 U 为机器人行动空间是合适的；但是为方便起见，它通常被简称为行动空间，暗含有机器人的意思。

这导致如下表述，它是对表述 9.1 的扩展。

表述 9.3（与大自然的对策）

1. 一个被称为（机器人）行动空间的非空集合 U。每个 $u \in U$ 都是一个行动。
2. 一个被称为大自然行动空间的非空集合 Θ。每个 $\theta \in \Theta$ 都是一个大自然行动。
3. 一个被称为代价函数的函数 $L: U \times \Theta \to \mathbb{R} \cup \{\infty\}$。

代价函数 L 现在依赖于 $u \in U$ 和 $\theta \in \Theta$。如果 U 和 Θ 是有限的，那么将 L 指定为 $|U| \times |\Theta|$ 矩阵是很方便的，它被称为代价矩阵。

例 9.8（与大自然的简单对策） 假设 U 和 Θ 都包含 3 个行动。这就产生了 9 个可能的输出，它们可以用如下代价矩阵来指定：

U \ Θ			
	1	−1	0
	−1	2	−2
	2	−1	1

机器人行动 $u \in U$ 选择一行，大自然行动 $\theta \in \Theta$ 选择一列。所产生的代价 $L(u, \theta)$ 由相应的矩阵项给出。 ■

在表述 9.3 中，两个决策者同时行动；大自然在进行决策之前不知道机器人的行动。而在许多情形下，大自然可能知道机器人的行动。在这种情况下，可以为每个 $u \in U$ 定义一个不同的大自然行动空间。这得到的是对表述 9.3 的推广。

表述 9.4（大自然知道机器人的行动）

1. 一个被称为行动空间的非空集合 U。每个 $u \in U$ 都是一个行动。
2. 对于每个 $u \in U$，一个被称为大自然行动空间的非空集合 $\Theta(u)$。
3. 一个被称为代价函数的函数 $L: U \times \Theta \to \mathbb{R} \cup \{\infty\}$。

如果机器人选择行动 $u \in U$，那么大自然选择 $\Theta(u)$。

9.2.2 非确定模型和概率模型

在与大自然的博弈中，对机器人来说，什么是最好的决策？这取决于机器人所拥有的关于大自然如何选择行动的信息。通常假设机器人不准确知道大自然所选择的行动；否则定义大自然就没有意义了。机器人对大自然进行建模可以用两种模型。从机器人的角度，可能的模型是：

- 非确定的：我不知道大自然将做什么。
- 概率的：我已经观测并收集到大自然的统计信息。

在这两种模型下，假设机器人知道表述 9.3 中的 Θ 或者表述 9.4 中对所有 $u \in U$ 的 $\Theta(u)$。非确定的和概率的这两个术语是从文献[315]借鉴而来的。在某些文献中，“可能的”通常用来代替“非确定的”。这是一个非常有用的概念，但是在英语中它与“概率的”太相似。

首先假设使用表述 9.3 且 U 和 Θ 是有限的。在非确定的模型下，没有多余的信息。在这种情况下，一个合理的方法是通过假设最坏情况来进行决策。甚至可以假设大自然知道机器人所采取的行动，它恶意地选择行动使得代价尽可能高。这种悲观的观点有时被称为 Murphy 原理[113]或者 Sod 原理。在这种情况下，最好的行动 $u^* \in U$ 选择为：

$$u^* = \operatorname*{arg\,min}_{u \in U} \left\{ \max_{\theta \in \Theta} \{ L(u, \theta) \} \right\} \tag{9.14}$$

行动 u^* 是用最坏情况分析所得到的最低代价。这有时被称为最小最大解，因为式（9.14）中有 min 和 max。如果 U 或者 Θ 是无限的，那么 min 或 max 可能不存在，而应该用 inf 或者 sup 来代替。

在某些应用中，最坏情况分析似乎过于悲观。更合理的是假设 Θ 中的所有行动是等可能性的。这可以作为概率模型的一个特殊情况来处理，下面将要进行描述。

在概率模型下，假设机器人已经收集了足够的信息来可靠地估计 $P(\theta)$（如果 Θ 是连续的，那么用 $p(\theta)$）。在这种情况下，假设大自然采用第 9.1.3 节所定义的随机策略。假设大自然所用的行动已经在许多次试验中得到观测，且在将来它们继续用相同的方式选择行动，就像通过分布 $P(\theta)$ 所预测的那样。不用最坏情况分析，而是采用期望分析(expected-case analysis)。这将最优化多次独立试验中所得到的平均代价。在这种情况下，最好行动 $u^* \in U$ 是：

$$u^* = \operatorname*{arg\,min}_{u \in U} \left\{ E_\theta \left[L(u, \theta) \right] \right\} \tag{9.15}$$

其中 E_θ 指的是根据 θ 上的概率分布（或者密度）所得到的期望。因为 Θ 和 $P(\theta)$ 一起形成了一个概率空间，$L(u, \theta)$ 可以被认为是对每个 u 的一个随机变量（它给样本空间的每个元素指定了一个实数值）[1]。采用 $P(\theta)$，式（9.15）中的期望可以表达为：

$$E_\theta \left[L(u, \theta) \right] = \sum_{\theta \in \Theta} L(u, \theta) P(\theta) \tag{9.16}$$

例 9.9（非确定模型与概率模型） 回到例 9.8 中。令 $U = \{u_1, u_2, u_3\}$ 表示机器人的行动，令 $\Theta = \{\theta_1, \theta_2, \theta_3\}$ 表示大自然的行动。

在大自然的非确定模型下，$u^* = u_1$，它使用式（9.14）产生最坏情况下的 $L(u^*, \theta) = 1$。在概率模型下，令 $P(\theta_1) = 1/5$，$P(\theta_2) = 1/5$，$P(\theta_3) = 3/5$。为了发现最优行动，可以采用式（9.15）。这涉及计算每个行动的期望代价：

$$\begin{aligned} E_\theta \left[L(u_1, \theta) \right] &= (1)1/5 + (-1)1/5 + (0)3/5 = 0 \\ E_\theta \left[L(u_2, \theta) \right] &= (-1)1/5 + (2)1/5 + (-2)3/5 = -1 \\ E_\theta \left[L(u_3, \theta) \right] &= (2)1/5 + (-1)1/5 + (1)3/5 = 4/5 \end{aligned} \tag{9.17}$$

最好的行动是 $u^* = u_2$，它产生最低期望代价 -1。

1　可选地，一个随机变量可能被定义在 $U \times \Theta$ 上，并且应该采用条件期望，其中 u 是给定的。

如果概率分布为 $P=[1/10\ \ 4/5\ \ 1/10]$，那么将得到 $u^*=u_1$。因此最好决策依赖于 $P(\theta)$；如果信息是统计有效的，那么它可以产生更精明的决策。如果这些信息不可用，那么采用非确定模型是更合适的。

但是在缺乏数据的情况下，可以将 $P(\theta)$ 作为均匀分布。这意味着大自然的所有行动是等可能性的；但是基于这种假设所得到的结果是危险的，参见第 9.5 节。 ■

在表述 9.4 中，大自然的行动空间 $\Theta(u)$ 依赖于机器人的行动 $u\in U$。在非确定模型下，式（9.14）变成：

$$u^*=\arg\min_{u\in U}\left\{\max_{\theta\in\Theta(u)}\{L(u,\theta)\}\right\} \tag{9.18}$$

非常遗憾，这些问题没有很好的矩阵表示，因为 $\Theta(u)$ 的大小随着 $u\in U$ 的不同而发生变化。在概率情况下，$P(\theta)$ 被条件概率分布 $P(\theta|u)$ 所代替。对这种分布进行估计要求对每个可能的 $u\in U$ 进行大量的独立试验。大自然的行为现在依赖于机器人的行动；但是仍然用随机策略来刻画大自然。在多次试验中它并不会适应性地修改它的策略。式（9.16）中的期望现在变成：

$$E_\theta[L(u,\theta)]=\sum_{\theta\in\Theta(u)}L(u,\theta)P(\theta|u) \tag{9.19}$$

其中用 $P(\theta|u)$ 代替了 $P(\theta)$。

遗憾（regret） 这里提出的模型并不是绝无仅有的，还有许多可接受的方法能够产生好的决策。在对策论中，关键的思路是使遗憾最小化。当你进行一次不理想的决策后希望对策结束后能够改变它，这就是你此时的感觉。假设选择某 $u\in U$ 后，得知大自然采取了某 $\theta\in\Theta$。在给定大自然所采取的行动的情况下，采取行动的代价比不知道大自然行动时所付出代价少的那部分代价就是所谓的遗憾。

对于 $u\in U$ 和 $\theta\in\Theta$ 每种组合，遗憾 T 被定义为：

$$T(u,\theta)=\max_{u'\in U}\{L(u,\theta)-L(u',\theta)\} \tag{9.20}$$

对于表述 9.3，如果 U 和 Θ 是有限的，那么可以定义 $|\Theta|\times|U|$ 的遗憾矩阵。

假设最小化遗憾是基本要求，这与实际付出的代价不同。在非确定模型下，能够最小化最坏情况下的遗憾的行动是：

$$u^*=\arg\min_{u\in U}\left\{\max_{\theta\in\Theta}\{T(u,\theta)\}\right\} \tag{9.21}$$

在概率模型中，最小化期望遗憾的行动是：

$$u^*=\arg\min_{u\in U}\{E_\theta[T(u,\theta)]\} \tag{9.22}$$

它们与式（9.14）和式（9.15）的不同之处在于 L 已经被 T 所代替。在第 9.3.2 节中，我们将会更详细地讨论遗憾，因为它形成了对策论中最优化概念的基础。

例 9.10（遗憾矩阵） 例 9.8 中的遗憾矩阵为：

		Θ	
U	2	0	2
	0	3	0
	3	0	3

用非确定模型，那么 $u^*=u_1$，它用式（9.21）产生最坏情况下的遗憾是 2。在概率模型下，令 $P(\theta_1)=P(\theta_2)=P(\theta_3)=1/3$。在这种情况下，$u^*=u_1$ 或者 $u^*=u_2$，因为它们用式（9.22）产生的期望遗憾都是 1。

9.2.3　利用观测

表述 9.3 和表述 9.4 不允许机器人在进行决策之前接收任何信息（除了 L 以外）。现在假设机器人有一个传感器，它在选择最好行动之前能够进行检测。这种传感器提供的观测或者测量包含关于大自然可能选择的行动的信息。在某些情况下，可将大自然行动假设为一种已经选择的状态。那么观测量将提供关于这种状态的信息。例如大自然可能选择曼谷目前的气温。那么观测可以是曼谷的一只用来读取温度的温度计。

表述问题　令 Y 表示观测空间，它是所有可能的观测 $y\in Y$ 的集合。为方便起见，假设 Y、U 和 Θ 都是离散的。假设给定 y，那么作为模型一部分的某些对 θ 的约束也是已知的。在非确定模型下，为每个 $\theta\in\Theta$ 指定一个集合 $Y(\theta)\subseteq Y$。给定大自然的行动 θ，该集合 $Y(\theta)$ 指示可能的观测。在概率模型下指定条件概率分布 $P(y\mid\theta)$。感测模型将在第 9.2.4 节给出。一些其他的模型将出现在第 11.1.1 节和第 11.5.1 节，尽管它们是用状态空间 X 表示的，在这里状态空间为 Θ。像以前一样，概率情形也要求给定先验分布 $P(\Theta)$。这产生如下表述：

表述 9.5（在有观测的情况下与大自然之间的对策）

1. 一个被称为行动空间的有限非空集合 U。每个 $u\in U$ 都是一个行动。
2. 一个被称为大自然行动空间的有限非空集合 Θ。
3. 一个被称为观测空间的有限非空集合 Y。
4. 为每个 $\theta\in\Theta$ 指定一个集合 $Y(\theta)\subset Y$ 或者概率分布 $P(y\mid\theta)$。如果 θ 是大自然的行动，那么这分别指示哪个观测是可能的。在概率情况下也必须指定先验概率 $P(\theta)$。
5. 一个被称为代价函数的函数 $L:U\times\Theta\to\mathbb{R}\cup\{\infty\}$。

考虑解决表述 9.5 中的问题。现在的策略比简单地指定一个行动更加复杂，因为我们想在得到观测之前完整地刻画机器人的行为。这可以通过将一个策略定义为一个函数 $\pi:Y\to U$ 来实现。对于每个可能的观测 $y\in Y$，策略提供了一个行动。我们现在想通过搜索策略空间来发现在所有可能观测上产生最好决策的策略。本节中 Y 实际上是信息空间的一个特殊情况，信息空间是第 11 章和第 12 章的主要内容。最后，一个策略（或者规划）将以一个信息状态为前提，信息状态是对观测的推广。

最优策略　首先考虑在非确定模型下寻找最优策略，用 π^* 表示。必须用集合 $Y(\theta)$ 来确定对于每个观测 $y\in Y$，大自然可能采取的是哪个行动。令 $\Theta(y)$ 表示该行动，它为：

$$\Theta(y)=\{\theta\in\Theta\mid y\in Y(\theta)\} \tag{9.23}$$

定义最优策略 π^*，通过为每个 $y\in Y$ 设置：

$$\pi^*(y)=\operatorname*{arg\,min}_{u\in U}\left\{\max_{\theta\in\Theta(y)}\{L(u,\theta)\}\right\} \tag{9.24}$$

将式（9.24）与式（9.14）进行比较，这里最大值是在所有 Θ 上取得的。拥有观测 y 的好处是集合被限定在 $\Theta(y)\subseteq\Theta$。

在概率模型下，必须进行一个与式（9.23）相似的运算。这涉及从 $P(y|\theta)$ 计算 $P(\theta|y)$ 来确定 y 所包含的关于 θ 的信息。采用分母边缘化的贝叶斯公式（9.9），结果是：

$$P(\theta|y)=\frac{P(y|\theta)P(\theta)}{\sum_{\theta\in\Theta}P(y|\theta)P(\theta)} \tag{9.25}$$

为了明确非确定模型和概率模型之间的联系，应定义一个概率分布 $P(y|\theta)$，它仅在 $y\in Y(\theta)$ 的情况下才是非零的，并且 $P(\theta)$ 采用的是均匀分布。在这种情况下，式（9.25）给式（9.23）$\Theta(y)$ 的元素指定非零概率。因此式（9.25）仅仅是式（9.23）的概率版本。对每个 $y\in Y$，指定最优策略 π^* 为：

$$\pi^*(y)=\operatorname*{arg\,min}_{u\in U}\{E_\theta[L(u,\theta)|y]\}=\operatorname*{arg\,min}_{u\in U}\left\{\sum_{\theta\in\Theta}L(u,\theta)P(\theta|y)\right\} \tag{9.26}$$

式（9.26）与式（9.15）和式（9.16）的不同之处在于，用 $P(\theta|y)$ 代替了 $P(\theta)$。对于每个 u，式（9.26）中的期望被称为条件贝叶斯风险。所选择的最优策略 π^* 总是最小化该风险。注意到式（9.26）中的 $P(\theta|y)$ 可以用式（9.25）来表达，其分母表示 $P(y)$ 且不依赖于 u；因此它不影响最优性。由此，在式（9.26）中 $P(\theta|y)$ 的位置上可以使用 $P(y|\theta)P(\theta)$，并且能够得到相同的 π^*。如果空间是连续的，那么在所有概率分布的位置上采用概率密度，而方法是相同的。

大自然行动两次 允许大自然行动两次可以给出另外一个方便的表达形式：

1. 首先选择大自然的行动 $\theta\in\Theta$，但是该行动对机器人来说是未知的。
2. 其次选择大自然观测行动来干扰机器人对 θ 的感测能力。

令 ψ 表示大自然观测行动，它选自于大自然观测行动空间 $\Psi(\theta)$。现在可以定义传感器映射 h，它对每个 $\theta\in\Theta$ 和 $\psi\in\Psi(\theta)$ 产生 $y=h(\theta,\psi)$。因此对两种类型的大自然行动 $\theta\in\Theta$ 和 $\psi\in\Psi$，给定观测 $y=h(\theta,\psi)$。这产生了对表述 9.5 的另外一种表达。

表述 9.6（大自然干扰观测）

1. 一个被称为行动空间的有限非空集合 U。
2. 一个被称为大自然行动空间的有限非空集合 Θ。
3. 一个被称为观测空间的有限非空集合 Y。
4. 对每个 $\theta\in\Theta$，一个被称为大自然观测行动空间的非空集合 $\Psi(\theta)$。
5. 一个传感器映射 $h:\Theta\times\Psi\to Y$。
6. 一个被称为代价函数的函数 $L:U\times\Theta\to\mathbb{R}\cup\{\infty\}$。

该表述用一个函数 h 将非确定模型和概率模型很好地进行了统一。为了表达非确定模型，假设任意 $\psi\in\Psi(\theta)$ 都是可能的。采用 h，那么：

$$\Theta(y)=\{\theta\in\Theta|\exists\psi\in\Psi(\theta),y=h(\theta,\psi)\} \tag{9.27}$$

对于概率模型，需要指定一个分布 $P(\psi|\theta)$（通常它可以变为 $P(\psi)$）。假设当 h 的定义域限制在某 $\theta\in\Theta$ 时，那么它形成了一个从 Ψ 到 Y 的单射。也就是说，当假设 θ 确定时，每个大自然观测行动产生唯一一个观测。采用 $P(\psi)$ 和 h，产生 $P(y|\theta)$：

$$P(y|\theta)=\begin{cases}P(\psi|\theta) & (\text{唯一}\psi\text{ 使得}y=h(\theta,\psi))\\ 0 & \text{其他情况}\end{cases} \tag{9.28}$$

如果取消单射假设，那么 $P(\psi|\theta)$ 用所有 ψ 上的总和来代替，其中 ψ 满足 $y=h(\theta,\psi)$。

在表述 9.6 中，非确定模型和概率模型之间的不同之处在于 ψ 的特征，它表示一种测量干扰。策略仍然采用 $\pi:\Theta\to U$ 的形式。甚至可能采用一种混合模型，其中大自然行动用非确定模型建模，其他的用概率模型建模。

多次观测　表述 9.5 的另外一种扩展是在进行决策之前允许多次观测 $y_1,y_2,\cdots,y_n$。假设每个 y_i 都属于一个观测空间 Y_i。策略 π 现在就依赖于所有的观测：

$$\pi:Y_1\times Y_2\times\cdots\times Y_n\to U \tag{9.29}$$

在非确定模型下，为每个 i 和 $\theta\in\Theta$ 指定 $Y_i(\theta)$。式（9.24）中的 $\Theta(y)$ 用下式代替：

$$\Theta(y_1)\cap\Theta(y_2)\cap\cdots\cap\Theta(y_n) \tag{9.30}$$

来得到最优行动 $\pi^*(y_1,\cdots,y_n)$。

在概率模型下，需要指定 $P(y_i\mid\theta)$。给定 θ 时，通常假设观测是条件独立的。这意味着对于 $i\neq j$ 的任意 y_i、θ 和 y_j，有 $P(y_i\mid\theta,y_j)=P(y_i\mid\theta)$。式（9.26）中的条件 $P(\theta\mid y)$ 用 $P(\theta\mid y_1,\cdots,y_n)$ 代替。应用贝叶斯公式、用给定 θ 情况下 y_i 的条件独立性，可以产生：

$$P(\theta\mid y_1,\cdots,y_n)=\frac{P(y_1\mid\theta)P(y_2\mid\theta)\cdots P(y_n\mid\theta)P(\theta)}{P(y_1,\cdots,y_n)} \tag{9.31}$$

分母可以作为一个不影响最优化的常数来对待。因此不需要计算它，除非最优行动以外还需要最优期望代价。

条件独立避免指定整个 $P(y\mid\theta)$，这种简化是非常显著的。有时当条件独立假设不正确时使用它只是为了利用这种简化。因此使用观测的条件独立性的方法通常被称为朴素(naive)贝叶斯方法。

9.2.4　最优决策的例子

目前的框架所描述的是统计决策论（statistical decision theory），它涵盖了广泛的应用范围和研究问题。实事上在决策必须由机器或人在遵循指定原则情况下自动进行的任何情况下都会用到这些决策论概念。在第 10 章～第 12 章，这种决策问题将被嵌入到复杂的规划问题中。规划将作为一个序贯决策过程迭代地修改状态空间中的状态。通常每个决策步骤比在决策论的普通应用中更简单些。这是因为规划问题被许多因素复杂化。如果在特定的应用中的决策步骤过于复杂而难以解决，那么即使将它扩展到规划，问题也是没有希望得到解决的。

尽管如此，认识到在本节内容的框架下进行建模和解决决策问题时所出现的挑战是非常重要的。这里提出一些例子来帮助阐述它的作用和应用范围。

9.2.4.1　模式分类

过去几十年在计算机视觉和机器学习中一个活跃的领域就是模式分类[273,297,714]。问题涉及使用一个数据集合来实现分类。例如在计算机视觉中，数据对应于从图像中获取的信息。这些信息代表所观测到的物体的特征，视觉系统使用这些特征来对物体进行分类。

这里介绍的内容是对模式分类的一个高度理想化。我们假设所有合适的模型都是可用的，其中包括所需要的概率分布。在某些情况下，这都可以通过在大量数据集合上进行统计得到。但是在一些应用中这些数据是很难得到的或得不到的，所以与其花费昂贵代价得

到好的信息，不如研究更好的分类技术。有些问题是无人操作的，这意味着分类必须自动进行。由于类似原因，模式分类依然是具有挑战性的研究领域。

一般情况是大自然首先确定类型，其次得到关于类型的观测，最后机器人试图根据观测来猜测类型。问题采用表述 9.5。令 Θ 表示类型的一个有限集合。因为机器人必须猜测类型 $U=\Theta$。定义一个简单的代价函数来测量 u 和 θ 之间的不匹配：

$$L(u,\theta)=\begin{cases}0 & (u=\theta)\\ 1 & (u\neq\theta)\end{cases} \tag{9.32}$$

其中 $u=\theta$ 表示正确分类，$u\neq\theta$ 表示不正确分类。如果可以用行动 u 来确定类型误差，那么非确定模型产生的代价为 1。在概率模型下，式（9.32）的期望给出在给定一个行动 u 的情况下分类误差的概率。

接下来考虑用来进行分类决策的信息。令 Y 表示特征空间，其中每个 $y\in Y$ 被称为一个特征或者特征向量（通常 $y\in\mathbb{R}^n$）。这种情况下的特征只是表述 9.5 所给出的一个观测。最好的分类器或者分类原则就是在最坏情况下或者期望情况下产生最小分类误差的一个策略 $\pi:Y\to U$。

贝叶斯分类器 在模式分类中，概率方法是最常见的。这就产生了贝叶斯分类器。这里假设给定 $P(y\mid\theta)$ 和 $P(\theta)$。对给定的类型来说特征的分布由 $P(y\mid\theta)$ 来指定。类型发生的频率由 $P(\theta)$ 给出。如果有大的、预分类的数据集，那么就可以可靠地统计这些分布。特征空间通常是连续的，它的密度是 $p(y\mid\theta)$，即使 $P(\theta)$ 仍然是一个离散概率分布。最优分类器 π^* 根据式（9.26）来设计。它是通过接收特征向量 y 并断言类型就是 $u=\pi^*(y)$ 来实现的。用式（9.32）所得到的期望代价是误差的概率。

例 9.11（**光学字符识别**）一个分类的例子是光学字符识别（optical character recognition）问题。假设摄像机拍摄出一页文本的数字图像。首先对图像进行划分来确定每个字母的位置。然后必须对单个字母进行正确分类。令 $\Theta=\{A,B,C,D,E,F,G,H\}$，它通常包括字母表的所有字母。

假设有三种不同的图像处理算法：

- **形状提取**：如果字母仅由直线边组成，那么返回 $s=0$；如果它至少包含一条曲线，那么返回 $s=1$。
- **端点计数器**：返回端点的数量 ε。例如 O 为零，X 为 4。
- **孔计数器**：返回被字母围绕的孔的数量 h。例如 X 的孔数为零，O 有一个孔。

特征向量是 $y=(s,e,h)$。在理想条件下应该报告的向量如图 9.1 所示。它们代表 $\Theta(s)$、$\Theta(e)$ 和 $\Theta(h)$。它们的交集产生 $\Theta(y)$。

Shape	0	A E F H
	1	B C D G
Ends	0	B D
	1	
	2	A C G
	3	F E
	4	H
Holes	0	C E F G H
	1	A D
	2	B

图 9.1 从字母到特征值的一个映射

假设在非确定模型下和特征总能够提供正确的信息的情况下进行分类。对于 $y=(0,2,1)$，仅可能的字母是 A。对于 $y=(1,0,2)$，仅可能的字母是 B。如果每个 (s,e,h) 仅与一个字母相对应或者不与任何字母相对应，那么就可以构建一个完美的分类器。非常遗憾，(0,3,0)与 E 和 F 相对应。在最坏情况下，使用式（9.32）产生的代价为 1。

解决这个问题的一个方法是引入一个新的特征。假设采用一个图像处理算法来对角进行检测。这是两条线以一定角度（90°）相交的地方。令 c 表示角的数量、新的特征向量为 $y=(s,e,h,c)$。新的算法明确区分 E 和 F，因为它们的 c 分别为 c=2 和 c=1。现在就可以准确无误地对所有字母进行分类。

当然在实际情况下，图像处理算法偶尔会发生错误。设计贝叶斯分类器来最大化成功的概率。假设观测是条件独立的，这意味着可以认为分类是朴素的。假设四种图像处理算法都在一个训练数据集上运行，并且记录结果。在每种情况下，进行正确的分类来得到概率 $P(s|\theta)$、$P(e|\theta)$、$P(h|\theta)$ 和 $P(c|\theta)$。例如，假设孔计数器收到字母 A 作为输入。在文本中多次出现 A 时，算法可以确定 $P(h=1|\theta=A)=0.9$，这是正确的答案。$P(h=0|\theta=A)=0.09$ 和 $P(h=2|\theta=A)=0.01$ 可能以小概率出现。假设在给定输入字母的情况下，每个图像处理算法的输出是相互独立的，那么联合概率为：

$$P(y|\theta)=P(s,e,h,c|\theta)=P(s|\theta)P(e|\theta)P(h|\theta)P(c|\theta) \tag{9.33}$$

先验概率 $P(\theta)$ 的值是通过在大量手工分类文本上运行分类器并且记录每个字母发生的相对次数来得到的。还可以融入一些特定信息。如果已知文本用的是西班牙语，那么 $P(\theta)$ 应该不同于用英语编写的文本。将 $P(\theta)$ 调整为将会出现的文本的类型有助于提高所产生的分类器的性能。

分类器通过选择最小化误差概率的行动来进行决策。忽略式（9.26）中贝叶斯公式的分母中的常数 $P(y)$，那么该误差与：

$$\sum_{\theta\in\Theta}P(s|\theta)P(e|\theta)P(h|\theta)P(c|\theta)P(\theta) \tag{9.34}$$

成比例。 ■

9.2.4.2　参数估计

本节决策的另外一个重要应用是参数估计[91,270]。在这种情况下，大自然选择一个参数 $\theta\in\Theta$，Θ 表示参数空间。通过一次或者多次独立的试验，可以得到一些观测。理想情况下，每个观测都应该是对 Θ 的一个直接测量，但是测量过程中的不完美使得观测发生偏差。通常 $\Theta\subseteq Y$，在一些情况下，$Y=\Theta$。机器人行动就是猜测大自然所选择的参数。因此 $U=\Theta$。在大多数应用中，所有空间都是 $\mathbb{R}^n$ 的连续子集。所设计的代价函数会随着误差 $\|u-\theta\|$ 增大而增大。

例 9.12（参数估计） 假设 $U=Y=\Theta=\mathbb{R}$。因此大自然选择实值参数，该参数需要估计。发生错误的代价是：

$$L(u,\theta)=(u-\theta)^2 \tag{9.35}$$

假设采用贝叶斯方法。给定先验概率密度 $p(\theta)$ 为区间 $[a,b]\subset\mathbb{R}$ 上的均匀分布。得到一个观测，但是它是有噪声的。噪声可以被建模为大自然的第二个行动，如第 9.2.3 节所描述的。这产生了一个密度 $p(y|\theta)$。假设噪声建模为高斯白噪声，那么：

$$p(y|\theta)=\frac{1}{\sqrt{2\pi\sigma}}e^{-(y-\theta)^2/2\sigma^2} \tag{9.36}$$

其中 θ 是均值，σ 是标准差。

基于 y 的最优参数估计的获得是通过选择 $u\in\mathbb{R}$ 来最小化：

$$\int_{-\infty}^{\infty} L(u,\theta)p(\theta \mid y)\mathrm{d}\theta \tag{9.37}$$

其中：

$$p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{p(y)} \tag{9.38}$$

是贝叶斯公式。$p(y)$ 这一项不依赖于 θ，因此在优化中忽略它。采用的先验密度在 $[a,b]$ 以外为 $p(\theta)=0$；因此积分区域可以限制在 $[a,b]$。$p(\theta)=1/(b-a)$ 也是一个可以在优化中忽略的常数。使用式（9.36），这意味着选择 u 来优化：

$$\int_a^b L(u,\theta)p(y \mid \theta)\mathrm{d}\theta \tag{9.39}$$

它可以用标准误差函数 $\mathrm{erf}(x)$ 来表达（一个区间上的高斯分布密度从 0 积分到一个常数）。

如果得到一个独立的观测序列 $y_1,\cdots,y_k$，那么式（9.39）将被：

$$\int_a^b L(u,\theta)p(y_1 \mid \theta)\cdots p(y_k \mid \theta)\mathrm{d}\theta \tag{9.40}$$

代替。 ■

9.3 两人零和对策

第 9.2 节涉及一个真实决策者（机器人）与一个虚拟决策者（大自然）之间的对策。现在假设第二个决策者是一个智能对手，它决策的方式与机器人相同。在这种情况下，两个决策者同时进行决策，它们互相不知道对方将采取什么行动。本节假设决策者的利益截然相反。它们所进行的对策是一方的损失就是另外一方的收获，相反亦然。这是对策论的基础，称为零和对策。

9.3.1 对策表述

假设有两个局中人 P_1 和 P_2，它们各自必须进行决策。它们分别都有一个有限的行动集合 U 和 V。集合 V“代替”表述 9.3 中对手所采取的行动的集合 Θ。每个局中人有一个代价函数，表示为 $L_i: U\times V \to \mathbb{R}$，$i=1,2$。零和对策的一个重要约束是：

$$L_1(u,v) = -L_2(u,v) \tag{9.41}$$

这意味着一个局中人的代价就是另外一个局中人的收益。这是零和的基础，即两个代价之和为零。在零和对策中，局中人的利益是完全相反的。在第 9.4 节中，将取消这种约束，我们更可以得到更一般的对策。

从式（9.41）来看，实际上只有一个代价。所有去掉下标，用 L 来表示 P_1 的代价 L_1。P_1 的目标是最小化 L。由于式（9.41），P_2 的目标是最大化 L。因此 L 可以认为是 P_2 的收益，P_1 的代价。

现在给出表述。

表述 9.7（零和对策）

1. 两个局中人 P_1 和 P_2。
2. 对 P_1 来说有一个被称为行动空间的非空有限集合 U。为便于描述，假设 U 是从 1

到$|U|$的连续整数的集合。每个$u \in U$都是P_1的一个行动。

3. 对P_2来说有一个被称为行动空间的非空有限集合 V。假设 V 是从 1 到$|V|$的连续整数的集合。每个$v \in V$都是P_2的一个行动。

4. 一个被称为P_1的代价函数的函数$L: U \times V \to \mathbb{R} \cup \{-\infty, \infty\}$。由于式（9.41），该函数也作为$P_2$的收益函数。

在讨论解决零和对策的意义之前，还需要一些假设。假设局中人互相知道对方的代价函数。这意味着对手的意图完全已知。还假设局中人都是理性的，在可能的情况下，它们都试图得到最好的代价。当较低代价的行动可行时，P_1不会选择导致更高代价的行动。同样地，P_2也不会选择导致更低代价的行动。最后，假设两个局中人同时进行决策。没有关于P_1的决策信息可供P_2利用，反之亦然。

表述 9.7 通常被称为矩阵对策，因为 L 可以用一个代价矩阵来表示，参见第 9.2 节。这里的矩阵指出P_1和P_2的代价，而不是机器人和大自然的代价。表述 9.7 中要求的所有信息都由单个矩阵指定，因此对于表达零和对策来说它是一个非常方便的表示形式。

例 9.13（零和对策的矩阵表示） 假设P_1的行动集合 U 包含 3 个行动，V 包含 4 个行动。那么在代价函数的指标中应该有 3×4=12 个值。这可以表示为一个代价矩阵：

		V			
U		1	3	3	2
		0	−1	2	1
		−2	2	0	1

（9.42）

其中每一行对应于某$u \in U$，每一列对应于某$v \in V$。每个元素产生$L(u,v)$，这是P_1的代价。这种表示与例 9.8 相似，除了大自然行动空间Θ被 V 代替以外。P_2的代价是$-L(u,v)$。■

9.3.2　确定的策略

什么构成了表述 9.7 的解？首先从P_1的角度来考虑对策。当试图说明P_2将要采取的行动时，应用最坏情况分析似乎是合理的。这样所产生的选择等价于假设P_2是第 9.2.2 节所考虑的在非确定模型下进行行动的大自然。对于一个矩阵对策来说，这可以通过首先确定每一行的最大代价来计算。在其中选择产生最小值的行动就表示P_1对它自己所能够确保的最小代价。这种选择被称为P_1的安全策略（security strategy）。

对于式（9.42）中的矩阵对策，安全策略阐述为：

		V				
U		1	3	3	2	→3
		0	−1	2	1	→2
		−2	2	0	1	→2

（9.43）

其中 u=2 和 u=3 是最好的行动。它们产生的代价不超过 2，不管P_2采取什么行动。

这可以用已有的符号来表示。P_1的一个安全策略u^*一般被定义为：

$$u^* = \arg\min_{u\in U}\left\{\max_{v\in V}\{L(u,v)\}\right\} \tag{9.44}$$

可能有多个安全策略满足 arg min；但是这不会引起麻烦，稍后将进行解释。令所产生的最坏情况代价用 $\bar{L}^*$ 表示，我们称之为对策的上值。它被定义为：

$$\bar{L}^* = \max_{v\in V}\left\{L(u^*,v)\right\} \tag{9.45}$$

现在交换角色，从 P_2 的角度考虑对策，它将最大化 L。仍然采用最坏情况分析，即所选择的行动保证高代价，尽管 P_1 的行动会减小它。P_2 的安全策略 v^* 被定义为：

$$v^* = \arg\max_{v\in V}\left\{\min_{u\in U}\{L(u,v)\}\right\} \tag{9.46}$$

它与式（9.44）相对称。P_2 可能有多个安全策略。安全策略 v^* 仅仅是第 9.2.2 节用的最坏情况分析的一个“颠倒”版本。下值 $\underline{L}^*$ 被定义为：

$$\underline{L}^* = \min_{u\in U}\left\{L(u,v^*)\right\} \tag{9.47}$$

回到式（9.42）的矩阵对策，通过使用式（9.46），那么会选择最后一列：

$$\begin{array}{cc} & V \\ U & \begin{array}{|cccc|}\hline 1 & 3 & 3 & 2 \\ 0 & -1 & 2 & 1 \\ -2 & 2 & 0 & 1 \\ \hline \downarrow & \downarrow & \downarrow & \downarrow \\ -2 & -1 & 0 & 1 \\ \hline \end{array} \end{array} \tag{9.48}$$

对于使用表述 9.7 的任何对策，上值和下值之间一个有趣的关系是 $\underline{L}^* \leqslant \bar{L}^*$。这可以通过下式来得到：

$$\underline{L}^* = \min_{u\in U}\left\{L(u,v^*)\right\} \leqslant L(u^*,v^*) \leqslant \max_{v\in V}\left\{L(u^*,v)\right\} = \bar{L}^* \tag{9.49}$$

其中 $L(u^*,v^*)$ 是当局中人应用它们各自的安全策略时所得到的代价。如果局中人是理性的决策者，那么所产生的代价总是在 $\underline{L}^*$ 和 $\bar{L}^*$ 之间。

遗憾 假设局中人应用安全策略 $u^*=2$ 和 $v^*=4$。这产生了一个代价 $L(2,4)=1$。那么结果产生之后局中人感觉如何？P_1 可能感觉满意，因为给定 P_2 的选择 $v^*=4$，它得到可能最低的代价。另一方面，按照 P_1 所选择的行动，P_2 可能对它自己的决策会感到遗憾。如果已知 P_1 将会选择 $u=2$，那么 P_2 将采取 $v=2$ 使得 $L(2,2)=2$，这比 $L(2,4)=1$ 好。如果对策重复进行，那么 P_2 会改变它的策略以得到更高的收益。

有什么方法使得两个局中人都感到满意呢？在 $\underline{L}^*$ 和 $\bar{L}^*$ 之间任何时候都有差距。有一个或者两个局中人会感到遗憾。如果 r_1 和 r_2 分别表示 P_1 和 P_2 所经历的遗憾的数量值，那么总遗憾为：

$$r_1 + r_2 = \bar{L}^* - \underline{L}^* \tag{9.50}$$

因此满足两个局中人的唯一方法是使 $\underline{L}^* = \bar{L}^*$。这是对策所具有的性质，但是它们不是局中人所决定的。对于某些对策来说，上值和下值是相等的，但是对于另外一些对策来说却是 $\underline{L}^* < \bar{L}^*$。幸运的是，通过采用随机策略，上值和下值总是一致的；这将在第 9.3.3

节介绍。

鞍点　如果 $\underline{L}^* = \overline{L}^*$，那么安全策略称为一个鞍点，$L^* = \underline{L}^* = \overline{L}^*$ 被称为对策值。如果这种情况发生，那么 max 和 min 的顺序可以交换而不会改变对策值：

$$L^* = \min_{u \in U}\left\{\max_{v \in V}\{L(u,v)\}\right\} = \max_{v \in V}\left\{\min_{u \in U}\{L(u,v)\}\right\} \tag{9.51}$$

鞍点有时称为平衡点，因为局中人没有改变它们各自选择的动机（因为没有遗憾）。一个鞍点定义为：对于所有 $u \in U$ 和 $v \in V$。使得：

$$L(u^*,v) \leqslant L(u^*,v^*) \leqslant L(u,v^*) \tag{9.52}$$

成立的任意 $u^* \in U$ 和 $v^* \in V$ 就是鞍点。注意到 $L^* = L(u^*,v^*)$。当观察一个矩阵对策，鞍点通过发现如图 9.2 所示的简单图形来得到。

例 9.14（一个确定的鞍点）　这里的矩阵对策有一个鞍点：

$$\begin{array}{c|ccc} & V & & \\ U & 3 & 3 & 5 \\ & 1 & -1 & 7 \\ & 0 & -2 & 4 \end{array} \tag{9.53}$$

应用式（9.52）（或者图 9.2），当 $u=3$ 和 $v=3$ 时可以得到鞍点。结果是 $L^* = 4$。在这种情况下，对策结束以后没有局中人感到遗憾。P_1 感到满意，因为在给定 P_2 选择第三列的情况下，4 是它所得到的最小代价。同样地，在给定 P_1 选择最后一行的情况下，4 是 P_2 得到的最高代价。■

如果在一个对策中有多个鞍点会怎样？这可能是个问题，因为局中人无法协调它们的决策。如果 P_1 试图执行一个鞍点，而 P_2 试图执行另外一个鞍点，那将会怎样？如果有多于一个的鞍点，那么鞍点一定至少有 4 个，如图 9.3 所示。当我们试图像图 9.2 所示产生两个“＋”图案时，它们会相交产生 4 个鞍点。当加入更多鞍点时也会发生相似的情况。

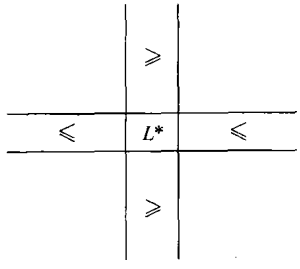


图 9.2　在一个矩阵中寻找一个值 L^* 使得它在所在列的所有元素中是最低的，并且在所在行的所有元素中是最大的，这样就可以得到一个鞍点

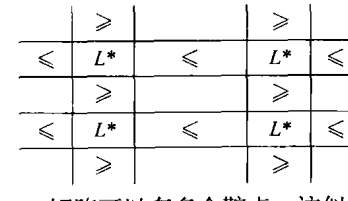


图 9.3　矩阵可以有多个鞍点，这似乎会产生局中人之间的协调问题；但幸运的是不存在这种问题，因为不管每个局中人选择哪个鞍点，都会得到相同的值

例 9.15（多个鞍点）　这个对策有多个鞍点，它遵从图 9.3 中的规律：

$$U \quad \begin{array}{|c|c|c|c|c|} \hline 4 & 3 & 5 & 1 & 2 \\ \hline -1 & 0 & -2 & 0 & -1 \\ \hline -4 & 1 & 4 & 3 & 5 \\ \hline -3 & 0 & -1 & 0 & -2 \\ \hline 3 & 2 & -7 & 3 & 8 \\ \hline \end{array} \quad V \tag{9.54}$$

令 (i,j) 分别表示 P_1 和 P_2 的选择对。(2,2)和(4,4)都是值 V=0 的鞍点。如果 P_1 选择 u=2，P_2 选择 v=4，那么会怎样？这不存在问题，因为(2,4)也是一个鞍点。同样地，(4,2)是另外一个鞍点。一般情况下，多个鞍点不会产生问题，因为所产生的代价独立于每个决策者所试图进行的鞍点。 ■

9.3.3 随机策略

某些零和对策没有鞍点，这是令人失望的，因为在这种情况下遗憾是不可避免的。我们稍微改变一下规则。假设在大量试验中 P_1 和 P_2 重复进行相同的对策。如果它们采用确定的策略，那么它们每次将选择相同的行动，也就产生相同的代价。它们可能在多个可选的安全策略之间进行切换，这引起代价值的起伏波动。如果它们执行随机策略将会怎样？采取第 9.1.3 节的思路，指定的每个策略将被作为行动上的概率分布。在极限状态下，当对策的进行次数趋近无穷时，将得到一个期望代价。在对策论中一个最著名的结论是：在随机策略的空间上，任何零和矩阵对策的鞍点都存在，但是必须采用期望代价。因此，如果采用随机化，将不会有遗憾。在单次试验中可能有遗憾；但是当代价在所有试验上平均时，两个局中人将都会感到满意。

9.3.3.1 对表述的扩展

因为表述 9.7 下的对策可以表示为一个矩阵，那么就可以用线性代数来表达期望代价。令 $|U|=m$ 和 $|V|=n$。就像第 9.1.3 节中介绍的那样，P_1 的一个随机策略可以表示为一个 m 维向量：

$$w=[w_1 \quad w_2 \quad \cdots \quad w_m] \tag{9.55}$$

必须满足第 9.1.2 节的概率性质：（1）对于所有 $i\in\{1,\cdots,m\}$，$w_i\geqslant 0$；（2）$w_1+\cdots+w_m=1$。如果认为 w 是 $\mathbb{R}^m$ 中的一个点，那么这两个约束条件的含义是该点必须落在一个 m–1 维单形上（见第 6.3.1 节）。如果 m=3，这意味着 w 落在 $\mathbb{R}^3$ 的一个三角形子集中。相似地，令 z 表示 P_2 的用 n 维向量表示的一个随机策略：

$$z=[z_1 \quad z_2 \quad \cdots \quad z_n]^T \tag{9.56}$$

它也满足概率性质。在式（9.56）中，上标 T 表示转置，它产生一个满足维数约束的列向量，这种维数约束在矩阵乘法中是必需的。

令 $\bar{L}(w,z)$ 表示期望代价，它是如果 P_1 采用 w、P_2 采用 z 时所得到的代价。这可以计算为：

$$\bar{L}(w,z)=\sum_{i=1}^{m}\sum_{j=1}^{n}L(i,j)w_i z_j \tag{9.57}$$

代价 $L(i,j)$ 利用了表述 9.7 中的假设，即行动是连续的整数。期望代价也可以用代价矩阵 A 来表达。即：

$$\bar{L}(w,z)=wAz \tag{9.58}$$

其中乘积 wAz 产生的一个标量值就是式（9.57）。为了便于理解，首先考虑乘积 Az。它产生一个 m 维向量，其中第 i 个元素是如果 P_1 尝试 $u=i$ 时所得到的期望代价。因此 P_1 将 P_2 看作概率模型下的一个大自然局中人。w 和 Az 相乘就得到一个标量，该量按照 w 的概率在向量 Az 中对代价进行平均。

令 W 和 Z 分别表示 P_1 和 P_2 的所有随机策略的集合。通过给单个行动分配概率 1，这些空间包括的策略等价于第 9.3.2 节中所考虑的确定策略。因此对照在确定情形下可行的策略，W 和 Z 可以认为是可能策略集的扩展。采用 W 和 Z，P_1 和 P_2 的随机安全策略分别被定义为：

$$w^*=\arg\min_{w\in W}\left\{\max_{z\in Z}\left\{\bar{L}(w,z)\right\}\right\} \tag{9.59}$$

$$z^*=\arg\max_{z\in Z}\left\{\min_{w\in W}\left\{\bar{L}(w,z)\right\}\right\} \tag{9.60}$$

将它们与式（9.44）和式（9.46）进行比较。不同之处在于策略空间已经被扩展，并且现在使用的是期望代价。

随机化的上值被定义为：

$$\bar{\mathcal{L}}^*=\max_{z\in Z}\left\{\bar{L}(w^*,z)\right\} \tag{9.61}$$

随机化的下值被定义为：

$$\underline{\mathcal{L}}^*=\min_{w\in W}\left\{\bar{L}(w,z^*)\right\} \tag{9.62}$$

因为 W 和 Z 包括确定的安全策略 $\bar{\mathcal{L}}^*\leqslant\bar{L}^*$ 和 $\underline{\mathcal{L}}^*\geqslant\underline{L}^*$。这些不等式暗示随机安全策略在一般情况下的愿望是缩小两个值之间的差距。

Von Neumann[956,957]给出零和对策论中最基本的结果，它论述到：对于表述 9.7 中的任何对策，有 $\underline{\mathcal{L}}^*=\bar{\mathcal{L}}^*$。这将产生对策的随机值 $\mathcal{L}^*=\underline{\mathcal{L}}^*=\bar{\mathcal{L}}^*$。这意味着如果局中人坚持采用安全策略，那么将不会有期望的遗憾。如果局中人应用它们的随机安全策略，那么得到是随机鞍点。这种鞍点不能像矩阵 A 中简单图形那样可以看见，因为它存在于 W 和 Z 上。

随机鞍点的存在性是一个重要的结论，因为当与智能对手进行决策时它展示的是随机化的值。例 9.7 中直观地指出：与智能对手进行对策时，随机化似乎是有帮助的。当用确定的策略重复进行对策时，其他局中人能够获悉这些策略并在每次对策中取得胜利。所以采用随机策略，局中人才不会有遗憾。

9.3.3.2　随机鞍点的计算

到目前为止已经确定随机鞍点总是存在的，但是如何发现鞍点呢？基于两方面考虑来得到问题的组合解：

（1）只考虑对手的确定策略来得到每个局中人的安全策略。

（2）如果其他局中人的策略固定，那么期望代价是还没有确定的概率的线性函数。

首先考虑为 P_1 确定安全策略。第一点意味着式（9.59）不需要考虑 P_2 的随机策略。在

argmin 中固定 w。什么随机策略 $z \in Z$ 最大化 $\bar{L}(w,z) = wAz$？如果固定 w，那么 wA 可以作为一个 n 维常数向量 s 来对待。这意味着 $\bar{L}(w,z) = s \cdot z$，其中“$\cdot$”表示内积（点积）。现在的任务是选择 z 来最大化 $s \cdot z$。这涉及选择 s 的最大元素；假设最大元素为 s_i。通过将所有概率质量放置在行动 i 上就可以得到所有 $z \in Z$ 上的最大代价。因此策略 $z_i = 1$ 和 $z_j = 0\,(i \neq j)$给出最高代价，它是确定的。

基于第一点，那么对于每个 $w \in W$，只需要考虑 P_2 的 n 个可能的响应。这是 n 个确定的策略，对于唯一 $i \in \{1,\cdots,n\}$，每个策略分配 $z_i = 1$。

现在考虑第二点。如果固定 z，那么期望代价 $\bar{L}(w,z) = wAz$ 是 w 的线性函数。因为基于第一点只需要在 n 个不同值处固定 z，w 选择在 n 个线性函数中最小的最大值发生的点处。这是线性函数族的上包络的最小值。这种包络曾在第 6.5.2 节讲述过。例 9.16 将阐述这个问题。这种优化的区域可以被设置为 $\mathbb{R}^{m-1}$ 中的一个三角形。即使 $W \subset \mathbb{R}^m$，也不需要最后一个坐标 w_m，因为 $w_m = 1 - (w_1 + \cdots + w_{m-1})$。所产生的优化就成为线性规划，有许多组合算法能够解决这种问题[261,266,667,734]。

在上述说明中，对 P_1 寻找其安全策略来说并没有什么特别的。相同的方法可以用来确定 P_2 的安全策略；但是每个最小化用最大化代替，反之亦然。总之，式（9.60）中的 min 仅需要考虑 W 中的确定策略。如果固定 w，那么 $\bar{L}(w,z) = wAz$ 还是线性函数，但是这一次它关于 z 是线性的。通过在 m 个线性函数中发现能给出最高的最小值的点 $z \in Z$ 来选择最好的随机行动。这是线性函数族的下包络的最小值。优化发生在 $\mathbb{R}^{n-1}$ 上，因为最后一个坐标可直接从 $z_n = 1 - (z_1 + \cdots + z_{n-1})$ 得到。

通过一个例子可以更好地理解这种计算方法。

例 9.16（计算一个随机鞍点） 最简单的情况是当两个局中人仅有两个行动。令代价矩阵的定义为：

		V	
U		3	0
		−1	1

（9.63）

考虑计算 P_1 的安全策略。考虑到 W 和 Z 只是 $\mathbb{R}^2$ 的一维子集。P_1 的一个随机策略是 $w = [w_1, w_2]$，其中 $w_1 \geqslant 0$，$w_2 \geqslant 0$，$w_1 + w_2 = 1$。因此优化的区域是 $w_1 \in [0,1]$，因为 w_2 总可以从 $w_2 = 1 - w_1$ 得到。使用上述的第一个观测，对 P_2 来说仅需要考虑两个确定的策略。当作为 w 的线性函数时，对于 $z_1 = 1$，它们是：

$$(3)w_1 + (-1)(1 - w_1) = 4w_1 - 1 \tag{9.64}$$

对于 $z_2 = 1$：

$$(0)w_1 + (1)(1 - w_1) = 1 - w_1 \tag{9.65}$$

如图 9.4（a）所示的绘制直线。安全策略通过沿着图中显示的上包络的最小值点来确定。这由粗线来指示，在一般情况下它总是分段线性函数。最低点发生在 $w_1 = 2/5$，所产生的值为 $\mathcal{L}^* = 3/5$。因此 $w^* = [2/5 \;\; 3/5]$。

应用一个相似的过程可以产生 z^*。对应于 P_1 确定策略的直线如图 9.4（b）所示。安全

策略通过发现沿着线的下包络的最大值来得到，如图中的粗线所示。这产生 $z^* = [1/5 \ \ 4/5]^T$，得到的值还是 $\mathcal{L}^* = 3/5$（它必须与前面的值一致，因为随机上值和下值是相同）。 ■

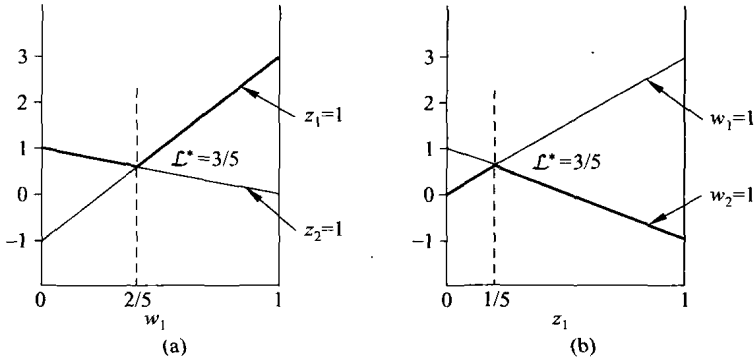


图 9.4　（a）计算 P_1 的随机安全策略 w^*；（b）计算 P_2 的随机安全策略 z^*

如果每个局中人仅有两个行动，那么该过程很简单。如果 $n = m = 100$，那么上包络和下包络是 $\mathbb{R}^{99}$ 中的分段线性函数。这对计算来说是不现实的，因为所有现有的线性规划算法的运行时间至少与维数呈指数关系[266]。

9.4　非零和对策

本节与第 9.3 节是并行的，只不过这里考虑的是更一般的非零和对策。它对有任何想要的冲突程度的对策进行建模。某些决策可能甚至对所有局中人都有利。这种情况的一个主要应用是在经济领域，在那里非零和对策用来解释竞争中的商业行为。

鞍点解将用 Nash 平衡点来代替，它也基于消除遗憾。因为局中人不必是相互冲突的，那么有可能对涉及任意数量局中人的对策进行建模。对于非零和对策，也会出现新的困难，诸如 Nash 平衡点的不唯一性以及在一般情况下随机 Nash 平衡点的计算并不适于线性规划等。

9.4.1　两人对策

为了实现从第 9.3 节的平滑过渡，首先考虑两人对策。这种情况比较容易理解，因为符号更简单。然后将思路从两人对策扩展到多人对策。对策表述如下。

表述 9.8（两人非零和对策）

1. 与表述 9.7 的组成相同，但是代价函数不同。
2. 一个被称为 P_1 的代价函数的函数 $L_1 : U \times V \to \mathbb{R} \cup \{\infty\}$。
3. 一个被称为 P_2 的代价函数的函数 $L_2 : U \times V \to \mathbb{R} \cup \{\infty\}$。

与表述 9.7 仅有的不同之处在于现在对于两个局中人有两个独立的代价函数 L_1 和 L_2。每个局中人将最小化它的代价。这里没有最大化；最大化出现在零和对策中，那是因为 P_2

的利益与 P_1 相反。表述 9.7 中的零和对策可以建模为 $L_1 = L$ 和 $L_2 = -L$。

并行于第 9.3 节，首先考虑应用确定的策略来解决对策问题。一种可能是一个局中人可以应用它的安全策略。为此，甚至可以不必观测其他局中人的代价函数。但是在决策过程中不考虑代价函数似乎是不恰当的。在多数情况下，安全策略会产生遗憾，这使得它们不适合非零和对策。

我们将给出避免遗憾的策略。一对行动 (u^*, v^*) 被定义为 Nash 平衡点，如果：

$$L(u^*, v^*) = \min_{u \in U} \{L(u, v^*)\} \tag{9.66}$$

$$L(u^*, v^*) = \min_{v \in V} \{L(u^*, v)\} \tag{9.67}$$

这含有 P_1 和 P_2 都没有遗憾的意思。方程（9.66）指出：在给定 P_2 的行动 v^* 时，P_1 对它自己的行动 u^* 感到满意。P_1 不能通过改变它自己的行动来进一步减小它的代价。同样地，式（9.67）指出 P_2 对它的行动 v^* 感到满意。

表述 9.8 中的对策完全可以用两个代价矩阵来表示。令 A 和 B 分别表示 P_1 和 P_2 代价矩阵。回忆图 9.2 显示的检测鞍点的方式。一个 Nash 平衡点的检测如图 9.5 所示。考虑它们两者之间的关系。如果 $A = -B$，那么 B 可以取反，然后添加在 A 的上面。这将产生图 9.2 的情况（因为取反，每个“≥”变成“≤”）。在这种情况下 L_a^* 和 L_b^* 一致。这意味着如果 $A = -B$，那么 Nash 平衡点与鞍点实际上是相同的概念。只不过 Nash 平衡点应用于更一般的对策。

例 9.17（一个确定的 Nash 平衡点） 考虑由代价矩阵 A 和 B 指定的对策。

A：

		V		
	9	4	7
U	6	−1	5
	1	4	2

B：

		V		
	2	1	6
U	5	0	2
	2	2	5

（9.68）

应用式（9.66）和式（9.67），或者采用图 9.5 的方式，可以看到 $u = 3$ 和 $v = 1$ 是一个 Nash 平衡点。所产生的代价是 $L_1 = 1$ 和 $L_2 = 2$。另外一个 Nash 平衡点出现在 $u = 2$ 和 $v = 2$。这产生的代价 $L_1 = -1$ 和 $L_2 = 0$ 对两个局中人来说更好。

对于零和对策，多个鞍点的存在不会产生任何问题；但是对于非零和对策来说却有大的麻烦。在本例中，其中一个 Nash 平衡点比其他 Nash 平衡点更好。因此理性的决策者应该选择更好的一个。下面将介绍多个 Nash 平衡点的问题。 ■

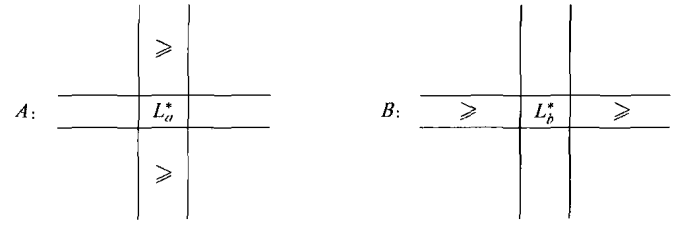


图 9.5　对于两个矩阵 A 和 B，寻找某(i,j)使得在 A 的第 j 列中的所有元素中 $L_a^* = L_1(i,j)$ 是最低的，也使得在 B 的第 i 行中的所有元素中 $L_b^* = L_2(i,j)$ 是最低的，这样就可以得到一个 Nash 平衡点；请与图 9.2 进行比较

9.4.1.1　处理多个 Nash 平衡点

由于多个 Nash 平衡点的存在，例 9.17 好像有点令人沮丧。一般情况下，可能有任意多个平衡点。每个局中人如何确定采用哪一个？如果每个局中人选择不同的 Nash 平衡点，那么不能保证它们像零和对策的鞍点那样落入另外一个 Nash 平衡点。用 Pareto 最优性可以消除一些平衡点，Pareto 最优性在第 9.1.1 节介绍过，它也作为多机器人的一个最优协调方法在第 7.7.2 节出现过。思路是将选择表示为一个多目标优化问题，即表述 9.2 所描述的问题。

考虑 (x_i, y_i) 形式的两维向量，其中 x 和 y 表示在 Nash 平衡点 π_i 所得到的代价 L_1 和 L_2。对于两个不同的平衡点 π_1 和 π_2，得到代价向量 (x_1, y_1) 和 (x_2, y_2)。在例 9.17 中，它们是(1,2)和(−1,0)。一般情况下，如果 $x_1 \leqslant x_2$ 和 $y_1 \leqslant y_2$，并且至少有一个不等式是严格的，那么认为 π_1 比 π_2 好。在例 9.17 中，产生(−1,0)的平衡点比产生(1,2)的平衡点好，因为两个局中人都受益。

“好”的定义在 Nash 平衡点空间上产生了一个偏序。之所以称为偏序，是因为某些向量是不可比较的。例如(−1,1)和(1,−1)。P_1 偏爱第一个向量，而 P_2 优先选择第二个。在对策论中，如果 Nash 平衡点关于这种偏序是最小的，那么称之为可允许的，也称为 Pareto 最优的。

最好的情形是对策只有一个 Nash 平衡点。如果有多个 Nash 平衡点，那么希望只有一个是可允许的。在这种情况下，希望理性的局中人有足够的智能来确定那些应该抛弃的非允许平衡点。遗憾的是，许多对策都有多个可允许 Nash 平衡点。在这种情况下，分析指出：局中人必须能够以某种形式进行通信和合作来消除产生遗憾的可能性。否则在最坏情况下遗憾是不可避免的。当然也可能没有 Nash 平衡点，但幸运的是，通过随机策略总能够保证随机 Nash 平衡点的存在。这将在下面两个例子之后进行介绍。

例 9.18（性别之战）　考虑由矩阵 A 和 B 指定的对策。

A:

U \ V	
−2	0
0	−1

B：（9.69）

U \ V		
	−1	0
	0	−2

这就是著名的被称为“性别之战”的对策。假设一个男人和一个女人在消遣方式的问题上存在分歧。男人偏爱购物，而女人偏爱看足球比赛。该对策涉及对这两种行动的选择。对他们每个人来说最好的情况是在做他们喜欢的事的同时他们仍然在一起。最坏的情况是选择不同的行动使得分开他们。这个对策是不切实际的，因为在多数情形中他们之间有某种合作。

$u=v=1$和$u=v=2$都是 Nash 平衡点，它们产生代价向量(−2,−1)和(−1,−2)。一个平衡点不比另外一个更好；因此它们都是可允许的。没有办法来避免遗憾，除非局中人进行合作。■

下面就是一个最著名的非零和对策的例子。

例 9.19（囚犯的艰难抉择） 下面的对策很容易表达，它阐述了一些有趣的问题。假设两个人有犯罪行为。司法部门知道他们是有罪的，但是没有足够的证据。因此制订了一个计划来欺骗疑犯。每个疑犯（或者局中人）被关在一个隔离的狱室内，并有两个选择。每个局中人可以与司法部门合作，即 u=1 或者 v=1，也可以拒绝合作，即 u=2 或者 v=2。如果合作，局中人承认有罪并提供证据。如果拒绝，局中人声称是自己是清白的并拒绝给司法部门提供任何帮助。

代价 L_i 表示局中人将被关押的年数。代价矩阵如下所示。

A：

U \ V		
	8	0
	32	2

B：（9.70）

U \ V		
	8	30
	0	2

目的是如果他们都与司法部门进行合作，判两个局中人 8 年监禁，这是因犯罪所判的年限和与司法部门进行合作而减刑后最后所应该关押的年限。如果他们都拒绝，那么他们将被判两年监禁，因为没有足够的证据。有趣的情况是如果一个拒绝而另外一个合作。拒绝合作的一个会有更大麻烦，因为另外一个提供的证据将会对他不利。合作的一方将获得自由（代价为 0）；但是另外一方将被判有罪而监禁 30 年。

局中人应该怎样做？你应该怎样做？如果他们之间进行了协调，那么好的选择是他们都拒绝。这样的代价是(2,2)。但是在这种情况下他们都有遗憾，因为他们都有机会获得自由。如果他们进行第二次对策，那么可能倾向于改变他们的决策。

该问题的 Nash 平衡点是对他们来说都与司法部门进行合作，代价是(8,8)。因此他们因不能进行通信和协调而付出了代价。该结果也是局中人的安全策略，因为，如果用最坏情况进行分析，那么这样的代价是最低的。 ■

9.4.1.2　随机 Nash 平衡点

如果在确定的策略空间上对策没有 Nash 平衡点将会怎样？通过将策略空间扩展到包括随机策略就可以使得问题得到缓和。第 9.3.3 节指出：表述 9.7 下的每个零和对策在随机策略空间上都有一个随机鞍点。Nash 指出：表述 9.8 下的每个非零和对策都有一个随机 Nash 平衡点[733]。这是一个好的结果；但是还有两个问题。一是可能还存在其他可允许的平衡点，即没有可靠的方法来避免遗憾，除非局中人进行合作。二是随机 Nash 平衡点不能用　第 9.3.3 节的线性规划方法来计算。这里的优化是非线性规划的形式[98,667,734]，它不一定产生一个好的组合解。

回忆第 9.3.3 节的随机策略的定义。对于一对随机策略(w,z)，期望代价$\bar{L}^1(w,z)$和$\bar{L}^2(w,z)$可以用式（9.57）来计算。一对策略(w^*,z^*)将被称为随机 Nash 平衡点，如果：

$$\bar{L}^1(w^*,z^*)=\min_{w\in W}\left\{\bar{L}^1(w,z^*)\right\} \tag{9.71}$$

$$\bar{L}^2(w^*,z^*)=\min_{z\in Z}\left\{\bar{L}^2(w^*,z)\right\} \tag{9.72}$$

在对策论文献中，这通常被称为混合 Nash 平衡点。注意到式（9.71）和式（9.72）只是式（9.66）和式（9.67）从确定策略空间到随机策略空间的推广。

采用代价矩阵 A 和 B，Nash 平衡点条件可以写为：

$$w^*Az=\min_{w\in W}\left\{wAz^*\right\} \tag{9.73}$$

$$w^*Bz=\min_{z\in Z}\left\{w^*Bz\right\} \tag{9.74}$$

非常遗憾，随机 Nash 平衡点比鞍点更难计算。主要原因是 Nash 平衡点不一定是安全策略。通过采用安全策略，有可能将局中人的决策解耦为多个单独的线性规划，如例 9.16 所示。对于随机 Nash 平衡点，局中人之间的优化仍是耦合的。所产生的优化通常被称为线性互补问题（linear complementarity problem）。这可以表述为一个非线性规划问题[667,734]，即它是一个涉及定义域上的等式和不等式约束的非线性优化（在这种特殊情况下，出现双线性规划问题[60]）。

例 9.20（发现一个随机 Nash 平衡点） 为了得到所要求的这种优化的思路，回忆例 9.18。可以发现区别于两个确定平衡点的随机 Nash 平衡点。采用例 9.18 的代价矩阵，给定随机策略 w 和 z 的情况下 P_1 期望代价是：

$$\begin{aligned}\bar{L}^1(w,z)=wAz&=\begin{pmatrix}w_1 & w_2\end{pmatrix}\begin{pmatrix}-2 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}z_1\\ z_2\end{pmatrix}\\ &=-2w_1z_1-w_2z_2=-3w_1z_1+w_1+z_1\end{aligned} \tag{9.75}$$

其中最后一步用到 $w_2=1-w_1$ 和 $z_2=1-z_1$。相似地，P_2 的期望代价为：

$$\begin{aligned}\bar{L}^2(w,z)=wBz&=\begin{pmatrix}w_1 & w_2\end{pmatrix}\begin{pmatrix}-1 & 0\\ 0 & -2\end{pmatrix}\begin{pmatrix}z_1\\ z_2\end{pmatrix}\\ &=-w_1z_1-2w_2z_2=-3w_1z_1+2w_1+2z_1\end{aligned} \tag{9.76}$$

如果固定 z，那么式（9.75）中最后一个等式关于 w 是线性的；同样地，如果固定 w，那么式（9.76）关于 z 是线性的。在计算零和对策的鞍点时，允许我们进行这种假设；但是在这里是不可能的。我们必须选择 (w^*,z^*) 来同时优化 $z=z^*$ 下的式（9.75）和 $w=w^*$ 下的式（9.76）。

这种问题可以用微积分来解决。采用经典的求导优化方法，通过计算下式可以得到解：

$$\frac{\partial \overline{L}^1(w_1,z_1)}{\partial w_1}=1-3z_1 \tag{9.77}$$

$$\frac{\partial \overline{L}^2(w_1,z_1)}{\partial z_1}=2-3w_1 \tag{9.78}$$

当这两个方程同时为 0 时产生极值。求解 $1-3z_1=0$ 和 $2-3w_1=0$ 产生随机 Nash 平衡点 $(w^*,z^*)=(2/3,1/3)$。这种方法不能得到确定的 Nash 平衡点，因为它们出现在 W 和 Z 的边界上，那里的导数没有定义。 ■

例 9.20 中的计算方法并不困难，因为每个局中人只有两个行动，并且一半的矩阵代价为 0。而在一般情况下必须同时求解两个复杂的方程。但表达式通常总是二次多项式，并且如果固定 w 或者 z，那么每个多项式变成关于其他变量是线性的。

可能的解 表述 9.8 的非零和对策可能有如下几种解。

（1）可能有多个可允许的（确定的）Nash 平衡点。

（2）可能没有（确定的）Nash 平衡点。

（3）至少有一个随机 Nash 平衡点。

9.4.2 多人对策

将第 9.4.1 节的思路扩展到任意数量的局中人。主要的困难在于记数使得概念更加困难。但是它们没有根本区别。有 n 个局中人的非零和对策表述如下。

表述 9.9（n 个局中人的非零和对策）

1. 局中人的一个集合 $\mathrm{P}_1,\mathrm{P}_2,\cdots,\mathrm{P}_n$。

2. 对每个 P_i 来说有一个被称为行动空间的非空有限集合 U^i。为方便起见，假设每个 U^i 是 $1\sim|U^i|$ 的连续整数的集合。每个 $u^i\in U^i$ 都是 P_i 的一个行动。

3. 对每个 P_i，有一个被称为代价函数的函数 $L_i:U^1\times U^2\times\cdots\times U^n\to\mathbb{R}\cup\{\infty\}$。

因为维数太多，所以不再可能有代价的矩阵表述。例如，如果对于每个局中人，$n=3$ 且 $|U^i|=2$，那么 $L_i(u^1,u^2,u^3)$ 由有 8 个项的 2×2×2 方体来指定。需要其中 3 个方体来指定对策。因此将 L_i 只当作一个多变量函数而避免使用矩阵，这样是非常有益的[1]。

在给定其他 $n-1$ 个局中人所选择的行动时，通过要求每个 P_i 没有遗憾，那么可以将 Nash 平衡点的思路进行扩展。正式地，一个行动集合 $(u^{1*},\cdots,u^{n*})$ 被称为（确定的）Nash 平衡点，如果对于每个 $i\in\{1,\cdots,n\}$，则有：

$$L_i(u^{1*},\cdots,u^{i*},\cdots,u^{n*})=\min_{u^i\in U^i}\left\{L_i(u^{1*},\cdots,u^{(i-1)*},u^i,u^{(i+1)*},\cdots,u^{n*})\right\} \tag{9.79}$$

1 如果你喜欢张量，那么可将这些用来获取 n 个局中人代价函数[109]。

在 $n>2$ 的情况下，第 9.4.1 节结尾所总结的情形都可能发生。可能没有确定的 Nash 平衡点或者多个 Nash 平衡点。通过在 n 维代价向量上定义“更好”这个概念就可以对可允许 Nash 平衡点的定义进行扩展。再次，关于所产生的偏序的最小向量被认为是可允许的（或者 Pareto 最优的）。遗憾的是仍然可能存在多个可允许 Nash 平衡点。

对于表述 9.9 下的任何对策，都存在一个随机 Nash 平衡点。令 z^i 表示 P_i 的一个随机策略。每个 P_i 的期望代价可以表示为：

$$\overline{L}^i(z^1,z^2,\cdots,z^n)=\sum_{i_1=1}^{m_1}\sum_{i_2=1}^{m_2}\cdots\sum_{i_n=1}^{m_n}L_i(i_1,i_2,\cdots,i_n)z_{i1}^1,z_{i2}^2,\cdots,z_{in}^n \tag{9.80}$$

令 Z^i 表示 P_i 的随机策略空间。所有局中人的随机策略 $(z^{1*},\cdots,z^{n*})$ 被称为随机 Nash 平衡点，如果对于所有 $i\in\{1,\cdots,n\}$，有：

$$\overline{L}^i(z^{1*},\cdots,z^{i*},\cdots,z^{n*})=\min_{z^i\in Z^i}\left\{\overline{L}^i(z^{1*},\cdots,z^{(i-1)*},z^i,z^{(i+1)*},\cdots,z^{n*})\right\} \tag{9.81}$$

正如所料，计算 $n>2$ 情况下的随机 Nash 平衡点比 $n=2$ 情况困难得多。例 9.20 的方法可以扩展到 n 个局中人；但是表达式更加复杂。有 n 个方程，如果固定其他 $n-1$ 个局中人的随机策略，那么每个方程是线性的。这样产生的结果是一族 n 阶多项式，在该族上 n 个优化问题必须被同时解决。

例 9.21（3 个局中人的非零和对策） 假设有 3 个局中人 P_1、P_2 和 P_3，它们都有两个行动 1 和 2。确定的策略由向量来指定，例如(1,2,1)表示 $u^1=1$、$u^2=2$ 和 $u^3=1$。

现在定义某些代价。为方便起见，对每个 $i,j,k\in\{1,2\}$，令：

$$L(i,j,k)=\left(L_1(i,j,k),L_2(i,j,k),L_3(i,j,k)\right) \tag{9.82}$$

令代价为：

$$\begin{aligned}
&L(1,1,1)=(1,1,-2) && L(1,1,2)=(-4,3,1)\\
&L(1,2,1)=(2,-4,2) && L(1,2,2)=(-5,-5,10)\\
&L(2,1,1)=(3,-2,-1) && L(2,1,2)=(-6,-6,12)\\
&L(2,2,1)=(2,2,-4) && L(2,2,2)=(-2,3,-1)
\end{aligned} \tag{9.83}$$

有两个确定的 Nash 平衡点，它们所产生的代价为(2,−4,2)和(3,−2,−1)。它们可以用式（9.79）进行验证。在给定其他局中人所选择的行动时，每个局中人都对输出感到满意。遗憾的是，这两个 Nash 平衡点都是可允许的。因此在局中人之间需要进行某种协作来确保不产生遗憾。■

9.5　对决策理论的进一步思考

本章介绍了大量的决策模型。它们为存在不确定性时的规划提供了基础，这是第III部分的核心。在构建以此为基础的规划模型以前，评估它对应用的适宜性是至关重要的。当阅读第 9.1 节～第 9.4 节的内容时，读者可能已经产生了许多疑问。如何确定代价？为什么我们应该相信对期望代价进行优化是正确的事情？如果先验概率分布不可用，那么会发生什么？最坏情况分析是不是过于保守？我们是否能够保证对策中的局中人将遵从所假设的合理行为？局中人知道其他局中人的代价函数，这是否现实？本节的目的就是从某种程度

上明确这些问题。一座大厦至多只能与它的地基一样牢靠。对决策论局限性的误解所产生的任何错误都将会影响规划表述的构建。

9.5.1 效用理论与合理性

本节将提出使用代价函数的理由，然后最小化表述 9.3 和表述 9.4 中代价的期望值。所产生的框架称为效用理论，它通常用收益代替代价。如第 9.1.1 节所述，代价乘以 -1，然后求最大值和求最小值交换顺序，这样就可以将代价转换为收益。因此我们谈论收益 R，直觉地认为更高的收益更好。

9.5.1.1 比较收益

假设给决策过程的不同输出赋予收益值。在某些应用中，数值可能是更自然的。例如收益可能是金融投资中所挣的钱的数量。在机器人应用中，可以将执行任务的时间或者所消耗的能量进行求反。例如收益可以是移动机器人构造一个地图之后所剩余的电池电量。

在某些应用中收益可能具有主观性。例如与打扫地板相比而言，洗盘子的收益是什么？对此每个人可能有不同的收益，并且每天都可能发生改变。这可能基于它们所处的环境或者在执行任务过程中的境遇、每个任务所花费的时间、对其他局中人的了解等。如果用决策论来自动进行“委托人”的决策过程，那么最好与委托人仔细商量来保证你知道他们的偏爱（选择优先级）。在这种情形下，应对它们的优先级进行排序，然后按顺序分配收益值。

一旦分配了收益值，应考虑表述 9.1 下的决策，该决策不涉及大自然。每个输出结果直接对应于一个行动 $u \in U$。如果 $R: U \to \mathbb{R}$ 给出收益，那么对于每个 $u \in U$，代价 L 可以定义为 $L(u) = -R(u)$。使委托人满意就是选择 u 最小化 L。

现在考虑与大自然的对策。现在的决策涉及在输出结果上进行概率分布的比较。所有概率分布的空间可能是巨大的，但是通过用期望将每个概率分布（或者密度）映射为一个真实值可以对之进行简化。我们所关心的是这种分布到实数的映射是否真的能够反映委托人的真实偏爱。下面的例子将阐述它的影响。

例 9.22（你喜欢赌博吗？） 假设给你 3 个选择：

（1）你能得到 1000 欧元。

（2）我们投掷一枚硬币，如果正面朝上，那么你可以得到 2000 欧元；否则你什么也得不到。

（3）你得到 3000 欧元的概率为 2/3；失去 3000 欧元的概率为 1/3。

每个选择的期望收益是 1000 欧元，但是你真的认为这些选择是等价的吗？因为期望值并没有考虑到你喜不喜欢赌博。在与大自然的对策中应该如何考虑这种问题？ ■

开始解决这个问题时，考虑另外一个场景是有帮助的。一些人可能偏向有更多钱。但是有趣的是，变得富有会降低对钱的价值的感知。下一个例子将阐述这个问题。

例 9.23（电视） 假设你幸运地出现在一个非常受欢迎的电视节目中。目的是测试你为了挣到一些钱而愿意欺骗你自己还是折磨你自己。你被要求做一些令人讨厌的事（如吃蟑螂或者将你的头没入水中很长时间等）。令 u_1 表示同意做这些事情的行动，u_2 表示你拒绝做这些事情。奖金是美元。假设你是一个贫困的学生。

下面是几个可能在电视节目中出现的场景。考虑你对每个场景的反应。

（1）假设 u_1 使你挣到 1 美元，u_2 则什么也挣不到。纯粹地对收益进行优化应该导致你选择 u_1，即完成讨厌的任务。但是这是 1 美元的价值吗？到现在我们还没有考虑在过程进行中对不舒服程度进行量化。构造一个收益函数，它减去你几乎不愿意完成任务的那部分美元，这是有意义的。

（2）假设 u_1 使你挣到 10 000 美元，u_2 则什么也挣不到。10 000 美元是个巨大的数字，值得忍受电视节目中的任何痛苦。因此选择 u_1 是可取的。

（3）现在假设电视节目主持人首先给你 10 000 000 美元作为你的出场费。那么你还愿意完成讨厌的任务来挣 10 000 美元吗？可能不会。这里发生了什么？当你得到更多钱时，你对钱的价值的感觉就会下降，是这样吗？如果参与者都是富有的石油公司总经理，那么节目会不太令人兴趣以至于没有人看。

（4）假设你已经完成任务即将赢得奖金。只是为了增加戏剧性，主持人给你一次赌博的机会。你选择 u_1 可以得到 10 000 美元；选择 u_2，通过投掷硬币你有 1/2 的概率赢得 25 000 美元。按照期望收益，应该选择 u_2。但是你刚完成了令人讨厌的任务而希望赢得奖金。失去全部奖金的风险是无法忍受的。在这种情况下，不同的人将有不同的选择。

（5）现在再次假设你完成了任务。如果选择 u_1 可以得到 100 美元；选择 u_2，通过投掷硬币你有 1/2 的概率赢得 250 美元。主持人非常善良，他让你进行 100 次选择。在这种情况下，两种行动的期望总数分别是 10 000 美元和 12 500 美元。这一次最好的选择应该是进行赌博。100 次独立的试验以后，我们预料赢得超过 10 000 美元奖金的概率非常高。因此如果允许我们进行大量的独立试验，期望情况分析似乎是有效的。在这种情况下，赢得接近期望收益的概率很高。

■

基于这个例子，决策系统的委托人或者评估人必须指出对输出结果的概率分布的偏爱。这是一个非常正式的方法，保证一旦分配了这些优先级，那么所设计的代价函数的期望如实地反映了对分布的优先级。这就产生了效用理论（utility theory），它涉及如下步骤：

（1）要求委托人在分配优先级时是理性的（rational）。这个概念由公理来定义。

（2）如果分配优先级的方法与公理相一致，那么一定存在效用函数。当优化期望效用时，这些优先级与委托人的优先级是匹配的。

（3）代价函数可以从效用函数中产生。

委托人必须在输出结果的概率分布中指定优先级。假设采用表述 9.2。为方便起见，假设 U 和 Θ 是有限的。令 X 表示基于输出的状态空间[1]。令 $f: U \times \Theta \to X$ 表示一个映射，它给每个输出结果赋一个状态。一个简单的例子是声明 $X = U \times \Theta$ 且使 f 为恒等映射。这使得输出结果的空间和状态空间相一致。用 f 将输出结果的空间缩为一个小的集合。如果两个结果用 f 可以映射到相同的状态，那么就所关心的收益或者代价而言，输出结果没有差别。

令 z 表示 X 上的一个概率分布，Z 表示 X 上的所有概率分布的集合。每个 $z \in Z$ 表示概率的一个 n 维向量，其中 $n = |X|$；因此它可以作为 $\mathbb{R}^n$ 的一个元素。这使得很容易对两个概

1　在多数效用论文献中，这被称为收益空间 $\mathcal{R}$ [91]。

率分布进行“混合”。例如令$\alpha\in(0,1)$为一个常数，令z_1和z_2为任意两个概率分布。采用标量乘法可以产生一个新的概率分布$\alpha z_1+(1-\alpha)z_2$，它是对z_1和z_2的混合。这很方便，因为不需要对结果进行归一化。假设z_1和z_2开始时值为1，混合后的值为$\alpha+(1-\alpha)=1$。

对决策过程进行建模的人必须与委托人进行协商以表示对Z中元素的偏爱。令$z_1\prec z_2$表示z_2优先于z_1。令$z_1\approx z_2$表示在优先级上是等价的。令$z_1\preceq z_2$表示$z_1\prec z_2$或者$z_1\approx z_2$。例9.24将阐述所分配的优先级。

例9.24（指示优先级的选择） 假设$U=\Theta=\{1,2\}$，这样将产生4种可能的输出：(1,1)、(1,2)、(2,1)和(2,2)。假设大自然代表一个根据概率分布产生1或者2的机器。行动是猜测机器产生的数。如果猜中，那么你将赢得那个数目的金币。如果没猜中，那么你不会赢得什么，也不会失去什么。

考虑用f构建状态空间X。结果(1,2)和(2,1)关于任何可能的收益来说是相同的。因此它们映射到相同的状态。另外两个输出是不同的。状态空间因此仅需要3个元素，可以定义为$X=\{0,1,2\}$。令$f(2,1)=f(1,2)=0$，$f(1,1)=1$，$f(2,2)=2$。因此最后两个状态代表会挣到的金币数。

现在考虑X上的概率分布的集合Z。每个$z\in Z$是一个三维向量。例如$z_1=[1/2\ \ 1/4\ \ 1/4]$代表状态为0的概率为1/2，状态为1的概率为1/4，状态为2的概率为1/4。假设$z_2=[1/3\ \ 1/3\ \ 1/3]$，那么你偏向选择哪个分布？在此情况下，z_2似乎一致优于z_1，因为有更大的机会赢得金币。因此我们声明$z_1\prec z_2$。分布$z_3=[1\ 0\ 0]$似乎是最坏的情况。因此我们可以宣布$z_3\prec z_1$和$z_1\prec z_2$。

对复杂的问题来说，确定优先级的过程可能变得十分乏味。在当前例子中，Z是$\mathbb{R}^3$的一个二维子集。该子集可以划分为区域的一个有限集合，在该区域上委托人能够清楚地指示其偏爱。该框架的一个不足之处是在Z上确定优先级是不切实际的[834]。

确定优先级后，是否能够保证X上存在实值函数，它的期望值能否精确反映优先级？如果分配的优先级满足合理性公理，那么答案是“是”。这些公理将在下面进行讨论。 ■

9.5.1.2 合理性公理

为了设计效用函数，优先级必须遵守合理性公理。它们是关于优先级中一致性的论述。只要遵守这些公理，那么效用函数一定存在（详见文献[270,834]）。决策者被称为是理性的，如果定义$\prec$和$\approx$以后，遵守如下公理[1]：

1. 如果$z_1,z_2\in Z$，那么有$z_1\preceq z_2$或者$z_2\preceq z_1$。（“你必须能够下决心。”）
2. 如果$z_1\preceq z_2$和$z_2\preceq z_3$，那么$z_1\preceq z_3$。（“优先级必须是可传递的。”）
3. 如果$z_1\prec z_2$，那么对于任意$z_3\in Z$以及$\alpha\in(0,1)$，有：

$$\alpha z_1+(1-\alpha)z_3\prec\alpha z_2+(1-\alpha)z_3 \tag{9.84}$$

（“混合而成的新的分布不改变优先级。”）

4. 如果$z_1\prec z_2\prec z_3$，那么存在某$\alpha\in(0,1)$和$\beta\in(0,1)$，使得：

$$\alpha z_1+(1-\alpha)z_3\prec z_2 \tag{9.85}$$

$$z_2\prec\beta z_1+(1-\beta)z_3 \tag{9.86}$$

1 存在选择公理系统[270,841]。

（“既没有天堂，也没有地狱。”）

每个公理都有一个具有实际意义的直观解释。第一个公理指出：总能够为两个分布推断优先级的方向。第二个公理指出优先级必须是可传递的[1]。最后两个公理有些复杂。在第三个公理中，z_2 严格优先于 z_1。通过混合到第三个分布 z_3，目的是为引起混乱。如果 z_3 的相同“数量”混合到 z_1 和 z_2，那么选择不应该被改变。最后一个公理涉及 z_1、z_2 和 z_3，它们每个都严格优于前面一个。第一个方程（9.85）指出：如果 z_2 严格优先于 z_1，那么少量 z_3 可以混合到 z_1 后，z_2 仍然是优先的。如果 z_3 已经像“天堂”（例如无穷多的收益），那么这就不可能了。式（9.86）指出：少量 z_1 可以混合到 z_3，结果依然比 z_2 好。即 z_1 不可能是收益为负无穷[2]的“地狱”。

9.5.1.3　构建效用函数

如果用与公理相一致的方式确定了优先级，那么效用函数总是存在的。这意味着存在一个函数 $\mathcal{U}: X \to \mathbb{R}$ 使得对于所有 $z_1, z_2 \in Z$，有：

$$z_1 \prec z_2 \quad \text{当且仅当} \quad E_{z_1}[\mathcal{U}] < E_{z_2}[\mathcal{U}] \tag{9.87}$$

其中 E_{z_i} 表示 $\mathcal{U}$ 的期望值，它可以作为概率分布 z_i 的一个随机变量来对待。$\mathcal{U}$ 的存在意味着可以通过最大化期望效用来确定最好的行动。

可以用效用函数 $\mathcal{U}$ 来定义收益函数，如 $R(u,\theta)=\mathcal{U}(f(u,\theta))$。效用函数可以转换为一个代价函数 $L(u,\theta)=-R(u,\theta)=-\mathcal{U}(f(u,\theta))$。在假设构建 $\mathcal{U}$ 来保护优先级的情况下，表述 9.3 和表述 9.4 所推荐的最小化期望代价似乎是有道理的。

非常遗憾，效用函数的存在性并不产生构建它的系统方法。在多数环境下，不得不用试错过程来设计 $\mathcal{U}$，这涉及重复检验所选择的优先级。在大多数应用中，创造效用函数和代价函数并未考虑本节讨论的含义。实际情况下可能会得到不希望的结论。因此重要的是不能对最优决策规则的品质过于自信。

如果采用最坏情况分析，那么这里讨论的大多数问题都可以避免。但是最坏情况分析有它自己的不足之处，将在第 9.5.3 节对其进行讨论。

例 9.25（金钱的效用） 通过描述一个经常应用于金钱的效用函数，我们给出本节的结论。假设状态空间 $X=\mathbb{R}$，它对应于所挣的美元数。金钱的效用指出：当总的财富增加时，金钱新增加的值降低了。效用函数可能是有界的。假设有某最大数量的美元，诸如 10^{100} 美元，那么另外的钱就没有价值了。一个典型的效用曲线如图 9.6 所示[91]。 ■

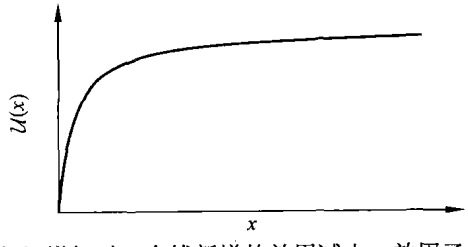


图 9.6　当总的财富积累增加时，金钱新增的效用减小，效用函数甚至可能是有界的

1　对于某些合理的问题，并不想要这种传递性。见文献[833]中的 Candorcet 和 Simpson 悖论。

2　某些公理系统允许无限收益，这导致有无穷大值的效用和代价函数，但是这里不考虑这种情况。

9.5.2 关于概率模型的问题

第 9.5.1 节讨论代价函数的来源和它们的期望的有效性。本节提出在第 9.2 节中使用的概率分布有效性问题。两个主要内容是对一般情况下贝叶斯方法的评论和构建概率分布的问题。

9.5.2.1 贝叶斯论与频率论

在过去的一个半世纪里，在统计学家中已经产生了关于概率的意义的辩论。实际上每个人对概率性质已经非常满足了，但是在此之外，当进行推理时它们的意义何在？两个主要阵营是频率论和贝叶斯论。一种形式的贝叶斯准则出版于 1763 年[82]。在 19 世纪的大多数时间里，贝叶斯分析在文献中几乎处于支配地位；但是在 20 世纪，频率论哲学作为一种更严密的概率解释而被更多人所接受。在最近几年，贝叶斯方法又重新升温。

众所周知，贝叶斯定理将概率阐述为对假设的置信程度。在这种原理下，从一个先验分布开始、收集许多观测，然后应用贝叶斯准则所产生的后验分布来进行决策，这样做是完全有效的。

从频率论的观点，贝叶斯分析对概率的使用过于自由。频率论者相信概率仅被定义为独立试验的次数趋近于无穷时的极限状况下得到的量。例如，如果多次投掷一枚硬币，1/2 概率表示当试验次数趋近无穷时，正面朝上的次数与总试验次数的比例将收敛到的值。另一方面，贝叶斯论可能会说下一次试验中正面朝上的概率为 1/2。对频率论来说，对概率的这种阐述太强大了。

研究人员已经研究了基于频率论基本原理的决策论；两者之比较出现在文献[834]中。作为一个例子，频率论提倡优化如下频率风险（frequentist rish）来得到决策：

$$R(\theta,\pi)=\int_{y}L(\pi(y),\theta)P(y\mid\theta)\mathrm{d}y \tag{9.88}$$

其中 π 表示策略 $\pi:Y\rightarrow U$。频率风险在所有数据上进行平均，而不是像式（9.26）中贝叶斯论所倡导的那样在单个观测的基础上进行决策。假设从独立数据试验次数趋近无穷的极限中得到概率 $P(y\mid\theta)$。使用式（9.88）的主要不足之处在于优化取决于 θ。最好的决策规则必须依赖于未知的 θ。在某些极限状况下，可能选择某 π 来针对所有 θ 最优化的式（9.88），但是这很少发生。因此频率风险可以看作是对策略的一个约束。但是它通常并不足以用来产生策略。这个问题是对第 9.1.1 节讨论的 Pareto 优化的回顾。频率论方法试图希望更加保守和严格，其结果是产生了关于决策的更弱的论述。

9.5.2.2 先验分布的来源

假设采用贝叶斯方法。在所有贝叶斯分析中最主要的问题是先验分布的来源。在第 9.2 节这用 $P(\theta)$ 表示（或者 $p(\theta)$），它表示大自然行动空间上的一个分布（或者密度）。得到 $P(\theta)$ 最好的方法是在大量独立试验上对分布进行估计。这使得它的定义与频率论观点相一致。这对例 9.11 来说是可能的，其中 $P(\theta)$ 可以通过大量文本中字母出现的次数来估计。分布甚至可以适应性修改到针对特定的语言和主题。

但是在多数使用决策论的应用中，进行这种试验是不可能的或者非常昂贵的。在这种情况下应该怎样做？如果先验概率只是“拼凑的”，那么所产生的后验概率可能是值得怀疑的。实际上根本不能称之为概率。有时在这种情况下会用到主观概率(subjective probabilities)这个术语。通常也都是这样做的，因为没有多少其他选择。选择之一是借助于频率论决策理论，但是就像所提及的那样，它在单个观测情况下不能工作。

非常幸运，当观测的数量增加时，先验分布对后验分布的影响就会消失。如果只有一个观测，或者甚至像表述 9.3 那样没有观测，那么先验分布的影响非常大。如果有少量或者没有关于 $P(\theta)$ 的信息，那么应该尽可能仔细地设计分布。也应该理解到无论这种假设产生什么结论，先验分布都使它们是有偏的。假设这个模型被用作规划方法的基础。你可能满足于计算“最优”规划，但是由于先验分布值的缘故，最优性依赖于某任意的初始偏差。

如果没有可用信息，那么 $P(\theta)$ 在 Θ 上应该尽可能均匀，这是合理的。这被拉普拉斯称为“不充分理由原理”[584]。如果没有理由相信一个元素比另外一个的可能性更大，那么它们应该分配相等的概率值。用信息论中的信息熵可以验证这种合理性[50,250,866]。在离散情况下，信息熵是：

$$-\sum_{\theta\in\Theta} P(\theta)\lg P(\theta) \tag{9.89}$$

在连续情况下，信息熵是：

$$-\int_{\Theta} p(\theta)\lg p(\theta)\mathrm{d}\theta \tag{9.90}$$

这种熵测度在通信系统中用来对通过有噪声的介质传递的信息进行解码所需要的最小比特数进行估计。一般情况下它代表与分布有关的不确定性的数量。熵的一个大值表示大量的不确定性。

当 $P(\theta)$ 是均匀分布时，熵函数最大化，这似乎验证了不充分理由原理的合理性。这可以被认为是一个非信息的先验信息。当 $\Theta=\mathbb{R}$ 时，该思路的应用十分频繁，这会导致一个不正确的先验信息。在所有 $\mathbb{R}$ 上密度函数不能保持非零常数值，因为它的积分是无穷大的。因为第 9.2 节中的决策不依赖于任何归一化因素，所以可以给 $p(\theta)$ 赋一个常数值，并且决策不会受到先验信息不正确性的影响。

在先验信息的选择中应用熵的主要困难在于 Θ 本身的选择具有随意性。均匀分配不同的 Θ 最终会产生不同的先验信息。考虑下面的例子。

例 9.26（带非信息先验信息的问题） 假设基于天气来决定采取什么行动。假设完全没有关于天气的任何信息。一种可能是用 $\Theta=\{p,c\}$，其中 p 表示“降水”，c 表示“晴朗”。如果想最大化式（9.89），那么应该分配 $P(p)=P(c)=1/2$。

经过仔细思考以后，我们可能愿意区分不同类型的降水。大自然行动的一个更好的集合是 $\Theta=\{r,s,c\}$，其中 c 仍然表示“晴朗”，但是降雨 p 分为“下雨” r 和“下雪” s。现在给每个大自然行动指派概率 1/3 来最大化式（9.89）。这与原来的情况有明显不同。现在我们区分不同类型的降水，似乎降水更可能发生。我们喜欢区分不同形式的降水，然而这确实能够影响天气吗？ ■

例 9.27（连续空间的非信息先验信息） 相似的问题也发生在连续空间。回顾例 9.12 所描述的参数估计问题。假设任务是基于某些数据点来估计一条线，假设这些点落在这条线上，但是由于测量过程中的噪声使得我们丢失了这些点。

给所有直线的集合Θ赋予什么样的初始概率密度？假设直线在$Z=\mathbb{R}^2$中。直线方程可以表示为：

$$\theta_1 z_1 + \theta_2 z_2 + \theta_3 = 0 \tag{9.91}$$

问题是如果参数向量$\theta=[\theta_1 \quad \theta_2 \quad \theta_3]$与一个标量常数相乘，那么可以得到相同的直线。因此即使$\theta\in\mathbb{R}^3$，必须增加一个约束。假设我们要求：

$$\theta_1^2 + \theta_2^2 + \theta_3^1 = 1 \tag{9.92}$$

且$\theta_1 \geqslant 0$。这几乎解决了该问题，并且保证每个参数值对应于唯一一条直线（除了在$\theta_1=0$处完全相同的情况，但是这里可以忽略它）。因此参数空间是一个球$\mathbb{S}^2$的上半部。最大熵先验信息建议给Θ分配一个均匀分布密度。从直观感觉上也应该这样，但是应用于参数空间来保证唯一性的特别约束却使得均匀性是有偏的。还有许多其他选择。例如我们可以通过迫使点落在方体表面的上半部这种约束来代替式（9.92），而不是落在一个球的上半部。在这个新的参数空间中分配的均匀概率密度不同于球上的概率密度。

在某些情形下，参数空间有一个自然的表示，它对特定变换是不变的。第5.1.4节介绍了Haar测度的概念。如果将Haar测度当作非信息先验信息来使用，那么可以得到一个关于均匀性的概念。例如，假设参数空间是$SO(3)$。例5.14中的单位四元数空间上的均匀概率质量就是一个非常好的非信息先验信息，因为它与Haar测度一致，而Haar测度对于应用于事件的群运算来说是不变的。非常遗憾，对多数实际应用中出现的空间来说不存在Haar测度[1]。 ■

9.5.2.3 关于条件分布的不正确假设

最后一个问题是分布经常被不正确地估计，因为人们可能任意假设它属于某一个分布族。例如高斯分布通常比较方便。因此总假设$p(y|\theta)$是高斯分布。可通过试验来估计参数的均值和方差。即使发现高斯分布具有某种最合适的品质，也不一定说明它是一个好的表示方法。基于这种模型所得到的结论可能是不正确的，特别是在真实的分布有不同的形状（诸如有一个大的尾巴或者是多模的）更是如此。在许多情况下，可能需要采取非参数方法来避免这种偏差。这种方法并不假设特定的分布族。例如，假设用记录在某固定θ下y发生次数的直方图来估计概率分布。这种直方图归一化得到概率分布的一种表示，而不需要对初始形状进行假设。

9.5.3 关于非确定模型的问题

令人感兴趣的是抛弃概率模型而采用非确定模型。这仅要求指定Θ，而不用指定关于不同行动的相关可能性的任何信息。那么第9.5.1节和第9.5.2节提出的大多数复杂问题就不存在了。非常遗憾，这付出的代价也很大。在非确定模型下通过最坏情况分析所进行的决策也有其不足之处。进行权衡折中以后，才能确定最适合特定应用的模型。

第一个问题是很难保证Θ足够大以覆盖所有可能性。考虑表述9.6，其中大自然行动两次。通过大自然观测行动空间$\Psi(\theta)$，在测量过程中产生干扰。假设$\Theta=\mathbb{R}$和

1 要求一个局部紧致拓扑群[348,838]。

$h(\theta,\psi)=\theta+\psi$。在这种情况下 $\Psi(\theta)$ 可以被理解为测量误差。可能发生的最大误差是多少？假设一个声纳正在测量机器人到墙的距离。基于传感器的性能指标，可以确定误差的界。但是误差偶尔可能比这个界大。声纳有时没有收到计算距离所需要的回声。在这种情况下所报告的距离为 ∞。由于反射，有时的误差可能非常大。尽管这种大的误差可能并不会频繁发生，但是如果我们想要保证性能，那么大的或者无穷大的误差也应该包含在 $\Psi(\theta)$ 中。问题是最坏情况分析可能总是认为传感器无效，因为传感器报告 ∞。应该忽略从传感器得到的任何统计有效的信息。在概率模型下，很容易产生非常大的 $\Psi(\theta)$，并且给大误差赋予非常小的概率。带有非确定不确定性的问题是需要 $\Psi(\theta)$ 比较小来产生合适的决策；但是理论上所“保证”的性能实际上并不一定能够真正得到保证。

一旦表述了非确定的模型，最优决策产生的结果对特定应用来说可能是非常荒谬的。问题是决策者不能容忍任何风险。只在结果可以得到保证的情况下才会采取行动，并没有希望比最坏情况下做得更好。考虑下面的例子。

例 9.28（保守决策问题） 假设一个朋友给你选择 1000 欧元支票或者 1 欧元现金。对于支票，你必须去银行兑现，但是有小的可能性——你朋友的账户上没有足够金额。如果是这样，你将什么也得不到。如果你选择 1 欧元现金，那么你肯定会得到。

如下代价矩阵反映输出结果（忽略效用理论）：

Θ \ U	1	1000
	1	0

（9.93）

采用概率分析，我们做出结论：最好选择支票。你可能知道朋友非常富有并且对银行的账户很负责。但是不能在决策过程中考虑这些信息。采用最坏情况分析，最优行动应该是选择 1 欧元现金。尽管你可能觉得它不太好。如果你后来知道账户上有足够资金兑现 1000 欧元，那该有多后悔。 ■

因此想要绝对保证结果是需要付出代价的，记住这一点是非常重要的。概率模型提供了融合统计信息的灵活性。有时可将概率模型看作是非确定模型的一般形式。如果假设大自然在机器人之后行动，那么大自然行动可以将这考虑进来，就像表述 9.4 一样。在非确定情况下，指定的是 $\Theta(u)$；在概率情况下，指定的是 $P(\theta|u)$。可以设计分布 $P(\theta|u)$ 使得大自然用很高的概率选择 $\theta\in\Theta$ 来最大化 $L(u,\theta)$。在例 9.28 中，这意味着支票遭银行退票（导致无收益）的概率非常高，如 0.999 999。在这种情况下，甚至概率模型下的最优行动也会选择 1 欧元现金。对于用最坏情况分析建模的任何实际决策问题来说，决策过程能够向后进行来生成可能的先验信息，对这些先验信息来说，采用概率分析也将得到相同的决策。在例 9.28 中，决策似乎是基于假设支票遭退票的概率很高，即使没有概率模型。

这意味着可将非确定模型下最坏情况分析看作是概率模型的一个特殊形式，其中先验分布给最坏输出分配了很高的概率。就像贝叶斯分析中其他先验概率受到置疑一样，非确定模型下最坏情况分析的合理性也受到置疑。那么这种指定特定概率的基础是什么？

9.5.4 关于对策论的有关问题

对策论的一个最基本的局限性是每个局中人必须知道其他局中人的代价函数。如在第9.5.1节中，对于单个决策者来说确定一个合适的代价函数是十分困难的。那么确定其他局中人的代价函数和意图就更加困难。在大多数实际应用中这种信息是无法得到的。一种可能性是对与其他局中人代价函数有关的不确定性进行建模。基于对局中人所采取行动的观测，可以用贝叶斯分析来思考代价函数。又一次出现指定先验信息的问题。在代价函数中允许存在不确定性的最大困难在于会发生一种“无穷反映”[394]。例如，如果我们正在进行对策，其他局中人知道我的代价函数吗？我不能确定，并且其他决策者是否知道我并不完全知道它们的代价函数呢？这种交替猜测可以无限进行下去，这导致先验分布的嵌套推理和分配[1]。

用随机策略可以保证鞍点或者Nash平衡点的存在。从数学上讲，这是对棘手问题的一个清晰解；但是它也代表对模型的一个巨大改变。许多对策只进行一次。对于期望情况下收敛的结果，对策则必须进行无穷多次。如果对策只进行一次或者一些次，那么局中人很可能会产生遗憾，即使基于期望情况分析的理论认为遗憾是可以消除的。

另外一个问题是智能的人在数次对策之后可能会改变他们的策略。对人类来说模拟一个随机策略是很困难的（假设他们想这样，也是不可能的）。甚至有这样的国际比赛，其中局中人重复进行诸如Rock-Paper-Scissors或者囚犯的艰难抉择这种经典对策。人们设计囚犯进行抉择的比赛程序，对这种比赛的一个有趣的讨论见文献[918]。可以观察到：多次迭代之后会发生某种合作，这对两个决策者都可以确保更大的收益，即使他们不能进行通信。在这种情形下产生了一个策略，称为Tit-for-Tat，其中每个阶段重复其他局中人在上一个阶段所选择的行动。该方法简单而且非常成功。

在非零和对策中，多个Nash平衡点的存在是非常棘手的。假设在多个Nash平衡点中仅有一个是可允许的。那么一个对手将会仔细思考不同的Nash平衡点而选择一个可允许的吗？可能一些局中人是比较保守的而采取安全策略，这样就完全破坏了使遗憾最小化的假设。如果有多个可允许的Nash平衡点，那么遗憾是不可避免的，除非在局中人之间有某种合作。如果这种合作是不可能的，那么结果并不好。

进一步阅读

第9.1节介绍的基本概念在许多书中和网络上都能够找到。更多关于Pareto最优性的内容，见文献[849,910,953,1005]。第9.2节的内容主要是受到决策论的启发和影响。文献[91]是一本非常好的入门书，其他文献还有[270,273,676,834]。“与大自然之间的对策”这种观点主要是基于文献[111]。模式分类是决策论的一个重要形式，它在文献[19,273,297,714]中都有介绍。贝叶斯网络[781]是人工智能研究中的一种非常普通的表示方法，它通常为复杂决策问题提供信息的紧凑编码。

在许多书中都可以找到（例如文献[60,762]）更多关于第9.3节和第9.4节对策论概念

1 熟悉电影 *The Princess Bride* 的读者可能还记得Vizzini和Roberts之间关于那个酒杯中含有毒药的富有幽默感的对话。

的内容。文献[918]是一本非常有趣的书籍，它包含许多例子和直观知识。对于有无限行动集合的对策，参见文献[60]。随机化的 Nash 平衡点的计算仍然是一个非常活跃的研究领域。文献[693]对这些方法进行了概述；也可参见文献[548,702]。出现在计算随机 Nash 平衡点中的多项式方程组似乎提示到可以应用计算代数几何中的算法，就像第 6.4 节的组合运动规划中用来解决这类问题所需要的那样。文献[263]给出了一种采用这种工具的方法。与第 9.4 节中定义的非合作对策相比，合作对策论则研究了多种方法，各个局中人用这些方法可以形成联盟来提高它们的收益[782]。

第 9.5 节中的部分内容来自于文献[91]。多数决策论书籍（例如文献[91]）和一些人工智能书籍（例如文献[842]）中都有关于效用理论的内容。关于贝叶斯论与频率论之间的深度讨论，请参见文献[834]。完整介绍决策中构建代价模型的内容，请参见文献[542]。

习　　题

1. 假设定义一个单步两目标决策问题，其中有两个目标和一个连续行动集合 $U=[-10,10]$。代价向量是 $L=[u^2 \quad u-1]$。确定最优 Pareto 行动集合。

2. 令：

		Θ		
	−1	3	2	−1
U	−1	0	7	−1
	1	5	5	−2

为决策者和大自然所做出的每个选择组合而定义代价。令大自然的随机策略为[1/5 2/5 1/10 3/10]。

（a）用非确定的推理来发现最小最大决策和最坏情况下的代价。

（b）用概率推理来发现最好期望情况决策和期望代价。

3. 除了本章介绍的另外，可能还有许多合理的决策原则。

（a）习题 2（a）反映的是极端悲观情况。假设用极端乐观情况。选择能够最优化习题 2 中矩阵的最好情况代价。

（b）一种方法是用一个乐观指数 $\alpha \in [0,1]$，它能够在两种极端情形之间进行插值。因此通过最小化：

$$\alpha \cdot \max_{\theta \in \Theta} \{L(u,\theta)\} + (1-\alpha) \min_{\theta \in \Theta} \{L(u,\theta)\} \tag{9.94}$$

来选择一个决策 $u \in U$。确定所有可能的 $\alpha \in [0,1]$ 下的最优决策。你的答案以选择列表的形式表示，每个选择对应 α 的一个特定范围。

4. 假设进行决策以后，观察大自然的选择。那么将你的代价与如果给定大自然的选择后你的选择所能够得到的最好代价相比较会如何？这两种代价之间的差别被称为“遗憾”或者最小值“Doh!”[1]，心理学家已经指出，某些人进行决策就是基于最小化遗憾的。它反

1　在 2001 年，Homer Simpson 的“Doh”作为遗憾的一种表达而被加入到牛津英语词典。

映的是如果你采取其他行动的话结果是多么糟糕。

（a）表达最坏情况的遗憾，并用它进行最小最大化遗憾的决策，采用习题 2 中的矩阵。

（b）表达期望的遗憾，并用它进行最小最大化期望遗憾的决策。

5. 用习题 2 的矩阵，考虑大自然所有概率分布的集合。刻画所有分布的集合，对于这些分布来说，最小最大化决策和最好期望决策将产生相同的选择。这指示如何反证先验信息。

6. 考虑一个代价函数为 L 的贝叶斯决策论情形。显示对于任意 $a>0$ 和 $b\in\mathbb{R}$，如果 $L(u,\theta)$ 被 $aL(u,\theta)+b$ 代替，那么决策原则从来不会发生变化。

7. 假设有两个类型 $\Omega=\{\omega_1,\omega_2\}$，其中 $P(\omega_1)=P(\omega_2)=\frac{1}{2}$。观测空间 Y 是 $\mathbb{R}$。回顾概率论中的正态（或者高斯）概率密度函数：

$$p(y)=\frac{1}{\sigma\sqrt{2\pi}}e^{-(y-\mu)^2/2\sigma^2} \tag{9.95}$$

其中 μ 表示均值，σ^2 表示方差。假设 $p(y|\omega_1)$ 是一个正态密度函数，$\mu=0$，$\sigma^2=1$。假设 $p(y|\omega_2)$ 是一个正态密度函数，$\mu=6$，$\sigma^2=4$。寻找最优分类原则 $\gamma:Y\to\Omega$。欢迎你用数值方法（用计算机）或者绘图方法（仔细绘制函数图）解决该问题。解释在任意情况下你是如何得到该答案的。

8. 令：

	Θ			
U	2	−2	−2	1
	−1	−2	−2	6
	4	0	−3	4

为决策者和大自然所做出的每个选择组合而定义代价。令大自然的随机化策略为[1/4 1/2 1/8 1/8]。

（a）用非确定的推理来发现最小最大决策和最坏情况的代价。

（b）用概率推理来发现最好期望情况决策和期望代价。

（c）刻画所有概率分布的集合，对这些分布来说，最小最大决策和最好期望决策产生相同的选择。

9. 在一个常数和对策中，对某个不依赖于 u 和 v 的常数，对于任意 $u\in U$ 和 $v\in V$ 的代价之和产生：

$$L_1(u,v)+L_2(u,v)=c \tag{9.96}$$

显示任意常数和对策都可以变换为一个零和对策，它的鞍点解可以用零和情况的方法来解决。

10. 将例 9.7 描述为一个零和对策，为局中人计算安全策略。对策的期望值是多少？

11. 假设对于两个零和对策，存在某非零 $c\in\mathbb{R}$ 使得一个对策的代价矩阵是通过将另外一个代价矩阵的所有元素乘以 c 得到的。证明这两个对策一定有相同的确定鞍点和随机鞍点。

12. 按照习题 11 的方法，证明如果所有矩阵元素加上 c，那么两个零和对策有相同的

确定鞍点和随机鞍点。

13. 证明用矩阵 A 和 B 所指定的一个非零和对策的多个 Nash 平衡点是可互换的，如果对策(A,B)所产生的 Nash 平衡点与$(A,-A)$产生的 Nash 平衡点是相同的。

14. 分析 3 个局中人的 Rock-Paper-Scissor 对策。对每个局中人，代价 1 表示输，0 表示平局，−1 表示赢。指定代价函数，那么有可能避免遗憾吗？它有一个确定的 Nash 平衡点吗？你能够发现一个随机 Nash 平衡点吗？

15. 为如下零和对策，计算随机平衡点：

		V	
U		0	−1
		−1	2

（9.97）

指出局中人的随机策略及其所产生的对策期望值。

实现

16. 考虑估计一个未知参数值 $\theta\in\mathbb{R}$。先验概率密度是一个正态分布：

$$p(\theta)=\frac{1}{\sigma\sqrt{2\pi}}e^{-(\theta-\mu)^2/2\sigma^2} \tag{9.98}$$

其中 $\mu=0$，$\sigma=4$。假设产生一个观测序列 $y_1,y_2,\cdots,y_k$，并且每个 $p(y_i|\theta)$ 都是 $\mu=\theta$ 和 $\sigma=9$ 的一个正态分布密度。假设 u 表示你对参数值的猜测。任务是选择 u 来最小化代价 $L(u,\theta)=(u-\theta)^2$ 的期望。假设 θ 的“真实”值为 4。对于每个 $i\in\{1,\cdots,k\}$，观测到 $y_i=4$ 以后，确定关于期望代价的最小行动 u^*。

（a）对 $k=1$，确定 u^*。

（b）对 $k=10$，确定 u^*。

（c）对 $k=1000$，确定 u^*。

该试验并不很现实，因为观测应该由对正态密度函数 $p(y_i|\theta)$ 的采样来生成。用正态分布所得到的值来代替每个 k 的 $y_k=4$，重做习题。

17. 实现计算零和对策的一个随机鞍点的算法。假设一个局中人的行动不超过两个，而另外一个局中人可以有任意有限数量的行动。

18. 假设用多目标来定义一个 K 阶段决策问题。有一个有限状态空间 X，每个 $x\in X$ 都有一个有限行动集合 $U(x)$。一个状态转移方程 $x_{k+1}=f(x_k,u_k)$ 从目前状态和输入中给出下一个状态。有 N 个如下形式的代价泛函：

$$L_i(u_1,\cdots,u_K)=\sum_{k=1}^{K}l(x_k,u_k)+l_F(x_F) \tag{9.99}$$

其中 $F=K+1$。如果 $x_F\in X_{goal}$（对于某些目标区域 $X_{goal}\in X$），那么 $l_F(x_F)=\infty$；否则 $l_F(x_F)=0$。假设没有终止行动（这样能够简化问题）。设计一个值迭代方法，它能够尽可能有效地发现最优 Pareto 规划的完整集合。如果两个或两个以上规划产生相同的代价向量，那么只需要返回其中一个表示。

第 10 章　序贯决策理论

第 9 章讨论的是在存在不确定性的情况下如何进行单个决策，从本质上第 9 章是对规划的突破。本章我们通过表述一个决策序列来进行规划。这通过将第 2 章的离散规划的概念扩展到融合多个决策者的影响。最重要的新决策者是大自然，在规划的执行过程中它引起不可预测的结果。状态空间和状态转移方程在本章不断出现；但是与第 2 章不同，在这里额外的决策者影响状态转移。作为这种影响所产生的后果，规划需要加入状态反馈，这使得规划可以根据目前的状态来选择相应的行动。当确定规划以后，没有人知道将来的状态。因此需要进行反馈，这与第 2 章中指定规划作为一个行动序列有所不同。第 2 章的规划只在行动是可以预测的情况下才成为可能。

本章假设目前的状态是已知的。仅存的不确定性是关于对将来状态的预测的不确定性。第 11 章和第 12 章将讨论状态未知的情况。这要求定义感测模型来对状态进行测量。主要的结果是，规划是在信息空间中进行的，而不是在状态空间中。当预测的不确定性和目前状态的不确定性同时存在时，本章的大多数思路将扩展到信息空间。

本章考虑的问题有广泛的应用。大多数思路都是在随机控制论中发展起来的[95,567,570]。概念可用来建模移动机器人问题，因为将来的状态通常是不可预测的，但有时可用概率方法来建模[1004]或者用最坏情况分析[593]。在工程、经济等许多领域还有其他一些应用。例如过程控制、赌博策略、投资规划等。

这里通常主要关注于到达目标状态。非确定的和概率的不确定性都会予以考虑。在非确定的情况下，任务是发现保证正常运行的规划，不管大自然如何干扰。在某些情况下，能够计算出既能实现目标，又具有在最坏情况下的最优性能的规划。在概率情况下，任务是发现产生最优期望性能的规划。即使在执行单规划时输出结果是不可预测的，如果规划在相同问题中执行大量次数，那么概率情况下的思路是减小平均代价。

10.1　与大自然之间的序贯对策

本节将第 2 章的思路扩展到大自然干扰行动的输出这种情况。第 10.1.1 节定义的规划问题是对第 2.1 节内容的直接扩展。由于不可预测性，第 10.1.2 节将介绍前向投影和后向投影分别描述将来和过去可能的状态。前向投影描述应用规划或者一系列行动所得到的将来的状态。第 2 章不需要这个概念，因为将来的状态序列总是可从规划和初始状态获得。在第 10.1.3 节将定义规划，使用前向投影来说明每次应用规划时它的执行是多么不同。

10.1.1　模型定义

本节的表述是对表述 2.3 的扩展，它融合了每阶段（或者称每步）大自然的影响。令 X

表示一个离散状态空间，令 $U(x)$ 表示对决策者（或者机器人）来说从 $x \in X$ 开始的可用的行动的集合。在每个阶段 k，假设大自然的行动 θ_k 从集合 $\Theta(x_k,u_k)$ 中选择。这可以认为是将表述 9.4 推广到多步情况，它引入了 $\Theta(u)$。现在 Θ 除了依赖于行动以外可能还依赖于状态，因为在目前情形下 x_k 和 u_k 都是可用的。也就是说，大自然在行动时已经知道决策者所选择的行动。而总假设在第 k 阶段，决策者不知道大自然所采取的行动。但是对于所有 $x_k \in X$ 和 $u_k \in U(x_k)$，决策者知道集合 $\Theta(x_k,u_k)$。

如第 9.2 节所述，大自然有两个模型：非确定的和概率的。如果采用非确定模型，那么只知道大自然将从 $\Theta(x_k,u_k)$ 选择行动。在这种情况下，用最坏情况分析来进行决策是合适的。

如果采用概率模型，那么 $\Theta(x_k,u_k)$ 上的概率分布也将作为模型的一部分给出。在这种情况下最重要的假设是大自然是马尔可夫过程的。一般情况下，这意味着概率只依赖于局部信息。在大多数应用中，这种局部性是关于时间的。在这里的表述中，意味着 $\Theta(x_k,u_k)$ 上的概率分布只依赖于当前阶段所得到的信息。在其他情形下，马尔可夫过程意味着对少量阶段的依赖性，或者是按照空间关系的局部依赖性，就像在马尔可夫随机场中那样[233,379]。

为了使马尔可夫假设更加精确，再次应用第 8.2.1 节中定义的状态和行动历史。令：

$$\tilde{x}_k = (x_1, x_2, \cdots, x_k) \tag{10.1}$$

$$\tilde{u}_k = (u_1, u_2, \cdots, u_k) \tag{10.2}$$

它们表示到第 k 阶段为止所有可用的信息。如果没有马尔可夫假设，那么大自然的概率分布可能是所有 $\tilde{x}_k$ 和 $\tilde{u}_k$ 上的条件概率 $P(\theta_k \mid \tilde{x}_k, \tilde{u}_k)$。马尔可夫假设指出，对于所有 $\theta_k \in \Theta(x_k,u_k)$：

$$P(\theta_k \mid \tilde{x}_k, \tilde{u}_k) = P(\theta_k \mid x_k, u_k) \tag{10.3}$$

这就能够删去所有过去的状态和行动，而只留下目前的状态和行动。一旦知道目前的状态和行动，那么从历史信息中就不能够得到关于大自然的任何额外信息。

大自然的影响将在状态转移方程中被定义，在给定 x_k、u_k 和 θ_k 的情况下，方程产生一个新的状态 x_{k+1}：

$$x_{k+1} = f(x_k, u_k, \theta_k) \tag{10.4}$$

从决策者的角度来看，没有给定 θ_k。因此能够推断，应用 u_k 和 x_k 将产生状态的一个特别集合：

$$X_{k+1}(x_k,u_k) = \{x_{k+1} \in X \mid \exists \theta_k \in \Theta(x_k,u_k),\ x_{k+1} = f(x_k,u_k,\theta_k)\} \tag{10.5}$$

在式（10.5）中，符号 $X_{k+1}(x_k,u_k)$ 表示在给定 x_k 和 u_k 的情况下，可能的 x_{k+1} 值的集合。符号 $X_{k+1}(\cdot)$ 通常用来表示用出现在变量中的信息可能产生的 x_k 值。

在概率情况下，对 x_k 应用 u_k 可以为第 $k+1$ 阶段产生 X 上的概率分布。$P(\theta_k \mid x_k, u_k)$ 作为问题的一部分而给定。用状态转移方程 $x_{k+1} = f(x_k,u_k,\theta_k)$，可以得到：

$$P(x_{k+1} \mid x_k, u_k) = \sum_{\theta_k \in \Theta'} P(\theta_k \mid x_k, u_k) \tag{10.6}$$

其中：

$$\Theta' = \{\theta_k \in \Theta(x_k,u_k) \mid x_{k+1} = f(x_k,u_k,\theta_k)\} \tag{10.7}$$

$P(x_{k+1}\mid x_k,u_k)$的计算只涉及所有概率质量的积累，这些概率质量通过大自然的各种行动来产生x_{k+1}。

将这些模型合并在一起并且增加一些表述 2.3 中的组成部分就可以形成如下表述。

表述 10.1（带大自然的离散规划）

1. 非空状态空间X是状态的一个有限的或者可数无限集合。

2. 对于每个状态$x\in X$，一个有限非空行动空间$U(x)$。假设U包含特殊的终止行动，它与表述 2.3 中的终止行动相同。

3. 对于每个$x\in X$和$u\in U(x)$，有一个有限的非空大自然行动空间$\Theta(x,u)$。

4. 有一个状态转移函数f，对于每个$x\in X$、$u\in U(x)$和$\theta\in\Theta(x,u)$，它产生状态$f(x,u,\theta)$。

5. 用k表示的阶段集合从$k=1$开始并无限继续。也可能有一个固定的最大阶段值$k=K+1=F$。

6. 初始状态$x_I\in X$。对于某些问题，初始状态可能没有给定，在这种情况下规划解必须从所有初始状态来发现。

7. 一个目标集合$x_G\subset X$。

8. 随阶段相加的代价函数L。令$\tilde{\theta}_K$表示到阶段K为止的大自然历史行动。对于状态、行动以及大自然历史的任何组合，代价函数产生：

$$L(\tilde{x}_K,\tilde{u}_K,\tilde{\theta}_K)=\sum_{k=1}^{K}l(x_k,u_k,\theta_k)+l_F(x_F) \tag{10.8}$$

其中$F=K+1$。如果在某阶段k应用终止行动u_T，那么对于所有$i\geqslant k$，$u_i=u_T$，$x_i=x_k$且$l(x_i,u_T,\theta_i)=0$。

采用表述 10.1，可以定义可行规划和最优规划问题。为了得到可行规划问题，令对于所有$x_k\in X$，$u_k\in U$和$\theta_k\in\Theta_k(u_k)$，$l(x_k,u_k,\theta_k)=0$，并且令：

$$l_F(x_F)=\begin{cases}0 & x_F\in X_G\\ \infty & \text{其他情况}\end{cases} \tag{10.9}$$

为了得到最优规划问题，一般情况下，如果$x_k\notin X_G$，可以假设$l(x_k,u_k,\theta_k)$为任意非负的有限值。对于涉及概率不确定性的问题，有时为$x_k\notin X_G$的$l_F(x_F)$赋予一个很高的有限值是适宜的，这与为没有实现目标而赋予一个无穷代价的做法不同。

应注意到在每个阶段，一般情况下代价项与大自然行动θ_k有关。如果采用概率不确定性，那么表述 10.1 通常称为受控的马尔可夫过程或者马尔可夫决策过程(Markov decision process，MDP)。如果从表述中删去行动，那么可简单地称其为马尔可夫过程。在大多数关于统计的文献中，当过程是离散阶段时，可用马尔可夫链代替马尔可夫过程（这与连续马尔可夫过程不同）。因此采用受控马尔可夫链和马尔可夫决策链的概念是更可取的。

在某些应用中，则尽量避免明确地刻画大自然。假设$l(x_k,u_k,\theta_k)=l(x_k,u_k)$。如果采用非确定的不确定性，那么对于所有$x_k\in X$和$u_k\in U(x_k)$，可以用$X_{k+1}(x_k,u_k)$代替状态转移方程；这就避免了涉及大自然。对状态$x_k$应用行动$u_k$直接产生$X$的特定子集。如果采用概率不确定性，那么可以直接定义$P(x_{k+1}\mid x_k,u_k)$来代替状态转移方程。如果对某状态x_k应用行动u_k，仍然避免了涉及大自然，这就产生了X上的一个概率分布。多数时间里，我们使

用涉及大自然的状态转移方程，但记住这些可选择的方法是非常重要的。它们出现在很多相关书籍和研究文献中。

就像第 2 章所讲述的那样，一个有向状态转移图有时便于表达规划问题。在这里可能采用相同的思路。就像第 2.1 节所讲述的那样，X 表示顶点集合，但是必须改变边的定义以反映大自然的影响。从状态 x 到 x' 存在一条有向边，如果存在某 $u\in U(x)$ 和 $\theta\in\Theta(x,u)$ 使得 $x'=f(x,u,\theta)$。通过将代价项 $l(x_k,u_k,\theta_k)$ 与每条边相联系就可以产生一个加权图。在概率模型情况下，发生转移的概率也可与每条边相联系。

由决策者和大自然来确定将到达哪个顶点。当决策者试图对状态 x 应用行动 u 时，由于大自然的缘故而可能产生几条向外的边。如果试图采用不同的行动，那么这个可能的往外去的边的集合会发生改变。一旦大自然采取行动，那么穿越特定的边将到达新的状态，但是这不完全受决策者控制。

例 10.1（穿越数字） 令 $X=\mathbb{Z}$，$U=\{-2,2,u_T\}$ 和 $\Theta=\{-1,0,1\}$。决策者和大自然的行动集合对所有状态来说是相同的。状态转移方程 $x_{k+1}=f(x_k,u_k,\theta_k)=x_k+u_k+\theta_k$。对每条边付出的是单位代价。因此对于所有 x,θ 和 $u\neq u_T$，有 $l(x,u,\theta)=1$。初始状态是 $x_I=100$，目标集合是 $X_G=\{-1,0,1\}$。

考虑在非确定的不确定性模型下执行一个行动序列(–2,–2,…,–2)。即我们试图在每个阶段向左移动两个单元。在执行第一个–2 以后，可能的下一个状态的集合是{97,98,99}。大自然可能会使过程减速到每个阶段运动一个单元，或者使过程加速到以每个阶段 3 个单元的速度接近 X_G。100 个阶段以后肯定到达目标，不管大自然如何行动。一旦到达 X_G，就应该应用 u_T。如果改变问题使得 $X_G=\{0\}$，那么将不可能保证一定能够到达目标，因为大自然可能引起错过目标。

现在令 $U=\{-1,1,u_T\}$ 和 $\Theta=\{-2,-1,0,1,2\}$。在非确定的不确定性下，问题不再能够得到解决，因为大自然现在强大到足以在最坏情况下使得状态在错误的方向上运动。但是可以定义和解决该问题的一个合理的概率版本。假设对于每个 $\theta\in\Theta$，$P(\theta)=1/5$。从 $P(\theta)$ 可以定义转移概率。例如，如果 $x_k=100$ 和 $u_k=-1$，那么对于 $97\leqslant x_k\leqslant 101$，有 $P(x_{k+1}\mid x_k,u_k)=1/5$；否则 $P(x_{k+1}\mid x_k,u_k)=0$。用概率表述，那么到达目标 $X_G=\{-1,0,1\}$ 的概率为非零，即使在最坏情况下也不能保证到达目标。 ■

例 10.2（在栅格上移动） 可以构造一个基于栅格的机器人规划模型。一个简单的例子如图 10.1 所示。状态空间是平面内 15×15 整数栅格的一个子集。状态用(i,j)表示，其中$1\leqslant i,j\leqslant 15$；但是中心区域的点（见图 10.1）不包括在 X 中。

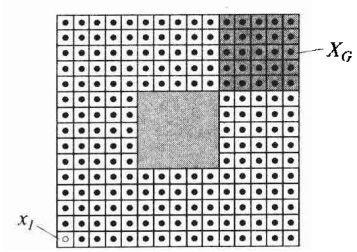


图 10.1　一个基于栅格的、有大自然干扰的最短路径问题

令 $A=\{0,1,2,3,4\}$ 为一个行动集合，它们分别表示“停止”、“向右”、“向上”、“向左”、“向下”。令 $U=A\cup u_T$。对于每个 $x\in X$，令 $U(x)$ 包含 u_T 以及从 x 所能应用的每个行动（沿着边界有些行动是不可应用的）。

令 $\Theta(x,u)$ 表示用 u 完成移动以后 A 中所有可应用的行动的集合。例如，如果 $x=(2,2)$ 且 u=3，那么机器人试图向(1,2)移动。从这个状态，有 3 个相邻状态分别对应于大自然的

一个行动。因此在这种情况下 $\Theta(x,u)$ 为 $\{0,1,2,4\}$。行动 $\theta=3$ 并不出现，因为没有状态在(1,2)的左边。假设采用概率模型，并且大自然的每个行动是等可能性的。

加入 u_k 和 θ_k 的影响可以形成状态转移函数 f。例如，如果 $x_k=(i,j)$、$u_k=1$ 且 $\theta_k=2$，那么 $x_{k+1}=(i+1,j+1)$。如果 θ_k 已经为 3，那么有两个行动不可用且 $x_{k+1}=(i,j)$。如果没有大自然，那么应该假设 $\theta_k=0$。像以前一样，一旦采用 u_T，那么状态将不再改变。不管大自然采取什么行动。

对于代价泛函，令 $l(x_k,u_k)=1$（除非 $u_k=u_T$；在这种情况下，$l(x_k,u_t)=0$）。对于最后阶段，如果 $x_F\in X_G$，那么令 $l_F(x_F)=0$；否则令 $l_F(x_F)=\infty$。一个合理的任务是使机器人用最少步数到达并停在 X_G 中。那么需要反馈规划，这将在第 10.1.3 节介绍，可以用第 10.2.1 节的方法来计算这个问题的最优规划。

这个例子很容易扩展到在复杂二维或者更高维数迷宫中移动的情况。如果栅格分辨率足够高，那么可以得到运动规划的一个近似。更可取的是允许在任意方向上移动，而不是仅在 4 个方向上移动。这种情况将在第 10.6 节介绍，那里讨论连续状态空间中的规划。 ■

10.1.2 前向投影与后向投影

前向投影是一个非常有用的概念，它刻画规划在执行过程中的行为。在考虑不确定性之前，规划如所预料的那样准确地执行。当一系列行动应用于初始状态时，所产生的状态序列可以用状态转移方程来计算。既然现在状态的转移是不可预知的，那么我们愿意想象在将来多个阶段中可能的状态是什么样的。在非确定的不确定性情况下，给定状态和规划，这涉及计算将来可能状态的一个集合。在概率情况下，计算的是状态的概率分布。

非确定的前向投影 为了书写方便，本节假设对所有 $x\in X$，有 $U(x)=U$。在第 10.1.3 节将取消这个假设。

假设初始状态 $x_1=x_I$ 已知。如果应用行动 $u_1\in U$，那么可能的下一个状态的集合是：

$$X_2(x_1,u_1)=\{x_2\in X\mid \exists\theta_1\in\Theta(x_1,u_1),x_2=f(x_1,u_1,\theta_1)\} \tag{10.10}$$

它是式（10.5）的特殊形式。现在假设将采取行动 $u_2\in U$。前向投影必须确定对 x_1 应用行动 u_1 和 u_2 后将到达哪个状态。这可以表示为：

$$\begin{aligned}X_3(x_1,u_1,u_2)=\{x_3\in X\mid &\exists\theta_1\in\Theta(x_1,u_1),\exists\theta_2\in\Theta(x_2,u_2)\\ &x_2=f(x_1,u_1,\theta_1),x_3=f(x_2,u_2,\theta_2)\}\end{aligned} \tag{10.11}$$

这个思路可以重复进行任意多次，但是用目前的符号，它非常不方便。那么可以递归地表述前向投影。假设固定历史行动 $\tilde{u}_k$。给定 X_k 为第 k 阶段的前向投影，令 $X_{k+1}(X_k,u_k)$ 表示第 $k+1$ 阶段的前向投影。这可以表示为：

$$X_{k+1}(X_k,u_k)=\{x_{k+1}\in X\mid \exists x_k\in X_k,\exists\theta_k\in\Theta(x_k,u_k),x_{k+1}=f(x_k,u_k,\theta_k)\} \tag{10.12}$$

从初始条件 $X_1=\{x_1\}$ 应用多次递归来计算 X_{k+1}。

例 10.3（非确定的前向投影） 回忆例 10.1 给出的第一个模型，其中 $U=\{-2,2,u_T\}$ 和 $\Theta=\{-1,0,1\}$。假设 $x_1=0$ 和应用行动 $u=2$。一步前向投影为 $X_2(0,2)=\{1,2,3\}$。如果再次应用 $u=2$，两步前向投影为 $X_3(0,2,2)=\{2,3,4,5,6\}$。重复进行该过程，那么 k 步前向投影

为 $\{k,\cdots,3k\}$。 ■

概率前向投影　概率前向投影可以被认为是一个马尔可夫过程，因为一旦给出行动，就不需要进行“决策”。假设给定 x_k 且应用 u_k。那么 x_{k+1} 的概率分布是多少？这已经由式（10.6）确定，并且是一个一步前向投影。考虑两步概率前向投影 $P(x_{k+2}|x_k,u_k,u_{k+1})$。这可以用边缘化来计算：

$$P(x_{k+2}|x_k,u_k,u_{k+1})=\sum_{x_{k+1}\in X}P(x_{k+2}|x_{k+1},u_{k+1})P(x_{k+1}|x_k,u_k) \tag{10.13}$$

继续计算前向投影要求嵌套的总和，它边缘化所有中间状态。例如三步前向投影为：

$$P(x_{k+3}|x_k,u_k,u_{k+1},u_{k+2})= \sum_{x_{k+1}\in X}\sum_{x_{k+2}\in X}P(x_{k+3}|x_{k+2},u_{k+2})P(x_{k+2}|x_{k+1},u_{k+1})P(x_{k+1}|x_k,u_k) \tag{10.14}$$

通过借鉴线性代数的代数特征可以得到概率前向投影的一个简洁表示。对于每个行动 $u\in U$，令状态转移矩阵 M_u 为概率的一个 $n\times n$ 矩阵，其中 $n=|X|$。矩阵定义为：

$$M_u=\begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,n} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,n} \\ \vdots & \vdots & & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,n} \end{pmatrix} \tag{10.15}$$

其中：

$$m_{i,j}=P(x_{k+1}=i|x_k=j,u) \tag{10.16}$$

对于每个 j，M_u 的第 j 列的和必须为 1，并且可以解释为从状态 $x_k=j$ 应用 u_k 所得到的 X 上的概率分布。

令 v 表示一个 n 维列向量，它表示 X 上的任意概率分布。乘积 $M_u v$ 产生一个表示 X 上任意概率分布的列向量，该分布是从 v 开始并应用 u 以后得到的。矩阵乘法进行 n 次内积，每个内积都是一个如式（10.13）所示的边缘化。第 k 步的前向投影现在可以用 $k-1$ 个状态转移矩阵的乘积来表示。假设固定 $\tilde{u}_{k-1}$。令 $v=(0\ 0\ \cdots 0\ 1\ 0\ \cdots 0)$，它指出 x_1 是已知的（以概率 1）。前向投影可以计算为：

$$v'=M_{u_{k-1}}M_{u_{k-2}}\cdots M_{u_2}M_{u_1}v \tag{10.17}$$

其中 v' 的第 i 个元素为

$$P(x_k=i|x_1,\tilde{u}_{k-1})$$

例 10.4（概率前向投影）　还采用例 10.1 给出的第一个模型，但是现在大自然每个行动的概率为 1/3。假设初始时 $x_1=0$ 且每步都用 $u=2$。一步前向投影在状态序列(1,2,3)上产生概率：

$$[1/3\ \ 1/3\ \ 1/3] \tag{10.18}$$

两步前向投影在 (2,3,4,5,6) 产生：

$$[1/9\ \ 2/9\ \ 3/9\ \ 2/9\ \ 1/9] \tag{10.19}$$

■

后向投影　有时可能需要定义一个以前状态的集合，从这些状态可以得到一个或者更多目前的状态。例如它们将在第 10.2.3 节中定义基于图的规划算法时非常有用。这涉及持续进

行后向投影，后向投影与前向投影相似，只不过反方向运行。后向投影在第 8.5.2 节被认为用来跟踪连续状态空间上类 Dijkstra 算法中的活动状态。目前后向投影需要讨论不确定性。

考虑非确定的不确定性。令给定状态 $x \in X$。在固定行动 u 的情况下。以前的什么状态 $x' \in X$ 可能导致 x 呢？这只取决于大自然可能的选择，所以可以表示为：

$$\mathrm{WB}(x,u) = \{x' \in X \,|\, \exists \theta \in \Theta(x',u),\ \ x = f(x',u,\theta)\} \tag{10.20}$$

符号 $\mathrm{WB}(x,u)$ 指的是在 u 作用下 x 的弱后向投影，它给出可能一步到达 x 的所有状态的集合。

之所以称为“弱”是因为它不能保证一定能够到达 x，而保证到达 x 是一个更强的条件。通过保证到达 x，u 作用下 x 的一个强后向投影定义为：

$$\mathrm{SB}(x,u) = \{x' \in X \,|\, \forall \theta \in \Theta(x',u), x = f(x',u,\theta)\} \tag{10.21}$$

式（10.20）和式（10.21）之间的区别在于一个是存在一个大自然的行动使得一定到达 x，而另外一个是对所有大自然行动来说都一定能够到达 x。$\mathrm{SB}(x,u) \subseteq \mathrm{WB}(x,u)$。在许多情况下，$\mathrm{SB}(x,u) = \phi$，而 $\mathrm{WB}(x,u)$ 很少为空。第 8.5.2 节的式（8.66）所介绍的后向投影没有涉及不确定性；因此弱后向投影和强后向投影之间没有区别。

进行两个有用的推广：（1）一个后向投影可以从一个状态集合定义；（2）不需要固定行动。考虑状态 x 的一个集合 $S \subseteq X$，而不是固定状态 x。什么状态在 u 作用下可能一步到达 S 中的元素？这是 u 作用下 S 的弱后向投影：

$$\mathrm{WB}(S,u) = \{x' \in X \,|\, \exists \theta \in \Theta(x',u), f(x',u,\theta) \in S\} \tag{10.22}$$

它也可以表示为：

$$\mathrm{WB}(S,u) = \bigcup_{x \in S} \mathrm{WB}(x,u) \tag{10.23}$$

相似地，u 作用下 S 的强后向投影定义为：

$$\mathrm{SB}(S,u) = \{x' \in X \,|\, \forall \theta \in \Theta(x',u), f(x',u,\theta) \in S\} \tag{10.24}$$

需要注意的是，$\mathrm{SB}(S,u)$ 不能由 $\mathrm{SB}(x,u)$ 在所有 $x \in S$ 上的并集形成。另外可以看到：对于每个 $x_k \in \mathrm{SB}(S,u_k)$，有 $X_{k+1}(x_k,u_k) \subseteq S$。

现在去掉对 u 的依赖性。这产生了集合 S 的一个后向投影。有一些这样的状态，存在一个行动使得这些状态能够到达 S。S 的弱后向投影为：

$$\mathrm{WB}(S) = \{x' \in X \,|\, \exists u \in U(x),\ \ x \in \mathrm{WB}(S,u)\} \tag{10.25}$$

S 的强后向投影为：

$$\mathrm{SB}(S) = \{x' \in X \,|\, \exists u \in U(x),\ \ x \in \mathrm{SB}(S,u)\} \tag{10.26}$$

注意到 $\mathrm{SB}(S) \subseteq \mathrm{WB}(S)$。

例 10.5（后向投影） 仍然考虑例 10.1 给出的第一个模型。后向投影 $\mathrm{WB}(0,2)$ 表示应用 $u = 2$ 可能到达 $x = 0$ 的所有状态的集合；结果是 $\mathrm{WB}(0,2) = \{-3,-2,-1\}$。从 $\mathrm{WB}(0,2)$ 中的任何状态都不能确定到达状态 0。因此 $\mathrm{SB}(0,2) = \phi$。

现在考虑从目标 $X_G = \{-1,0,1\}$ 应用行动 $u = 2$ 的后向投影。弱后向投影为：

$$\mathrm{WB}(X_G,2) = \mathrm{WB}(-1,2) \cup \mathrm{WB}(0,2) \cup \mathrm{WB}(1,2) = \{-4,-3,-2,-1,0\} \tag{10.27}$$

强后向投影 $\mathrm{SB}(X_G,2) = \{-2\}$。从 $\mathrm{WB}(X_G,2)$ 中的任何其他状态，大自然都会导致错过目标。注意到 $\mathrm{SB}(X_G,2)$ 不能由每个 $x \in X_G$ 上的 $\mathrm{SB}(x,2)$ 的并集来构造。

最后考虑不依赖于某个行动的后向投影。那么有 $\mathrm{WB}(X_G)=\{-4,-3,\cdots,4\}$ 且 $\mathrm{SB}(X_G)=X_G$。在后一种情况下，X_G 中的所有状态落在 $\mathrm{SB}(X_G)$ 中，因为应用的是 u_T。如果不允许 u_T，那么我们得到 $\mathrm{SB}(X_G)=\{-2,2\}$。■

也可能有其他类型的后向投影，但是我们不定义它们。一种是多步后向投影。另一种是定义概率情况下的后向投影。这相当复杂。一个概率后向投影的例子是发现至少能够以概率 p 到达 S 中的一个状态的所有状态的集合。

10.1.3　一个规划及其执行

在第 2 章中，规划（结果）由一系列行动来给定。在那里这是可能的，因为行动的效果是完全可以预测的。在整个第 II 部分中，一个规划是指一条路径，它是行动序列的连续版本。第 8.2.1 节介绍的规划表示为状态空间上的一个函数。那里没有对不确定性进行建模（除了初始状态可能存在不确定性以外）。

作为由大自然引起的不可预知的结果，现在重要的是将规划的定义和它的执行分开。相同的规划可能从相同的初始状态执行许多次；但是因为有大自然的影响，那么可能得到不同的未来状态。这要求以将状态映射到行动的一个规划的形式使用反馈。

定义一个规划　表述 10.1 的（反馈）规划被定义为一个函数 $\pi: X \to U$，它为每个 $x \in X$ 产生一个行动 $\pi(x) \in U(x)$。尽管由于大自然的原因，未来的状态可能并不知道，但是如果给定 π，那么至少知道对未来状态采取什么行动。在其他研究工作中，π 被称为反馈策略、反馈控制律、反应规划[342]和条件规划。

对于某些问题，特别是当 K 固定为某有限值时，可能需要与阶段有关的规划。这使得在不同阶段即使对相同的状态也可以选择不同的行动。令 $\mathcal{K}$ 表示阶段的集合 $\{1,\cdots,K\}$。依赖于阶段的规划定义为 $\pi: X \times \mathbb{K} \to U$。因此行动是 $u=\pi(x,k)$。第 2.3 节给出的 K 步规划的定义是目前定义的特殊情况。在那里行动只依赖于阶段，因为未来的状态总是可预测的。这里它们不再是可预测的了，必须包括在 π 的定义域中。除非专门提及，默认情况下都假设 π 不依赖于阶段。

一旦表述 π，那么状态仅仅是目前状态和大自然的函数。下一个状态由 $f(x,\pi(x),\theta)$ 给出。代价项也是如此。

固定规划下的前向投影　现在定义在执行特定规划下的前向投影。行动的表达用 π 代替。每次需要对状态 $x \in X$ 采取行动时都得到 $\pi(x)$。在这种表述中，对每个 $x \in X$ 可以采取不同的 $U(x)$，即对于每个 $x \in X$，正确定义 π 来使用 $U(x)$ 中实际可行的任何行动。

首先我们考虑非确定的情况。假设初始状态 x_1 和规划 π 已知。即 $u_1=\pi(x_1)$，它可以代入式（10.10）来计算一步前向投影。为了计算两步前向投影，用 $\pi(x_2)$ 确定 u_2。固定规划下非确定的前向投影的递归表示为：

$$X_{k+1}(x_1,\pi)=\{x_{k+1} \in X \mid \exists \theta_k \in \Theta(x_k,\pi(x_k)) \\ x_k \in X_k(x_1,\pi), x_{k+1}=f(x_k,\pi(x_k),\theta_k)\} \tag{10.28}$$

式（10.10）中的概率前向投影可以进行适应性修改来使用 π，它产生：

$$P(x_{k+2}|x_k,\pi)=\sum_{x_{k+1} \in X} P\left(x_{k+2}|x_{k+1},\pi(x_{k+1})\right)P\left(x_{k+1}|x_k,\pi(x_k)\right) \tag{10.29}$$

基本思路进行 $k-1$ 次就可以计算 $P(x_k|x_1,\pi)$。

再次使用状态转移矩阵来表达概率前向投影。在式（10.15）中，所有列对应于行动 u 的应用。令 M_π 表示由于固定规划 π 而得到的前向投影。M_π 的每列可能表示一个不同行动，因为每列表示一个不同的状态 x_k。M_π 的每一项是：

$$m_{i,j}=P(x_{k+1}=i|x_k=j,\pi(x_k)) \tag{10.30}$$

所产生的 M_π 定义了一个应用规划 π 而产生的马尔可夫过程。

规划的图表示 与大自然之间的对策涉及两个决策者：大自然和机器人。一旦表述了规划，机器人的决策就固定了，这样大自然就成为是仅有的决策者。采用这种阐述，有向图可以用与第 2.1 节相同的方法进行定义，除了用大自然行动代替机器人的行动。甚至可以假设大自然本身面临一个与表述 2.1 相同的离散可行规划问题，其中用 $\Theta(x,\pi(x))$ 代替 $U(x)$ 且没有目标集合。令 $\mathcal{G}_\pi$ 表示基于规划的状态转移图，它是在执行 π 所产生的。$\mathcal{G}_\pi$ 的顶点集合是 X。如果存在某 $\theta\in\Theta(x,\pi(x))$ 使得 $x'=f(x,\pi(x),\theta)$，那么 $\mathcal{G}_\pi$ 中存在从 x 到 x' 的一条有向边。因此从 $\mathcal{G}_\pi$ 的每个顶点，给定按照 π 采取的行动，向外的边的集合表示向下一步状态所进行的转移。在概率不确定情况下，$\mathcal{G}_\pi$ 变成一个加权图，图中每条边的概率为 $P(x'|x,\pi(x),\theta)$。在这种情况下，$\mathcal{G}_\pi$ 对应于通常用来描述马尔可夫链的图。

沿着从目前状态指向外的边，非确定的前向投影很容易从 $\mathcal{G}_\pi$ 产生。向外的边产生一步前向投影的状态。这些状态的向外边又产生两步前向投影，如此继续。概率前向投影也可以从 $\mathcal{G}_\pi$ 产生。

反馈规划的代价 考虑从状态 $x_1\in X$ 执行一个规划的尚需代价。所产生的代价取决于所访问的状态序列、规划所用的行动以及大自然的行动。在第 2 章通过加入代价项就可以得到，但是现在还依赖于大自然。可以用最坏情况分析和期望情况分析，它们都是将第 9.2 节扩展到状态空间和多步过程。

令 $\mathcal{H}(\pi,x_1)$ 表示状态-行动-大自然历史的集合，它们是从初始状态 x_1 应用 π 而产生的。从 x_1 在给定规划 π 的情况下，尚需代价 $G_\pi(x_1)$ 可以采用最坏情况分析来得到：

$$G_\pi(x_1)=\max_{(\tilde{x},\tilde{u},\tilde{\theta})\in\mathcal{H}(\pi,x_1)}\left\{L(\tilde{x},\tilde{u},\tilde{\theta})\right\} \tag{10.31}$$

它是从 x_1 在规划 π 作用下所有可能轨迹上的最大代价。如果其中任意一个没有终止在目标中，那么代价将变成无穷大。在式（10.31）中，$\tilde{x},\tilde{u}$ 和 $\tilde{\theta}$ 是无限过去的历史，尽管可以预料它们对代价的影响由于应用 u_T 而很早就终止了。

用最坏情况分析所得到的一个最优规划是在所有可能规划（所有对状态指派行动的方法）上最小化 $G_\pi(x_1)$ 的规划。在可行规划情况下，通常有许多等价选择。有时任务可能仅仅是发现一个可行规划，这意味着所有轨迹都必须到达目标，但是不需要最优化代价。

采用概率不确定性，规划的代价可以用期望情况分析来得到：

$$G_\pi(x_1)=E_{\mathcal{H}(\pi,x_1)}\left[L(\tilde{x},\tilde{u},\tilde{\theta})\right] \tag{10.32}$$

其中 E 表示 $\mathcal{H}(\pi,x_1)$ 上的数学期望（例如根据加权和进行计算，其中每一项都有一个状态-行动-大自然历史及其代价 $L(\tilde{x},\tilde{u},\tilde{\theta})$ 的概率的权值）。这也可以解释为从 x_1 开始的轨迹上期望的代价。如果其中任意一个的概率非零且没有终止在目标，那么 $G_\pi(x_1)=\infty$。在概率情形下，任务实际上是发现一个规划来最小化 $G_\pi(x_1)$。

现在出现了一个有趣的问题：从每个初始状态 $x_1 \in X$，相同的规划 π 都是最优的吗？或者说我们是否需要为每个初始状态发现不同的最优规划呢？幸运的是，一个规划对所有初始状态都是最优的。为什么？这种行为也可以从第 8.2.1 节观察到。如果 π 从某 x_1 是最优的，那么在从某 x_1 执行 π 所能够访问的每个其他状态上也必须是最优的。令 x 表示某已经访问过的状态。如果从该状态的 π 不是最优的，那么应该存在一个更好的规划，用该规划从 x 到达目标所用代价更小。这与 π 的最优性相矛盾，因为解必须经过 x。令 π^* 表示一个从每个初始状态都是最优的规划。

10.2　计算反馈规划的算法

10.2.1　值迭代

幸运的是，第 2.3.1.1 节的值迭代方法可以扩展到处理预测中的不确定性。这就是第 2 章介绍值迭代的原因。在第 2 章很容易描述值迭代，因为没有考虑大自然。在目前情况下，值迭代仍然很有效，它能够很容易地解决涉及成千上万个状态的问题。

在第 10.2.1 节都假设状态空间 X 是有限的。如果在整个空间上的尚需代价值不需要递增计算，那么可以扩展到可数无限状态空间的情况。

这里只考虑后向值迭代。也可以定义值迭代的前向版本。

非确定的情况　假设采用大自然的非确定的模型。将产生动态规划式（10.39）。这直接产生一个迭代方法来计算最小化最坏情况代价的规划。下面的表示方法遵循第 2.3.1.1 节的思路，因此可以参考那些内容。

最优规划 π^* 通过计算最优尚需代价函数来发现。对于 $1 \leqslant k \leqslant F$，令 G_k^* 表示在执行最优规划下从第 k 步积累到第 F 步所得到的最坏情况代价：

$$G_k^*(x_k) = \min_{u_k} \max_{\theta_k} \min_{u_{k+1}} \max_{\theta_{k+1}} \cdots \min_{u_K} \max_{\theta_K} \left\{ \sum_{i=k}^{K} l(x_i, u_i, \theta_i) + l_F(x_F) \right\} \tag{10.33}$$

式（10.33）中的 min 和 max 的内部是代价泛函式（10.8）的最后 $F-k$ 项。为简单起见，在式（10.33）的 min 和 max 中每个 u_i 和 θ_i 的范围并没有规定。$k=F$ 的最优尚需代价为：

$$G_k^*(x_F) = l_F(x_F) \tag{10.34}$$

它与式（2.6）相同。

现在考虑在 X 上进行 K 步，k 从 F 变化到 1，每次从 G_{k+1}^* 计算 G_k^*。在第一次迭代时，G_F^* 从 l_F 复制而来。在第二次迭代时，为每个 $x_K \in X$ 计算 G_K^*（与式（2.7）进行比较）：

$$G_K^*(x_K) = \min_{u_K} \max_{\theta_K} \{ l(x_K, u_K, \theta_K) + l_F(x_F) \} \tag{10.35}$$

其中 $u_K \in U(x_K)$，$\theta_K \in \Theta(x_K, u_K)$。因为 $l_F = G_F^*$，$x_F = f(x_K, u_K, \theta_K)$，带入式（10.35）得到：

$$G_K^*(x_K) = \min_{u_K} \max_{\theta_K} \{ l(x_K, u_K, \theta_K) + G_F^*(f(x_K, u_K, \theta_K)) \} \tag{10.36}$$

它计算从 K 到 $F = K+1$ 的所有最优一步规划的代价。

更一般情况下，一旦给定 G_{k+1}^*，那么就可以计算 G_k^*。仔细研究式（10.33），注意到它可以记为（与式（2.9）进行比较）：

$$G_k^*(x_k)=\min_{u_k}\max_{\theta_k}\left\{\min_{u_{k+1}}\max_{\theta_{k+1}}\cdots\min_{u_K}\max_{\theta_K}\left\{l(x_k,u_k,\theta_k)+\sum_{i=k+1}^{K}l(x_i,u_i,\theta_i)+l_F(x_F)\right\}\right\} \tag{10.37}$$

它将第一项放在和式之外，将 u_k 上的最小化与其他 $u_{k+1}\sim u_K$ 分开。第二个 min 和 max 不影响 $l(x_k,u_k,\theta_k)$ 项；因此 $l(x_k,u_k,\theta_k)$ 可以放到外面得到（与式（2.10）进行比较）：

$$G_k^*(x_k)=\min_{u_k}\max_{\theta_k}\left\{l(x_k,u_k,\theta_k)+\min_{u_{k+1}}\max_{\theta_{k+1}}\cdots\min_{u_K}\max_{\theta_K}\left\{\sum_{i=k+1}^{K}l(x_i,u_i,\theta_i)+l(x_F)\right\}\right\} \tag{10.38}$$

内部的多个 min 和 max 表示 G_{k+1}^*，它产生递归（与式（2.11）进行比较）：

$$G_k^*(x_k)=\min_{u_k\in U(x_k)}\left\{\max_{\theta_k}\left\{l(x_k,u_k,\theta_k)+G_{k+1}^*(x_{k+1})\right\}\right\} \tag{10.39}$$

概率情况 现在考虑概率情况。还用第 2.3.1.1 节的表示来得到值迭代方法。对于 k 从 1 到 F，令 G_k^* 表示在执行最优规划时从第 k 步到第 F 步的期望代价（与式（2.5）进行比较）：

$$G_k^*(x_k)=\min_{u_k,\cdots,u_K}\left\{E_{\theta_k,\cdots,\theta_K}\left[\sum_{i=k}^{K}l(x_i,u_i,\theta_i)+l_F(x_F)\right]\right\} \tag{10.40}$$

边界条件 $k=F$ 下的最优尚需代价变为式（10.34）。

再次，算法在 X 上进行 K 步，k 从 F 到 1，每次从 G_{k+1}^* 计算 G_k^*。在第一次迭代中，G_F^* 从 l_F 复制。在第二次迭代中，为每个 $x_K\in X$ 计算 G_K^*（与式（2.7）进行比较）：

$$G_K^*(x_K)=\min_{u_K}\left\{E_{\theta_K}\left[l(x_K,u_K,\theta_K)+l_F(x_F)\right]\right\} \tag{10.41}$$

其中 $u_K\in U(x_K)$，并且数学期望是在 θ_K 上的。带入式（10.41）得到（与式（2.8）进行比较）：

$$G_K^*(x_K)=\min_{u_K}\left\{E_{\theta_K}\left[l(x_K,u_K,\theta_K)+G_F^*(f(x_K,u_K,\theta_K)\right]\right\} \tag{10.42}$$

一般迭代过程为：

$$G_k^*(x_k)=\min_{u_k}\left\{E_{\theta_k}\left[\min_{u_{k+1},\cdots,u_K}\left\{E_{\theta_{k+1},\cdots,\theta_K}\left[l(x_k,u_k,\theta_k)+\sum_{i=k+1}^{K}l(x_i,u_i,\theta_i)+l_F(x_F)\right]\right\}\right]\right\} \tag{10.43}$$

这是将第一项放在和式之外、将 u_k 上的最小化与其他的最小化分开而得到的。第二个 min 和期望不影响 $l(x_k,u_k,\theta_k)$ 项；因此将 $l(x_k,u_k,\theta_k)$ 放到外面得到：

$$G_k^*(x_k)=\min_{u_k}\left\{E_{\theta_k}\left[l(x_k,u_k,\theta_k)+\min_{u_{k+1},\cdots,u_K}\left\{E_{\theta_{k+1},\cdots,\theta_K}\left[\sum_{i=k+1}^{K}l(x_i,u_i,\theta_i)+l_F(x_F)\right]\right\}\right]\right\} \tag{10.44}$$

内部的 min 和数学期望定义了 G_{k+1}^*，它产生递归（与式（2.11）和式（10.39）进行比较）：

$$\begin{aligned}G_k^*(x_k)&=\min_{u_k\in U(x_k)}\left\{E_{\theta_k}\left[l(x_k,u_k,\theta_k)+G_{k+1}^*(x_{k+1})\right]\right\}\\&=\min_{u_K\in U(x_k)}\left\{\sum_{\theta\in\Theta(x_k,u_k)}\left(l(x_k,u_k,\theta_k)+G_{k+1}^*(f(x_k,u_k,\theta_k))\right)P\left(\theta_k\middle|x_k,u_k\right)\right\}\end{aligned} \tag{10.45}$$

如果代价项不依赖于 θ_k，那么它可以记为 $l(x_k,u_k)$，式（10.45）简化为：

$$G_k^*(x_k)=\min_{u_k\in U(x_k)}\left\{l(x_k,u_k)+\sum_{x_{k+1}\in X}G_{k+1}^*(x_{k+1})P\left(x_{k+1}\middle|x_k,u_k\right)\right\} \tag{10.46}$$

表达式 $P(x_{k+1}|x_k,u_k)$ 隐含着状态转移对 θ_k 的依赖性，它的定义用到 $P(\theta_k|x_k,u_k)$ 和状态转移

方程 f。式（10.46）可能比式（10.45）更便于实现。

收敛问题　如果在问题的定义中固定最大阶段（步）数，那么能够保证收敛。但是假设对阶段数没有限制。回忆第 2.3.2 节中每次值迭代将总路径长度增加 1。阶段索引在后向动态规划中并不重要，因为任意移动阶段索引对值都没有影响。最后，算法结束，因为已经为从目标可以到达的所有状态计算了最优尚需代价值。这产生了一个平稳尚需代价函数，因为它们的值不再改变。从目标所能够到达的状态的代价值收敛到有限值，其他的则还是无穷大。如第 2.3.2 节所述，影响平稳尚需代价函数的存在性的问题只有图中的负循环。在这种情况下，最好的规划将永远环绕着这个循环，这将代价减到 $-\infty$。

在目前情形下，也会出现一个平稳尚需代价函数，但是循环使得收敛困难。这种情形由于大自然的影响而更加复杂。考虑基于规划的状态转移图 $\mathcal{G}_\pi$ 是有帮助的。首先考虑非确定的情况，如果从某状态 x_1 存在一个规划，对于该状态大自然所有可能行动引起绕着积累负代价值的循环，那么 x_1 的最优尚需代价收敛到 $-\infty$，它使得值迭代不能停止。这些情况可以提前检测到，可以避免这样的初始状态（某些甚至可能是在状态空间的不同连通分支中）。

也可能有不可避免的正循环。在第 2.3.2 节中，尚需代价函数不同的行为取决于目标集合是否是可达的。由于大自然的影响，目标集合可能是可达的或者一定可达的，如图 10.2 所示。为了从某初始状态是可达的，必须存在一个规划 π，对该规划来说存在将会将状态引到目标集的大自然的行动序列。一定可达是指不管大自然采取什么行动序列，都必须到达目标。如果目标从某状态 x_1 可能是可达的，但不是一定可达的，并且所有的边有正代价，那么当值迭代重复进行时，x_1 的尚需代价值趋近于无穷。例如每个基于规划的状态转移图可能包含一个正代价的循环，并且在最坏情况下，大自然可能引起状态无限循环。如果值迭代的收敛性只在保证能够到达目标集的状态上进行评估，并且如果没有负循环。那么当所有尚需代价值保持平稳时算法应该终止。

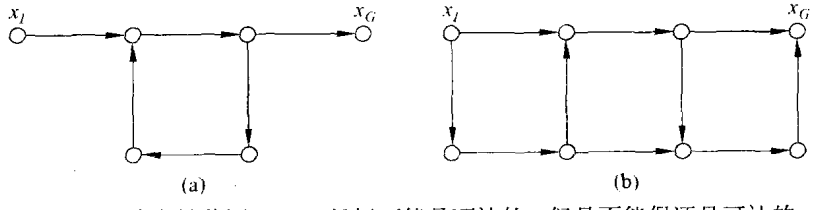


图 10.2　基于规划的状态转移图；（a）目标可能是可达的，但是不能保证是可达的，因为可能出现无限循环；（b）目标确保是可达的，因为所有的流动都导向目标

对于概率，有三种情况：

（1）有限次迭代后，值迭代到达一个平稳尚需代价函数。

（2）从任何意义上，值迭代都不收敛。

（3）值迭代只渐进收敛到一个平稳尚需代价函数。当值收敛时，迭代的次数趋近于无穷。

前两种情形已经发生过。如果存在一个规划 π，它的 $\mathcal{G}_\pi$ 没有循环，那么发生第一种情况。如果有负或者正的循环，对于这些循环来说循环中的所有边的概率为 1，那么发生第二种情况。这种情况本质上等价于非确定的情况。最坏情况分析假设大自然采用的是可能

的最坏的行动。对于概率情况，设置大自然行动的概率为 1 来迫使产生这些行动。

第三种情况是概率情况所特有的。这由 $\mathcal{G}_\pi$ 中的正或负循环引起，对 $\mathcal{G}_\pi$ 来说边的概率在(0,1)之间。甚至最优规划可能也有这种循环。当值迭代考虑越来越长的路径时，可能环绕循环许多次。但是每次环绕循环时概率减少。概率以阶段数的指数形式减少，因此代价只是线性积累。当迭代的次数趋近无穷时，尚需代价函数的变化逐渐减少并收敛到固定值。如果允许某近似误差，那么一旦所有 X 上的最大变化小于某门限 ϵ，那么迭代就终止。为得到一定性能的解而需要进行的值迭代次数依赖于跟踪循环的概率和它们的代价。如果概率低，那么算法收敛迅速。

例 10.6（转移图中的循环） 假设选择规划 π 产生如图 10.3 所示的基于规划的状态转移图。使用概率模型，概率显示在每条边上。为简单起见，假设每次转移在所有 x,u 和 θ 上产生单位代价 $l(x,u,\theta)=1$。

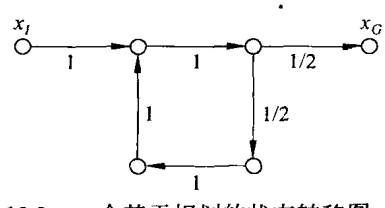


图 10.3 一个基于规划的状态转移图，它引起渐近收敛；转移概率显示在每条边上；随着循环的增加，路径会越来越长，但是概率变得越来越小

很容易计算从 x_I 开始的期望代价。到达 x_G 的代价为 3 的概率 1/2。代价为 7 的概率为 1/4。代价为 11 的概率为 1/8。每循环一次，代价增加 4，但是概率减半。这导致无穷级数：

$$G_\pi(x_I)=3+4\sum_{i=0}^{\infty}\frac{1}{2^i}=7 \qquad (10.47)$$

无穷和是标准的几何级数，它收敛于 1；因此式（10.47）收敛于 7。

即使代价收敛于一个有限值，这也仅发生在极限状态下。理论上需要无穷多次值迭代来得到该结果。对于多数应用，近似解就足够了，用少量次数的迭代就可以得到非常高的精度（例如 20 次迭代后，变化的数量级为十亿分之一）。因此一般情况下，在尚需代价最大变化不超过基于精度的门限值以后，最明智的是终止值迭代。

如果用非确定的不确定性，那么值迭代不收敛，因为在最坏情况下，大自然将使状态永远循环。即使目标不是一定可达的，概率不确定性模型也有合理的解。 ■

使用规划 假设对阶段数没有限制。值迭代终止以后，在 X 上确定尚需代价函数。这不是一个规划，因为每个 $x\in X$ 需要一个行动。通过记录产生式（10.45）或者式（10.39）中最小代价值的 $u\in U(x)$ 就可以得到行动。

假设值迭代已经收敛到平稳的尚需代价函数。在介绍不确定性之前，用式（2.19）确定最优行动。式（2.19）的非确定的和概率的版本分别为：

$$\pi^*(x)=\operatorname*{arg\,min}_{u\in U(x)}\left\{\max_{\theta\in\Theta(x,u)}\left\{l(x,u,\theta)+G^*(f(x,u,\theta))\right\}\right\} \qquad (10.48)$$

$$\pi^*(x)=\operatorname*{arg\,min}_{u\in U(x)}\left\{E_\theta\left[l(x,u,\theta)+G^*(f(x,u,\theta))\right]\right\} \qquad (10.49)$$

对于最优尚需代价值已知的每个 $x\in X$，计算式（10.45）产生最好的行动。

在执行规划过程中可以直接得到最优行动，而不需要存储行动。执行规划过程中每次得到 x_k，通过计算 x_k 处的式（10.48）或者式（10.49）来选择适宜的行动 $u_k=\pi^*(x_k)$。这意味着一旦了解到需要局部算子来得到行动，那么尚需代价函数自身可以解释为最优规划的一种表示形式。似乎有些奇怪的是这种局部计算能够产生全局最优；这是因为尚需代价

函数已经编码了全局代价。对于第 8.4.1 节的连续状态空间，我们也可以看到这种行为，其中导航函数用来定义一个反馈运动规划。在那里，需要梯度算子来得到运动的方向。在目前情形下，式（10.48）和式（10.49）的目的也是如此。

10.2.2　策略迭代

第 10.2.1 节的值迭代的进行是通过更新状态空间上的尚需代价值。在规划空间中进行迭代搜索也可以得到最优规划。这导致了一种称为策略迭代（policy iteration）的方法[86]，策略这个词与规划的意思相同。为概率不确定性的情况而阐述策略迭代方法，并假设 X 是有限的。只需要少量适应性修改，也可以得到非确定的不确定性的策略迭代。

策略迭代不断要求为给定的规划而计算尚需代价 π。回忆式（10.32）中对 G_π 的定义。首先假设没有不确定性，并且状态转移方程为 $x' = f(x,u)$。在 π 的作用下，可以采用第 2.3.2 节的动态规划方程（2.18）来为每个状态 $x \in X$ 产生尚需代价。为每个 $x \in X$ 复制式（2.18），去掉 min，用给定的行动 $u = \pi(x)$ 产生：

$$G_\pi(x) = l(x,\pi(x)) + G_\pi(f(x,\pi(x))) \tag{10.50}$$

在式（10.50）中，用 G_π 代替 G^*，因为没有变量需要优化（它仅仅是应用 π 时的代价）。可以认为方程（10.50）是动态规划的一种平凡形式，其中对可能规划的选择被限制在单个规划 π。如果动态规划（2.18）在所有规划的空间上成立，那么它必须在由单个规划组成的空间上成立，这反映在式（10.50）中。

如果有 n 个状态，式（10.50）产生 n 个方程，每个方程给出不同状态下 $G_\pi(x)$ 的一个表达式。对于状态 $x \in X_G$，已知 $G_\pi(x) = 0$。既然如此，用 $G_\pi(f(x,\pi(x))) = 0$ 的式（10.50）可以计算能够一步到达 X_G 的所有状态的尚需代价。一旦计算了这些尚需代价值，那么可从能够一步到达这些状态的状态计算下一波次的代价值。这个过程一直进行直到计算完所有状态的尚需代价值。这与 Dijkstra 算法很相似。

确定尚需代价的过程不应该太神秘。方程（10.50）表示状态转移图中相邻状态之间的代价是如何不一致。因为所有的这种差异已经被指出，并且为 X_G 给定初始条件，所以将方程（10.50）所表示的差异加起来就可以产生所有其他代价。相似的思路出现 Hamilton-Jacobi-Bellman 方程和庞特里亚金最小值原理中，这是在第 15.2 节介绍的。

现在我们介绍概率不确定性的情况。与式（2.18）相似的是式（10.49）。为简单起见，考虑 $l(x,u,\theta)$ 不依赖于 θ 这种特殊情况，这产生：

$$\pi^*(x) = \operatorname*{arg\,min}_{u \in U(x)} \left\{ l(x,u) + \sum_{x' \in X} G^*(x') P(x' | x,u) \right\} \tag{10.51}$$

其中 $x' = f(x,u)$。尚需代价函数 G^* 满足动态规划：

$$G^*(x) = \min_{u \in U(x)} \left\{ l(x,u) + \sum_{x' \in X} G^*(x') P(x' | x,u) \right\} \tag{10.52}$$

将式（10.52）的行动集合限制在单个规划 π，那么可以构造方程（10.50）的概率情形：

$$G_\pi(x) = l(x,\pi(x)) + \sum_{x' \in X} G_\pi(x') P(x' | x,\pi(x)) \tag{10.53}$$

其中 $x' = f(x,\pi(x))$。

每个 $x \in X$ 在应用 π 的情况下的尚需代价可以通过表示每个状态的式（10.53）来确定。那么除了 G_π 以外的所有量都给定了。这就是说如果有 n 状态，那么就有 n 线性方程和 n 未知量（每个 $x \in X$ 的 $G_\pi(x)$）。当使用方程（10.50）时也是如此，只不过方程更简单些。在概率情形下，必须解决有 n 个线性方程的系统来确定 G_π。这可以用经典线性代数方法来完成，如奇异值分解等[401,961]。

既然我们有方法计算一个规划的代价，那么策略迭代方法就很简单，如图 10.4 所示。在第 3 步，用一个规划的尚需代价 G_π 来计算其他规划。结果是这样得到的代价：如果在第一阶段尝试一个新的行动，然后在其他阶段采用 π。如果一个新的行动不能减小代价，那么 π 一定是最优的，因为这意味着式（10.54）等价于动态规划方程（10.49）。如果有可能提高 π，那么可以得到一个新的行动。新的行动必须严格优于以前的规划，并且总共只有有限次可能的规划。因此策略迭代方法在有限次迭代后收敛。

策略迭代算法：

1. 取一个初始规划 π，其中 u_T 应用在每个 $x \in X_G$，并且所有其他行动任意选择。
2. 用式（10.53）来计算每个 $x \in X$ 在规划 π 下的 G_π。
3. 用计算的 G_π 代替 G^*，用式（10.51）来计算一个更好的规划 π'：

$$\pi'(x) = \arg\min_{u \in U(x)} \left\{ l(x,u) + \sum_{x' \in X} G_\pi(x') P(x'|x,u) \right\} \tag{10.54}$$

4. 如果 π' 产生至少一个比 π 更低的尚需代价值，那么令 $\pi = \pi'$ 并且重复步骤 2 和步骤 3。否则就宣布 π 是最优规划 π^*。

图 10.4 通过计算和改进规划，策略迭代算法迭代地搜索规划空间

例 10.7（策略迭代） 用一个简单的例子来说明策略迭代。令 $X = \{a,b,c\}$，$U = \{1,2,u_T\}$。令 $X_G = \{c\}$。对所有 $x \in X$ 和 $u \in U \setminus \{u_T\}$（如果采用 u_T，则没有代价），令 $l(x,u) = 1$。每个行动的概率状态转移图如图 10.5 所示。第一步是取得一个初始规划。令 $\pi(a) = 1$，$\pi(b) = 1$；令 $\pi(c) = u_T$，因为 $c \in X_G$。

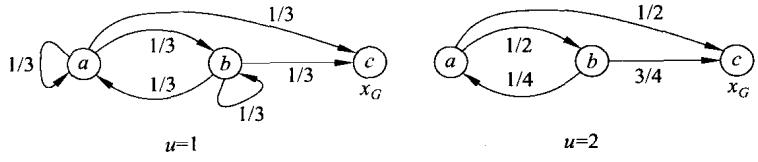


图 10.5 $u=1$ 和 $u=2$ 的概率状态转移图；没有显示 c 外的转移，因为假设对 x_g 总是采取终止行动

现在用式（10.53）计算 G_π。这产生三个方程：

$$G_\pi(a) = 1 + G_\pi(a)P(a|a,1) + G_\pi(b)P(b|a,1) + G_\pi(c)P(c|a,1) \tag{10.55}$$

$$G_\pi(b) = 1 + G_\pi(a)P(a|b,1) + G_\pi(b)P(b|b,1) + G_\pi(c)P(c|b,1) \tag{10.56}$$

$$G_\pi(c) = 0 + G_\pi(a)P(a|c,u_T) + G_\pi(b)P(b|c,u_T) + G_\pi(c)P(c|c,u_T) \tag{10.57}$$

每个方程表示一个不同的状态，并使用 π 中相应的行动。因为应用终止条件，所以最后一个方程变成 $G_\pi(c) = 0$。代入 $P(x'|x,u)$ 的值并用 $G_\pi(c) = 0$ 以后，另外两个方程变为：

$$G_\pi(a) = 1 + \frac{1}{3}G_\pi(a) + \frac{1}{3}G_\pi(b) \tag{10.58}$$

$$G_\pi(b)=1+\frac{1}{3}G_\pi(a)+\frac{1}{3}G_\pi(b) \tag{10.59}$$

解为 $G_\pi(a)=G_\pi(b)=3$。

为每个有 $G_\pi(a)=G_\pi(b)=3$ 和 $G_\pi(c)=0$ 的状态使用式（10.54）来发现一个更好的规划 π'。在状态 a 处，可以通过解如下方程得到:

$$\pi'(a)=\arg\min_{u\in U}\left\{l(x,a)+\sum_{x'\in X}G_\pi(x')P(x'|x,a)\right\} \tag{10.60}$$

最好的行动为 $u=2$，它产生代价 5/2:

$$l(x,2)+\sum_{x'\in X}G_\pi(x')P(x'|x,2)=1+0+(3)\frac{1}{2}+(0)\frac{1}{4}=\frac{5}{2} \tag{10.61}$$

因此 $\pi'(a)=2$。相似地，可以计算 $\pi'(b)=2$，它产生的代价为 7/4。$\pi'(c)=u_T$，它完成了对一个改进规划的确定。

因为已经发现一个改进规划，用 π' 代替 π，返回到步骤 2。新的规划产生方程:

$$G_\pi(a)=1+\frac{1}{2}G_\pi(b) \tag{10.62}$$

$$G_\pi(b)=1+\frac{1}{4}G_\pi(a) \tag{10.63}$$

它们产生 $G_\pi(a)=12/7$ 和 $G_\pi(b)=10/7$。下一步试图用式（10.54）发现一个更好的规划，但是可以断定目前的规划不能进一步改进提高了。那么报告 $\pi^*=\pi$，策略迭代方法终止。■

策略迭代可能看起来比值迭代更可取，特别是因为它通常收敛的迭代次数比值迭代少。确定一个规划的代价的方程也有效地考虑了多个阶段的问题。但是对大多数规划问题来说，X 很大，并且每次迭代时需要求解的大线性系统变得非常难以处理。在某些应用中，状态空间可能非常小，或者在大的状态空间上应用稀疏矩阵技术来解决问题。一般情况下，采用基于值的方法对大多数规划问题来说是更可取的。

10.2.3　图搜索方法

值迭代方法非常具有一般性；但是在一些情况下，大多数时间都消耗在那些不能进一步更新值的状态上，因为要么最优尚需代价已经已知，要么目标还没有到达。策略迭代似乎能够缓解这个问题，但是它局限于小的状态空间。这些不足之处促进人们对其他方法的思考，诸如第 2.2 节的对图搜索方法进行扩展等。在某些情况下，甚至可以扩展 Dijkstra 算法来快速得到最优解，但是对解的结构需要有很强的假设。在非确定的情况下，可以研究只产生可行解的搜索方法，而不需要考虑最优性。对于本节的方法，在第 2.2 节中定义的意义下，不需要 X 是有限的，只要搜索方法是系统性的。

带后向投影的后向搜索　通过使用后向投影从 X_G 向外递增生长一个规划，这样可以引导后向搜索。在非确定的不确定性下计算可行规划一个完整的算法如图 10.6 所示。令 S 表示已经计算了规划的那些状态的集合。初始情况下，$S=X_G$，并且如果可能，S 可以生长直到 $S=X$。对于每个 $x\in X_G$，规划定义从 $\pi(x)=u_T$ 开始并且在执行过程中递增地扩展到新的状态。

后向投影算法：

1. 初始化 $S=X_G$，并且对于每个 $x\in X_G$，令 $\pi(x)=u_T$。

2. 对于每个 $x\in X\setminus S$，如果存在某 $u\in U(x)$ 使得 $x\in \mathrm{SB}(S,u)$，那么：(1)令 $\pi(x)=u$；(2)将 x 插入到 S。

3. 如果步骤 2 扩展 S 失败，那么算法停止。这意味着 $\mathrm{SB}(S)=S$，即没有更进一步的进展。否则，回到步骤 2。

图 10.6 在非确定的不确定性下，计算一个可行规划的一般算法

步骤 2 取得不在 S 中的每个状态，检测它们是否应该加入。这要求确定是否可以从 x 应用某行动 u，保证下一个状态一定落在 S 中，如图 10.7 所示。如果是这样，那么指定 $\pi(x)=u$，并且扩展 S 使之包含 x。如果没有这种进展，那么算法必须终止。否则，返回步骤 2 再次检测每个状态。这是必需的，因为 S 已经增长，并且在下一次迭代时，新的状态可能落在它的强后向投影中。

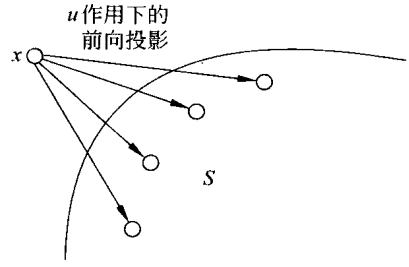


图 10.7 状态 x 可以加到 S，如果存在一个行动 $u\in U(x)$ 使得 S 包含一步前向投影

为更加有效，步骤 2 中的 $X\setminus S$ 集合可以用一个更小的集合 $\mathrm{WB}(S)\setminus S$ 代替，因为对于 X 中的其他状态来说不可能受到影响。根据具体的问题，这个条件为裁剪掉那些毫无希望的状态提供了一种快速方法。例如考虑类似栅格环境，其中有可能在给定时刻在任何方向上移动两步。简单的距离测试就可以在步骤 2 去除掉可能包含在 S 中的一些状态。

只要将状态包含在 S 中是按系统进行的，如第 2.2 节所考虑的，那么图 10.6 中的算法可能发生巨大变化。一种可能是跟踪尚需代价，并且基于递增地插入最小代价状态来生长 S。这就导致了非确定情况下的 Dijkstra 算法。

非确定的 Dijkstra 算法 图 10.8 显示了为解决非确定的不确定性下表述 10.1 的问题而对 Dijkstra 算法的一个扩展。它也可以看作是图 10.6 中算法的一种变形，因为它采用后向投影来生长 S。图 10.8 中的算法表示了 Dijkstra 算法的一种后向搜索版本；因此它保持最坏情况的尚需代价 G，该代价有时会成为最优的、最坏情况的尚需代价 G^*。初始情况下，对目标中的状态来说 $G=0$，对于所有其他状态，$G=\infty$。

非确定的 Dijkstra 算法：

1. 初始化 $S=\phi$ 和 $A=X_G$。将 u_T 和每个 $x\in A$ 联系起来。对所有 $x\in A$ 赋值 $G(x)=0$，对所有其他状态赋值 $G(x)=\infty$。

2. 除非 A 是空的，否则删除使得 G 为最小的 $x_s\in A$ 及其相应的 u。如果 A 是空的，那么算法停止（不可能有更进一步的进展了）。

3. 指定 $\pi^*(x_s)=u$ 作为最优规划的一部分，并将 x_s 插入到 S 中。宣布 $G^*(x_s)=G(x_s)$。

4. 用式（10.64）为边界集 $\mathrm{front}(x_s,S)$ 中的任意 x 而计算 $G(x)$，并将 $\mathrm{front}(x_s,S)$ 及其相关的行动插入 A 中。对于已经在 A 中的状态来说，保留更低的 G 值，要么是原始值，要么是新计算的值。返回步骤 2。

图 10.8 一个基于 Dijkstra 的算法，它用于计算在非确定的不确定性下的一个最优可行规划

步骤 1 进行初始化。步骤 2 在 A 中选择有最小值的状态。与 Dijkstra 算法中处理确定

性的问题一样，这种状态的尚需代价可能是最小的。因此在步骤 3 中宣布 $G^*(x_s)=G(x_s)$，且 π^* 被扩展到包含 x_s。

步骤 4 为某些状态更新代价，并且扩大行动集合 A。将 x_s 插入到 S 会使哪个代价立刻受到影响？这是状态 $x_k \in X \setminus S$，对这些状态来说，存在产生一步前向投影 $X_{k+1}(x_k,u_k)$ 的某 $u_k \in U(x_k)$ 使得：（1） $x_s \in X_{k+1}(x_k,u_k)$；（2） $X_{k+1}(x_k,u_k) \subseteq S$。如图 10.9 所示，令满足这些约束的状态的集合称为边界集，用 $\text{front}(x_s,S)$ 表示。对于每个 $x \in \text{front}(x_s,S)$，令 $U_f(x) \subseteq U(x)$ 表示所有行动的集合，对于这些行动来说前向投影满足前面两个条件。

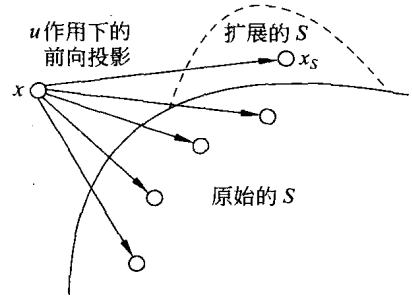


图 10.9　计算任意状态 x 的最坏情况的尚需代价使得存在一个行动 $u \in U(x)$，对该行动来说在更新过的 S 中包含一步前向投影，并且前向投影中的一个状态是 x_s

边界集可以用后向投影来说明。弱后向投影 $\text{WB}(x_s)$ 产生所有可能一步到达 x_s 的状态。但是对于 $\text{SB}(S)$ 中的状态来说尚需代价是有限的（这里的 S 已经包含 x_s）。应该排除 S 中的状态，因为它们的最优代价已知。这将候选边界状态的集合减少到 $\big(\text{WB}(x_s) \cap \text{SB}(S)\big) \setminus S$。该集合依然太大，因为相同的行动 u 必须产生一个一步前向投影，该投影包含 x_s 且是 S 的一个子集。

为所有 $x \in \text{front}(x_s,S)$，计算最坏情况下的尚需代价：

$$G(x)=\min_{u \in U_f(x)} \left\{ \max_{\theta \in \Theta(x,u)} \{l(x,u,\theta)+G(f(x,u,\theta))\} \right\} \tag{10.64}$$

其中用到受限行动集合 $U_f(x)$。如果 x 已经在 A 中且已经计算以前的 $G(x)$，那么保留它以前的最小值和式（10.64）。

概率 Dijkstra　Dijkstra 算法的概率版本一般情况下是不存在的；但是在某些问题中，它却是可用的。用如下方程代替式（10.64），图 10.8 中的算法就可以被修改，以适应概率情况：

$$G(x)=\min_{u \in U_f(x)} \left\{ E_\theta \left[l(x,u,\theta)+G(f(x,u,\theta)) \right] \right\} \tag{10.65}$$

边界的定义相同，非确定的前向投影仍然可以应用于概率情况。在转移图中有非零概率的边被认为是可能的将来的状态。从前向投影中排除零概率的边，因为它们不影响计算的代价值。

如果存在一个规划 π，对它来说，从任意 $x_k \in X$，依概率 1 使得 $G_\pi(x_{k+1}) < G_\pi(x_k)$，那么可以应用 Dijkstra 算法的概率版本。这个条件意味着什么？从任意 x_k，所有可能的下一个有非零概率的状态必须有更低的代价值。如果所有边的代价为正，这意味着在多阶段前向投影中的所有路径将朝着目标单调前进。在确定性情况下，如果 $l(x,u)$ 总是正的，那么这总能发生。如果路径有可能是非单调的，那么 Dijkstra 算法终止，因为尚需代价值变化的区域不再包含在传播带内，在确定性问题的 Dijkstra 算法中会出现这种情况。

10.3 无限范围问题

在随机控制理论和人工智能的研究中，到目前所考虑的多数问题没有指明目标集。因此也就没有相关的终止条件。我们的任务是得到在一定阶段（步）数上能够最小化期望代价（或者最大化期望收益）的一个规划。如果阶段数是有限的，那么可以直接应用第 10.2.1 节的值迭代方法。后向值迭代的修改版本当到达第一阶段时就终止了。如果阶段数是无限的，那么问题就更加复杂了。这称为无限范围问题（infinite-horizon problem）。

第 10.1 节所考虑的规划问题的阶段数也是无限的，但是寄期望于如果能够到达目标，那么在有限次迭代后发生终止。如果没有终止条件，那么代价趋近于无穷。有两个可供选择的代价模型可以迫使代价变成有限的。当阶段向将来延伸时，折扣代价模型（discounted cost model）缩小每阶段代价，这为总代价产生了一个能够收敛于一个有限值的几何级数。阶段平均代价模型（average cost-per-stage model）将总代价除以阶段数。这实质上是对积累的代价进行归一化，防止其发散。第 10.2 节的一些计算方法可以进行适应性修改而应用于这些模型。本节将描述这两种无限范围代价模型并提供相应计算方法。

10.3.1 问题表述

设计前面提到的两种代价模型以迫使累积的代价为有限的，即使阶段数是无限的。这两种模型是在对表述 10.1 中代价泛函进行稍微修改的基础上产生的。

如下表述将在整个第 10.3 节中使用。

表述 10.2（无限范围问题）

1. 一个非空有限状态空间 X。
2. 对于每个状态 $x \in X$，一个有限行动空间 $U(x)$（没有终止行动，它与表述 10.1 不同）。
3. 对于每个 $x \in X$ 和 $u \in U(x)$，有一个有限的大自然行动空间 $\Theta(x,u)$。
4. 有一个状态转移函数 f，对于每个 $x \in X$、$u \in U(x)$ 和 $\theta \in \Theta(x,u)$，函数产生状态 $f(x,u,\theta)$。
5. 用 k 表示的一个阶段集合，它从 $k=1$ 开始并无限继续。
6. 随阶段相加的代价泛函 $L(\tilde{x},\tilde{u},\tilde{\theta})$。其中 $\tilde{x}$、$\tilde{u}$ 和 $\tilde{\theta}$ 分别表示无限状态、行动和大自然历史。稍后给出有两种可选形式的 L。

与表述 10.1 比较而言，这里没有初始状态或目标状态。因此没有终止行动。由于没有指定目标集合，这好像是一种很奇怪的规划。反馈规划 π 也采用相同的形式；$\pi(x)$ 为每个 $x \in X$ 产生一个行动 $u \in U(x)$。

有一个可能的应用，假设工厂中的一个机器人将材料从多个可能的起始地址搬运到多个目标地址。机器人进行长的移位操作，它有关于搬运材料的请求什么时候到来的一个概率模型。表述 10.2 可以用来定义目标是最小化材料等待搬运的平均时间这样的问题。策略不应该依赖于移位的长度；因此阶段数可以是无限的。如果移位太短的情况，机器人可能只能进行一种搬运，或者它没有足够的时间来进行搬运。

折扣代价　在表述 10.2 中，必须仔细定义第 6 项中的代价泛函来保证总能够得到有限值，即使阶段数趋近于无穷。通过在将来阶段中快速减少代价，折扣代价模型提供了一种简单的实现方法。它的定义是基于标准几何级数。对于任意实数 $\alpha \in (0,1)$：

$$\lim_{K\to\infty}\left(\sum_{k=0}^{K}\alpha^k\right)=\frac{1}{1-\alpha} \tag{10.66}$$

在最简单的情况下，$\alpha=1/2$ 产生 1+1/2+1/4+1/8+…，这收敛于 2。

令 $\alpha \in (0,1)$ 表示一个折扣因子，它被应用于代价泛函的定义中：

$$L(\tilde{x},\tilde{u},\tilde{\theta})=\lim_{K\to\infty}\left(\sum_{k=0}^{K}\alpha^k l(x_k,u_k,\theta_k)\right) \tag{10.67}$$

令 l_k 表示在第 k 步的代价 $l(x_k,u_k,\theta_k)$。为方便起见，第一步 $k=0$，而不是以前所用到的 $k=1$。当最大步数 K 增加时，将来的代价的重要性明显减弱，如图 10.10 所示。

步骤	L_K^*
K=0	l_0
K=1	$l_0+\alpha l_1$
K=2	$l_0+\alpha l_1+\alpha^2 l_2$
K=3	$l_0+\alpha l_1+\alpha^2 l_1+\alpha^3 l_3$
K=4	$l_0+\alpha l_1+\alpha^2 l_2+\alpha^3 l_3+\alpha^4 l_4$
⋮	

图 10.10　代价的数量随阶段呈指数下降

代价减小的速度很大程度上依赖于 α。例如，如果 $\alpha=1/2$，那么代价减小非常快。如果 $\alpha=0.999$，那么代价会非常慢地收敛到零。一种折中是用一个大的 α，并用更多步数，这样设计的规划质量通常非常高。如果用很小的 α 值，诸如值迭代这样的方法，收敛速度非常快；但是由于“近视”的缘故，解的质量可能很差。

式（10.67）中的 $l(x_k,u_k,\theta_k)$ 项对于不同的 x_k、u_k 和 θ_k 会呈现不同的值。因为这些值是有限的，那么它们一定以某正常数 c[1] 为界。因此：

$$\lim_{K\to\infty}\left(\sum_{k=0}^{K}\alpha^k l(x_k,u_k,\theta_k)\right)\leqslant\lim_{K\to\infty}\left(\sum_{k=0}^{K}\alpha^k c\right)\leqslant\frac{c}{1-\alpha} \tag{10.68}$$

这意味着正如所期望的那样，$L(\tilde{x},\tilde{u},\tilde{\theta})$ 以上式为界。可以构造一个相似的下界，它保证所产生的总代价总是有限的。

阶段平均代价　另外一种代价模型是阶段平均代价模型，通过除以总阶段数，它总能够保持累积代价为有限：

$$L(\tilde{x},\tilde{u},\tilde{\theta})=\lim_{K\to\infty}\left(\frac{1}{K}\sum_{k=0}^{K-1}l(x_k,u_k,\theta_k)\right) \tag{10.69}$$

使用最大的阶段代价界限 c，式（10.69）的增长不会超过 c，甚至当 $K\to\infty$。这种模型有时是很受青睐的，因为代价不依赖于具有随意性的参数 α。

10.3.2　求解技术

第 10.2 节的值迭代和策略迭代方法经过简单修改就可以应用于无限范围问题。这里给出这些内容，但是还存在关于它们的收敛性和数值稳定性等另外一些重要的问题[98]。这些算法都有其他的变化形式，由于这些变化过于复杂，这里将不予考虑，但它们是非常重

1　状态空间 X 甚至可能是无限的，但是这要求可能代价的集合是有界的。

要的，因为它们讨论了关于收敛性和数值稳定性等问题。本节的重点在于理解第 10.2 节和第 10.3 节所考虑问题之间的联系。

折扣代价的值迭代 采用第 10.2.1 节给出的方法将给出后向值迭代解。为书写方便，令第一阶段为 $k=0$ 使得 α^{k-1} 可以用 α^k 代替。在概率情况下，期望的最优尚需代价为：

$$G^*(x)=\lim_{K\to\infty}\left(\min_{\tilde{u}}\left\{E_{\tilde{\theta}}\left[\sum_{k=1}^{K}\alpha^k l(x_k,u_k,\theta_k)\right]\right\}\right) \tag{10.70}$$

期望是在所有大自然历史上进行的，每个历史都是大自然行动的一个无穷序列。非确定的情况下的相应表达式为：

$$G^*(x)=\lim_{K\to\infty}\left(\min_{\tilde{u}}\left\{\max_{\tilde{\theta}}\left\{\sum_{k=1}^{K}\alpha^k l(x_k,u_k,\theta_k)\right\}\right\}\right) \tag{10.71}$$

因为概率情况更加普遍，所以这里介绍它。非确定的版本可以用相似的方法进行处理（见习题 17）。与以前一样，我们将进行后向值迭代，因为它们表达起来更简单。折扣因子所产生新的问题并不严重，但必须解决这些问题才能使得动态规划的递归正确进行。

一种困难在于阶段索引以 α^k 的形式出现在代价函数中。这意味着第 2.3.1.1 节的移动不变性将不再保持。我们必须仔细安排阶段索引。这是一个问题，因为对于后向值迭代来说，最后一个阶段的索引未知且不重要。

通过增加最大阶段索引：$K=0$，$K=1$，$K=2,\cdots$ 来考虑折扣决策问题序列。观察图 10.10 中相邻代价表达式。那么 $K+1$ 阶段（步）问题的最优代价和 K 阶段（步）问题的最优代价之间有什么区别？在图 10.10 中，$K=4$ 情况下代价的最后 3 项通过用 α 乘以 $K=3$ 情况下的所有项，再加一个新的项 l_0 就可以得到。仅有的不同之处在于从 $K=3$ 情况借来的每个 l_i 的阶段索引需要移动一个位置。一般情况下，K 步优化问题的最优代价可以用作 $K+1$ 步问题的最优代价（如果它们首先乘以 α）。$K+1$ 步优化问题可以通过最优化一步代价与 K 步问题的最优代价乘以折扣因子 α 之后的和来解决。

用如下的动态规划可以得到这种结论。假设 K 固定，代价可以从 0 到 K 递归表达为：

$$G_k^*(x_k)=\min_{u_k\in U(x_k)}\left\{E_{\theta_k}\left[\alpha^k l(x_k,u_k,\theta_k)+G_{k+1}^*(x_{k+1})\right]\right\} \tag{10.72}$$

其中 $x_{k+1}=l(x_k,u_k,\theta_k)$。问题是递归因 α^k 而依赖于 k，这使得它不是平稳的。

进行符号的变换后就可以用图 10.10 所示的相邻代价值的思路。令 $J_{K-k}^*(x_k)=\alpha^{-k}G_k^*(x_k)$。通过颠倒阶段索引的方向来避免指定最后阶段且用因子 α^{-k} 来补偿索引变化。带入式（10.72）产生：

$$\alpha^k J_{K-k}^*(x_k)=\min_{u_k\in U(x_k)}\left\{E_{\theta_k}\left[\alpha^k l(x_k,u_k,\theta_k)+\alpha^{k+1}J_{K-k-1}^*(x_{k+1})\right]\right\} \tag{10.73}$$

除以 α^k 且令 $i=K-k$，那么产生：

$$J_i^*(x_k)=\min_{u_k\in U(x_k)}\left\{E_{\theta_k}\left[l(x_k,u_k,\theta_k)+\alpha J_{i-1}^*(x_{k+1})\right]\right\} \tag{10.74}$$

其中 J_i^* 表示有限范围折扣问题在 $K=i$ 时的期望代价。式（10.74）用 J_{i-1}^* 表示 J_i^*，但是给定 x_k，并且等式右边用的是 x_{k+1}。索引运行的方向相反是因为这是带符号变化的后向值迭代，它颠倒了某些索引。x_k 和 x_{k+1} 的特定阶段索引在式（10.74）中并不重要，只要 $x_{k+1}=f(x_k,u_k,\theta_k)$（例如可以进行 $x=x_k$、$x'=x_{k+1}$、$u=u_k$、$\theta=\theta_k$ 这些替换）。

值迭代过程的进行首先对于所有 $x \in X$，令 $J_0^*(x_0)=0$。后续尚需代价函数通过在状态空间上迭代进行式（10.74）来计算。在第 10.2.1 节的避免循环的假设下，由于无限范围的缘故，收敛通常是渐近的。折扣逐渐引起代价之间的区别消失直到这些区别处在允许的范围内。动态规划的稳态形式就是 i 趋近无穷时的极限情况得到的：

$$J^*(x)=\min_{u\in U(x)}\left\{E_{\theta_k}\left[l(x,u,\theta)+\alpha J^*(f(x,u,\theta))\right]\right\} \tag{10.75}$$

如果代价项不依赖于大自然，那么简化形式为：

$$J^*(x)=\min_{u\in U(x)}\left\{l(x,u)+\alpha\sum_{x'\in X}J^*(x')P(x'|x,u)\right\} \tag{10.76}$$

就像第 10.2.1 节一样，最优行动 $\pi^*(x)$ 被指定为在 x 处满足式（10.75）或者式（10.76）的 $u\in U(x)$。

折扣代价的策略迭代　策略迭代方法可以应用于概率折扣代价问题。回忆图 10.4，一般方法是相同的：通过在每次迭代中解线性系统方程，在规划空间上引导一个搜索。在步骤 2 中，式（10.53）用下式代替：

$$J_\pi(x)=l(x,u)+\alpha\sum_{x'\in X}J_\pi(x')P(x'|x,u) \tag{10.77}$$

它是式（10.76）的特殊形式，用来计算一个固定规划。在步骤 3 中，式（10.54）用下式代替：

$$\pi'(x)=\arg\min_{u\in U(x)}\left\{l(x,u)+\alpha\sum_{x'\in X}J_\pi(x')P(x'|x,u)\right\} \tag{10.78}$$

策略迭代算法进行的过程与第 10.2.2 节介绍的相同。

对阶段平均代价模型的求解　不除以 K 就可以得到阶段平均代价模型的一种值迭代算法。如果所选择的行动能够最优化总代价，那么它也能够最优化阶段数接近无穷时的平均代价。这可能引起代价向 $\pm\infty$ 方向增加，但是在实际情况下，只能够进行有限次迭代。

可以采用第 10.2.1 节的后向值迭代方法，但需要一些修改。初始情况下，对于所有 $x_F\in X$，令 $G^*(x_F)=0$。用标准形式计算值迭代：

$$G_k^*(x_k)=\min_{u_k\in U(x_k)}\left\{\sum_{\theta\in\Theta(x_k,u_k)}\left(l(x_k,u_k,\theta_k)+G_{k+1}^*\left(f(x_k,u_k,\theta_k)\right)\right)P(\theta_k|x_k,u_k)\right\} \tag{10.79}$$

迭代过程继续直到收敛。为了确定是否得到了一个质量足够好的解，一个合理的准则是：

$$\max_{x\in X}\left\{\left|G_k^*(x)/N-G_{k+1}^*(x)/(N-1)\right|\right\}<\epsilon \tag{10.80}$$

其中 ϵ 为允许偏差，N 为已经完成的值迭代次数（式（10.80）要求 $N>1$）。一旦满足式（10.80），那么迭代便可以终止。

$G^*(x)$ 的增长会引起数值问题。通过周期地用一个固定因子使所有值减小来保证数值在允许的浮点范围内，这样可以减小数值问题的影响。文献[98]提出了一种称为相对值迭代的方法，它任意地选择一个状态 $s\in X$，并用减去 s 处的代价来表达尚需代价值。这样就使所有值同时得到整理，在保证它们是有界的同时保持算法的收敛性。

图 10.4 给出的方法经过稍微修改后就可以用于进行策略迭代。

10.4 强化学习

10.4.1 一般原理

本节将简要介绍近年来在人工智能领域广泛应用的框架的基本思路。它最早是由机器学习的研究人员所开发和使用的[19,75]，因此本节被称为强化学习(reinforcement learning)。该问题涉及为概率情况下的无限范围问题计算最优规划。基本的思路是将学习概率分布 $P(\theta|x,u)$ 和计算最优规划问题组合到同一个算法中。

术语 在详细介绍该方法之前，需要对已有的名词进行一些说明。首先考虑强化学习这个名词。在机器学习中，多数决策理论模型用收益来代替代价。因此任务是产生决策或者发现规划来最大化一个收益函数。在这种模型下选择好的行动就能够以收益的形式提供正强化。因此采用了强化这个术语。如果采用代价和最小化，那么可选的名词有决策论学习（decision-theoretic learning）和基于代价的学习（cost-based learning）。

学习这个名词与问题有关，因为估计概率分布 $P(\theta|x,u)$ 或者 $P(x'|x,u)$ 显然是一个学习问题。但是需要记住的是也有计算尚需代价函数（或者尚需收益函数）的规划问题和确定能够优化代价（或者收益）的规划问题。因此强化学习这个术语是合理的。

一般框架在文献[99]中被称为神经动态规划（neuro-dynamic programming），因为表述和所产生的算法都是基于动态规划。经常可以得到值迭代的一个变化形式。“神经”指的是能够用来近似规划和尚需代价值的函数族。该术语是特有的，并且对于某些问题（例如在一个小的有限状态空间上），代价值和规划不用近似表示。

文献[97]采用“基于模拟的方法”这个名词，这可能是一个最准确的名称（当用在动态规划中）。因此基于模拟的动态规划或者基于模拟的规划这两个名词很好地反映了这里所说明的框架。“模拟”这个名词来源于这样的事实：用蒙特卡罗方法模拟产生采样点，这些点用来在规划过程中学习所需要的分布。当然你也可以使用你所喜爱的名称，但它们所用的思路是一样的。如果你想阅读相关文献，记住这一点是非常有用的。

一般框架 框架通常应用于存在概率不确定性的无限范围问题。折扣代价模型是最常用的，但是我们几乎都用表述 10.1，因为它更接近于本书的主题。已经假设在表述 10.1 下进行规划时，所有组成部分都是已知的，包括 $P(x_{k+1}|x_k,u_k)$。这可以认为是一个传统的框架，其中有三个阶段。

- 学习阶段：访问 X 中的状态、尝试行动、收集统计特性，这样就可以估计转移概率。当该阶段结束时，环境模型则完全已知了。
- 规划阶段：用诸如值迭代或者策略迭代等方法来计算反馈规划。
- 执行阶段：在学习阶段所处的环境中，机器执行规划。

基于模拟的框架将这三个阶段组合为一个阶段。学习、规划和执行用一个初始时不知道关于状态转移或者代价项的任何信息的机器来进行。理想情况下，机器应该处在一个物理环境中，对于该环境来说马尔可夫模型是成立的。但是在几乎所有实现过程中，机器将连接到蒙特卡罗模拟器，如图 10.11 所示。基于目前的状态，算法向模拟器发送一个行动 u_k，

模拟器按照它的内部概率分布进行采样来计算行动的效果。显然，模拟器的设计者知道转移概率，但是这些没有直接送给规划算法。模拟器向算法回送下一个状态 x_{k+1} 和代价 $l(x_k,u_k)$。

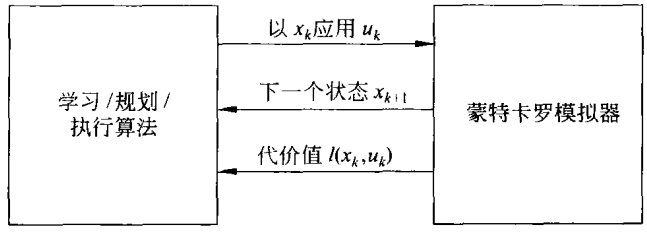


图 10.11　加强学习（或者基于模拟的动态规划）的一般架构

为简单起见，使用 $l(x_k,u_k)$，那么代价不依赖于特定的大自然行动。在这个框架中通常不需要大自然的明确特征。直接学习概率 $P(x_{k+1}|x_k,u_k)$，而不指定大自然的行动。将代价项从 $l(x_k,u_k)$ 推广到 $l(x_k,u_k,x_{k+1})$ 是很平常的，但是为方便起见，这里没有如此。基本思路都一样，即将出现的方程只需要微小变化就可以处理这种推广。

模拟器用来模拟机器与物理世界相互影响的“现实”世界。它代替图 1.16（b）中的环境。这里提出的算法可以认为是图 1.18（b）所示的一个规划。如果学习部分终止，那么所产生的反馈规划可以编程到另外一个机器，如图 1.18（a）所示。但往往不这样进行，因为通常假设机器在它的全寿命期间不停地学习。

一个主要问题是在探索与开发利用之间进行权衡比较[75]。需要对状态空间进行某种重复探索来收集用于可靠地估计模型的足够数据。对于理论收敛来说，每个状态-行动对必须进行无限次尝试。另一方面，应该开发利用关于模型的信息来高效地完成任务。这两个目标经常相互抵触。过于集中于探索将不能最优化代价。过于集中于开发可能会妨碍有用的解的发现，因为没有发现更好的选择。

10.4.2　通过模拟来评估规划

模拟是基于对从采样点中递增得到的信息进行平均。假设你得到一个代价序列 $c_1,c_2,\cdots$，并且想递增地计算它们的平均值。但是并未提前告知你采样点的总数是多少，在任意点上都可能要求你报告目前的平均。令 m_i 表示前 i 个采样点的平均：

$$m_i=\frac{1}{i}\sum_{j=1}^{i}c_j \tag{10.81}$$

为了有效地从 m_{i-1} 计算 m_i，用 m_{i-1} 乘以 $i-1$ 恢复总代价，然后加上 c_i，再除以 i：

$$m_i=\frac{(i-1)m_{i-1}+c_i}{i} \tag{10.82}$$

对式（10.82）进行变形得到：

$$m_i=m_{i-1}+\frac{1}{i}(c_i-m_{i-1}) \tag{10.83}$$

现在考虑为某固定规划 π 而计算每个 $x\in X$ 处的期望尚需代价 $G_\pi(x)$ 的问题。如果已知 $P(x'|x,u)$ 和代价 $l(x,u)$，那么可以计算：

$$G_\pi(x)=l(x,u)+\sum_{x'}P(x'|x,u)G_\pi(x') \tag{10.84}$$

但是如果没有这些信息，我们必须依靠模拟器。

从每个 $x\in X$，假设进行了 1000 次试验，到达目标所产生的代价被记录和平均。每次试验都是一个迭代过程，其中用 π 来选择行动，模拟器指示下一个状态及其增量代价。一旦到达目标状态，总和所有代价就产生了对那次试验所测量到的尚需代价（对于所有 $x\in X_G$，这假设 $\pi(x)=u_T$）。如果 c_i 表示第 i 次试验的总代价，那么在 i 次试验上的平均 m_i 提供了对 $G_\pi(x)$ 的一个估计。当 i 趋近无穷时，我们期望 m_i 收敛到 $G_\pi(x)$。更新式（10.83）可以用来维持尚需代价估计的改进序列。令 $\hat{G}_\pi(x)$ 表示 $G_\pi(x)$ 的目前估计。基于式（10.83）所产生的更新公式可以表达为：

$$\hat{G}_\pi(x):=\hat{G}_\pi(x)+\frac{1}{i}(l(x_1,u_1)+l(x_2,u_2)+\cdots+l(x_K,u_K)-\hat{G}_\pi(x)) \tag{10.85}$$

其中 := 表示编程语言中的赋值。

单个试验可以为多个状态产生更新值[75,98]。假设从 x 进行一次试验，这导致已访问状态序列 $x_1=x,x_2,\cdots,x_k,\cdots,x_K,x_F$。对于序列中的每个状态 x_k，通过记录从 x_k 到 x_K 累积的代价就可以测量尚需代价值：

$$c_k(x_k)=\sum_{j=k}^{K}l(x_j,u_j) \tag{10.86}$$

在沿着路径所访问的每个状态上利用式（10.85）会更有效率。

即时差分 与其等待直到每次试验结束来计算 $c_i(x_i)$，不如在每个状态 x_i 被访问并从模拟器得到 $l(x_i,u_i)$ 后立刻更新每个状态，这样做是有可能的。这导致了著名的估计尚需代价的方法——即时差分（temporal differences）[930]。它与已经给出的方法很相似，但是更加复杂。它将在这里介绍，因为这些方法频繁地出现在强化学习的文献中，并且对它的扩展为更新估计的尚需代价而产生了非常好的基于模拟的方法。

还考虑由试验生成的序列 $x_1,\cdots,x_K,x_F$。令 d_k 表示一个即时差分，它被定义为：

$$d_k=l(x_k,u_k)+\hat{G}_\pi(x_{k+1})-\hat{G}_\pi(x_k) \tag{10.87}$$

注意到 $l(x_k,u_k)+\hat{G}_\pi(x_{k+1})$ 和 $\hat{G}_\pi(x_k)$ 都可以作为 $G_\pi(x_k)$ 的估计。差分就是式（10.87）的右边利用了从模拟器得到的第一步的最新代价，然后为估计其余代价而使用 $\hat{G}_\pi(x_{k+1})$。在这里和后续的表达式中，用规划 $u_k=\pi(x_k)$ 来选择每个行动 u_k。

令 v_k 表示已经访问 x_k 的次数，$1\leqslant k\leqslant K$，包括以前的试验和目前的访问。在试验过程中可以采用如下更新算法。当到达 x_2 时，x_1 处的值更新为：

$$\hat{G}_\pi(x_1):=\hat{G}_\pi(x_1)+\frac{1}{v_1}d_1 \tag{10.88}$$

当到达 x_3，x_1 和 x_2 处的值更新为：

$$\hat{G}_\pi(x_1):=\hat{G}_\pi(x_1)+\frac{1}{v_1}d_2 \tag{10.89}$$

$$\hat{G}_\pi(x_2):=\hat{G}_\pi(x_2)+\frac{1}{v_2}d_2 \tag{10.90}$$

现在考虑到目前在 x_1 处做了些什么。即时差分部分收缩为：

$$\begin{aligned}\hat{G}_\pi(x_1) &:= \hat{G}_\pi(x_1) + \frac{1}{v_1}d_1 + \frac{1}{v_1}d_2 \\ &= \hat{G}_\pi(x_1) + \frac{1}{v_1}(l(x_1,u_1) + \hat{G}_\pi(x_2) - \hat{G}_\pi(x_1) + l(x_2,u_2) + \hat{G}_\pi(x_3) - \hat{G}_\pi(x_2)) \\ &= \hat{G}_\pi(x_1) + \frac{1}{v_1}(l(x_1,u_1) + l(x_2,u_2) - \hat{G}_\pi(x_1) + \hat{G}_\pi(x_3))\end{aligned} \tag{10.91}$$

当到达 x_4 时，可以实现相似的更新。在 x_k 处的更新为：

$$\begin{aligned}\hat{G}_\pi(x_1) &:= \hat{G}_\pi(x_1) + \frac{1}{v_1}d_k \\ \hat{G}_\pi(x_2) &:= \hat{G}_\pi(x_2) + \frac{1}{v_2}d_k \\ &\vdots \\ \hat{G}_\pi(x_k) &:= \hat{G}_\pi(x_k) + \frac{1}{v_k}d_k\end{aligned} \tag{10.92}$$

用这种方法进行更新直到到达 $x_F \in X_G$。现在考虑为每个 x_k 做了些什么计算。即时差分形成了一个收缩和，它像式（10.91）那样在两次迭代后发生收缩。在所有迭代完成以后，x_k 处的值更新为：

$$\begin{aligned}\hat{G}_\pi(x_k) &:= \hat{G}_\pi(x_k) + \frac{1}{v_k}d_k + \frac{1}{v_{k+1}}d_{k+1} + \cdots + \frac{1}{v_K}d_K + \frac{1}{v_F}d_F \\ &= \hat{G}_\pi(x_k) + \frac{1}{v_k}(l(x_1,u_1) + l(x_2,u_2) + \cdots + l(x_K,u_K) - \hat{G}_\pi(x_k) + \hat{G}_\pi(x_F)) \\ &= \hat{G}_\pi(x_k) + \frac{1}{v_k}(l(x_1,u_1) + l(x_2,u_2) + \cdots + l(x_K,u_K) - \hat{G}_\pi(x_k))\end{aligned} \tag{10.93}$$

最后的 $\hat{G}_\pi(x_F)$ 已被删掉，因为假设 π 应用终止行动，所以它的值为零。所产生的最后一个表达式等价于式（10.85），如果序列中每个被访问的状态互不相同。但实际情况往往不如此，那么上面讨论的方法与式（10.85）稍有不同，因为在即时差分的试验过程中计数 v_k 可能发生改变。但这种差别是可以忽略的，即时差分方法计算的估计值收敛到 $\hat{G}_\pi$[98,99]。

到目前为止所介绍的即时差分方法可以用导致更快收敛的方式进行推广。令 $\lambda \in [0,1]$ 为一个指定的参数。$TD(\lambda)$ 即时差分方法用如下方程代替式（10.92）：

$$\begin{aligned}\hat{G}_\pi(x_1) &:= \hat{G}_\pi(x_1) + \lambda^{k-1}\left(\frac{1}{v_1}d_k\right) \\ \hat{G}_\pi(x_2) &:= \hat{G}_\pi(x_2) + \lambda^{k-2}\left(\frac{1}{v_2}d_k\right) \\ &\vdots \\ \hat{G}_\pi(x_{k-1}) &:= \hat{G}_\pi(x_{k-1}) + \lambda\left(\frac{1}{v_{k-1}}d_k\right) \\ \hat{G}_\pi(x_k) &:= \hat{G}_\pi(x_k) + \frac{1}{v_k}d_k\end{aligned} \tag{10.94}$$

这有对从远离 x_k 的状态所得到的代价进行折扣的效果。式（10.92）中的方法是 $\lambda=1$ 的特殊情况，它产生 $TD(1)$。

另外一个有趣的特殊情况是 $TD(0)$，它变成：

$$\hat{G}_\pi(x_k)=\hat{G}_\pi(x_k)+\frac{1}{v_k}(l(x_k,u_k)+\hat{G}_\pi(x_{k+1})-\hat{G}_\pi(x_k)) \tag{10.95}$$

这种形式经常出现在强化学习的文献中（尽管它应用于折扣代价模型）。实验现象表明较低的 λ 值有助于提高收敛速度。对于所有 λ 值的收敛问题将在文献[99]中进行证明。

关于为什么会产生式（10.95），一个直觉的原因是它是随机迭代算法或者 Robbins-Monro 算法的特殊情况[90,99,569]。这是一个一般随机估计技术，它通过采样点序列来解决 $h(y)=y$ 形式的系统。每个采样点表示用蒙特卡罗模拟所得到的 $h(y)$ 的一个测量。这种迭代方法的一般形式是用下式更新 y：

$$y:=(1-\rho)y+\rho h(y) \tag{10.96}$$

其中 $\rho\in[0,1]$ 是一个参数，它的值影响收敛速度。从直觉上看，式（10.96）通过在原始值和 $h(y)$ 的最新采样点之间进行插值来更新 y。这里没有给出该算法的收敛性证明，详细内容参见文献[99]。典型结果是当在模拟过程中有大量噪声时，更小的 ρ 值导致更加可靠的估计，但是这降低了收敛速度。收敛是渐近的，它要求基于规划的状态转移图中的所有边（概率为非零）应该无限次被访问。

得到 $\hat{G}_\pi$ 的一般方法可以从随机迭代框架中通过推广 $TD(0)$ 来产生：

$$\hat{G}_\pi(x)=(1-\rho)\hat{G}_\pi(x)+\rho(l(x,u)+\hat{G}_\pi(x')) \tag{10.97}$$

式（10.95）中的 $TD(0)$ 本质上是在产生过程中选择参数 ρ 的，但是在式（10.97）中可以选择任意 $\rho\in(0,1)$。

更新方程没有考虑转移概率，这可能是不正确的。在模拟过程中应对它们进行考虑，因为最可能发生的转移对式（10.97）会产生非常大的影响。当用来自于分布的采样点对非均匀概率密度函数的均值进行估计时也会发生同样的情况。以更高频率发生的值对平均值的贡献更大，那么频率就自动给它们赋予了相应的权值。

10.4.3 Q-学习：计算最优规划

本节从计算规划转移到在基于模拟的框架下计算一个最优规划。最重要的思路是计算 Q-因子 $Q^*(x,u)$。它是对最优尚需代价 G^* 的扩展，G^* 记录状态 $x\in X$ 和 $u\in U(x)$ 的每个可能的组合的最优代价。从状态 x 开始应用 u，然后遵循从下一个状态 $x'=f(x,u,\theta)$ 所产生的最优规划，这样所产生的期望代价就是 $Q^*(x,u)$。如果 u 正好是最优规划 $\pi^*(x)$ 所选择的行动，那么 $Q^*(x,u)=G^*(x)$。因此如果第一阶段进行任意选择，后来才进行最优决策，那么 Q-值可以认为是这样所需的代价。

值迭代 从 Q-因子可以构造值迭代的基于模拟的版本。用 Q-因子代替 G^* 的原因在于在动态规划中可以避免在 $U(x)$ 上所进行的最小化。避免这种最小化可以使能逐采样点的方法，该方法估计最优值并最终得到最优规划。最优尚需代价可以从 Q-因子得到：

$$G^*(x)=\min_{u\in U(x)}\{Q^*(x,u)\} \tag{10.98}$$

这使得式（10.46）中的动态规划可以表达为：

$$Q^*(x,u)=l(x,u)+\sum_{x'\in X}P(x'|x,u)\min_{u'\in U(x')}\left\{Q^*(x',u')\right\} \tag{10.99}$$

将式（10.98）代入式（10.99）的右侧，那么它也可以用 G^* 表示为：

$$Q^*(x,u)=l(x,u)+\sum_{x'\in X}P(x'|x,u)G^*(x') \tag{10.100}$$

如果已知 $P(x'|x,u)$ 和 $l(x,u)$，那么式（10.99）将导致一个密集存储的方法来进行值迭代。收敛以后，式（10.98）可以用来得到 G^* 的值。最优规划构造为：

$$\pi^*(x)=\arg\min_{u\in U(x)}\left\{Q^*(x,u)\right\} \tag{10.101}$$

因为代价和转移概率未知，所以需要基于模拟的方法。可以再次应用随机迭代算法的思路。回忆式（10.97）用单独的采样点来估计规划的代价，并且要求一个收敛速率参数 ρ。这里采用相同的思路，那么可以产生值迭代的基于模拟的版本：

$$\hat{Q}^*(x,u):=(1-\rho)\hat{Q}^*(x,u)+\rho\left(l(x,u)+\min_{u'\in U(x')}\left\{\hat{Q}^*(x,u)\right\}\right) \tag{10.102}$$

其中 x' 是下一个状态，并且 $l(x,u)$ 是当 u 应用于 x 时从模拟器得到的代价。初始情况下，所有 Q-因子被置为零。到达目标的采样点轨迹可以用模拟来产生，式（10.102）应用于每步产生的状态和代价。再次强调，更新方程可能是不正确的，因为没有明确提及转移概率，但是这在整个模拟中自动予以考虑。

在多数文献中，Q-学习应用于折扣代价模型。这产生了式（10.102）的变化形式：

$$\hat{Q}^*(x,u):=(1-\rho)\hat{Q}^*(x,u)+\rho\left(l(x,u)+\alpha\min_{u'\in U(x')}\left\{\hat{Q}^*(x,u)\right\}\right) \tag{10.103}$$

其中出现了折扣因子 α，因为更新方程从式（10.76）产生。

策略迭代　一个基于模拟的策略迭代算法可以用 Q-因子产生。回忆第 10.2.2 节，我们需要方法来：（1）计算给定的规划 π；（2）通过选择更好的行动来对规划进行改进提高。以前计算规划涉及求解线性方程。现在任何规划 π 都可以用第 10.4.2 节的方法进行计算，而不需要知道 $P(x'|x,u)$。一旦从每个 $x\in X$ 可靠地计算出 $\hat{G}_\pi$，那么可以用进一步的模拟来计算每个 $x\in X$ 和 $u\in U$ 的 $Q_\pi(x,u)$。这可以用定义式（10.100）的一个被约束到 π 的版本来完成：

$$Q_\pi(x,u)=l(x,u)+\sum_{x'\in X}P(x'|x,u)G_\pi(x') \tag{10.104}$$

不需要知道转移概率。Q-因子可用模拟和平均来计算。进行如下设置来改进规划：

$$\pi'(x)=\arg\min_{u\in U(x)}\left\{Q^*(x,u)\right\} \tag{10.105}$$

它基于式（10.98）。

10.5　序贯对策论

到目前为止，序贯决策过程只涉及与大自然之间的对策。在本节引进其他决策者。第 9.3 节和第 9.4 节中的单阶段对策及其平衡点的概念将扩展到对策序列中。第 10.5.1 节引

入用对策树表示的序贯零和对策，它有助于对概念的形象化。第 10.5.2 节讨论用状态空间表示的序贯零和对策。第 10.5.3 节简要介绍其他对策，包括非零和对策和涉及大自然的对策。该节中的表示称为序贯对策论（sequential game theory）。另外一个常见的名称是动态对策论[60]。如果有第 13.5 节所考虑的阶段的一个连续统，那么就可得到微分对策论[60,480,786,985]。

10.5.1 对策树

在多数文献中，序贯对策以对策树的形式进行表示。第 10.5.2 节将提出一个状态空间表示，它与本章使用的表示是一致的。树的表示是对策的扩展形式（与标准形式相对应，标准形式是第 9 章所使用的代价矩阵表示）。这种表示有助于对策论中一些问题的形象化。它可能对形象化信息状态最有帮助，这种情况将保留到第 11.7 节正式介绍信息空间以后。这里提出的对策树是针对很容易描述的情形，而没有深入介绍信息空间。

介绍序贯对策之前，考虑用树的形式表示一个单阶段（步）对策。回忆例 9.14，它是一个零和 3×3 矩阵对策。它可以被表示为一个如图 10.12 所示的对策树。在根部，P_1 有三个选择。在下一级，P_2 有三个选择，基于以上的选择，可以到达 9 个可能的叶子之一。在该点上可以得到代价，它写在叶子的下面。其值是代价矩阵（9.53）的元素项。每个非叶子顶点被称为决策顶点：一个决策者必须选择一个行动。

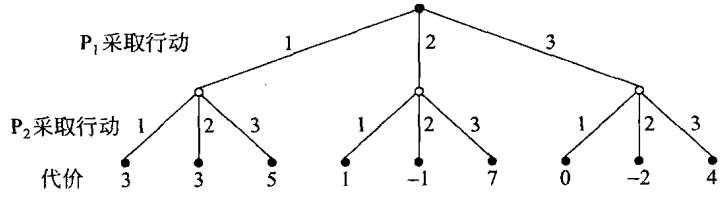


图 10.12 用一个对策树表示的一个 3×3 矩阵对策

如图 10.12 所示，对策有两种可能的解释。

（1）在进行决策之前，P_2 知道 P_1 采取什么行动。这不对应于第 9.3 节的零和对策表述，因为 P_2 似乎与大自然一样强大。在这种情况下，它不等价于例 9.14 中的对策。

（2）P_2 不知道 P_1 采取的行动。这等价于假设 P_1 和 P_2 同时进行决策，这种情况与表述 9.7 一致。树也可以用 P_2 首先行动这种情况来表示。

现在假设 P_1 和 P_2 进行一系列对策。将对策树思路扩展到更多层次就可以定义第 9.3 节中的零和对策的序贯版本。这将建模如下序贯对策。

表述 10.3（以树的形式表示的零和序贯对策）

1. 两个局中人 P_1 和 P_2 轮流进行对策。以前考虑的一个阶段现在展为两个子阶段，其中每子阶段中有一个局中人单独行动。通常假设从 P_1 开始，然后 P_2、P_1、P_2 交替进行对策，直到对策结束。模型反映了许多普通游戏如象棋或者扑克等的规则。令 $\mathcal{K}=\{1,\cdots,K\}$ 表示阶段的集合，在这些阶段中 P_1 和 P_2 轮流进行。

2. 当每个局中人进行对策时，它从非空、有限行动集合中进行选择。可行集合依赖于决策顶点。

3. 对策结束时，基于每个局中人所选择的行动序列就可以得到 P_1 的代价。该代价被阐述为 P_2 的收益。

4. 当进行决策时，必须指定每个局中人所拥有的信息量。这通常用指明行动历史的哪些部分已知来表达。例如，如果 P_1 刚刚进行行动，P_2 知道它的选择吗？它知道以前某个阶段 P_1 所选择的行动吗？

现在详细论述对策树。图 10.13 显示了一个有两个阶段的例子（因此 $K=2$，$\mathcal{K}=\{1,2\}$）。每个顶点对应于一个局中人需要进行决策的点。从顶点发出的每个边代表一个行动。树的根指示对策的开始，这通常意味着 P_1 选择一个行动。树的叶子表示对策的结束，这是产生代价的地方。代价通常显示在每片叶子下面。最后需要关心的是指定每个局中人在决策之前的可用信息。在它自己和其他局中人以前所应用的那些行动中，知道哪些行动？

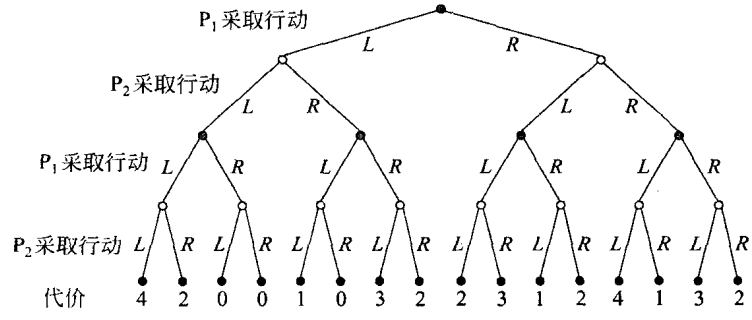


图 10.13　用一个对策树表示的两人两阶段对策

对于图 10.13 中的对策树，有两个局中人和两个阶段。因此有四级决策顶点。局中人的行动集合为 $U=V=\{L,R\}$，分别表示“左”和“右”。因为总有两个行动，所以得到一个二叉树。有 16 个可能的输出，它们分布对应于每个局中人的 4 个两阶段规划的所有成对组合。

对于一个单阶段对策，可以定义确定性的和随机性的策略来得到鞍点。回忆到需要第 9.3.3 节的随机策略来保证鞍点的存在。对于一个序贯对策，这些可以分别扩展到确定性的和随机性的规划。在第 10.1.3 节，一个（确定性的）规划定义为从状态空间到行动空间的映射。每个局中人都可以应用该定义，但是我们必须确定对策树的“状态”是什么。这取决于每个局中人在进行对策时所拥有的可行信息。

第 11.7 节将介绍表示对策树中的信息的一般框架。这里讨论 3 种简单类型的信息。在每种情况下，每个局中人知道自己以前所采取的行动。3 种类型的不同之处在于对其他局中人所采取的行动有不同的认识和了解。这些都定义了用来在规划中进行决策的“状态”。

这里所考虑的 3 种信息模型如下。

交替对策：局中人轮流进行对策，所有局中人知道以前所采取的所有行动。例如国际象棋就是这种情形。为了定义规划，令 N_1 和 N_2 表示所有顶点的集合，从这些顶点 P_1 和 P_2 必须分别进行决策。在图 10.13 中，N_1 是暗色顶点的集合，N_2 是白色顶点的集合。令 $U(n_1)$ 和 $V(n_2)$ 分别是 P_1 和 P_2 的行动空间，它们依赖于顶点。P_1 的一个（确定性的）规划定义为 N_1 上的一个函数 π_1，它为每个 $n_1 \in N_1$ 产生一个行动 $u \in U(n_1)$。同样地，P_2 的一个（确定性的）规划定义为 N_2 上的一个函数 π_2，它为每个 $n_2 \in N_2$ 产生一个行动 $v \in V(n_2)$。对于随机情况，

令 $W(n_1)$ 和 $Z(n_2)$ 分别表示 $U(n_1)$ 和 $V(n_2)$ 上的所有概率分布的集合。P_1 的一个随机规划定义为一个函数，它为每个 $n_1 \in N_1$ 产生某行动 $w \in W(n_1)$。同样地，P_2 的一个随机规划定义为从 N_2 映射到 $Z(n_2)$ 的一个函数。

逐阶段决策：每个局中人知道其他局中人以前所采取的行动；但是没有关于其他局中人目前的行动信息。这意味着在每个阶段局中人同时进行决策。在这种情况下，P_1 的确定的或者随机的规划像在交替对策中那样进行定义；但是 P_2 的规划定义为 N_1 上的函数，而不是 N_2 的函数。这是因为它在进行决策的时刻，P_2 拥有与 P_1 相同的可用信息。P_2 的行动空间必须依赖于 N_1 的元素，而不是 N_2 的；否则，P_2 将不知道什么行动可用。因此它们定义为 $V(n_1)$，对每个 $n_1 \in N_1$。

开环对策：每个局中人不知道其他局中人以前的行动。它们只知道到目前为止采取了多少行动，也就是说对策的阶段。规划定义为阶段集合 $\mathcal{K}$ 上的函数，因为不知道详细的顶点。开环规划只是在确定情况下的一个行动序列（就像第 2.3 节介绍的那样）和随机情况下的概率分布序列。再次，行动空间必须与信息相符。因此它们是 $U(k)$ 和 $V(k)$，对每个 $k \in \mathcal{K}$。

对于如图 10.12 所示的单阶段对策，逐阶段对策模型与开环对策模型是等价的。

10.5.1.1 确定一个安全规划

第 9.3.2 节的安全策略概念自然地扩展到序贯对策。这产生了安全规划(security plan)，其中通过将其他局中人当作非确定的不确定性下的大自然来对待，每个局中人都进行最坏情况分析。一个安全规划和它的代价可以通过从叶子开始传播代价直到根部来计算。图 10.13 的对策中 P_1 的安全规划的计算如图 10.14 所示。在树的第二层到最后一层中的所有顶点处确定 P_2 将采取的行动。因为 P_2 试图最大化代价，在这些顶点处所记录的代价在它孩子的代价上是最大的。在下一个更高层次上，确定 P_1 所选择的行动。在每个顶点处，记录它的孩子中的最小代价。在下一层考虑 P_2，如此重复直到到达根部。这时 P_1 所能确保的最低代价已知。这为序贯对策产生了上值（upper value）$\overline{L}^*$。通过提供选择最低代价孩子顶点的行动，就可以为每个 $n_1 \in N_1$ 定义安全规划。如果 P_2 理性地响应 P_1 的安全规划，那么将会沿着图 10.14（d）中用粗线显示的路径。P_1 安全策略的执行为 P_1 产生了行动序列 (L,L)，为 P_2 产生了行动序列 (R,L)。上值为 $\overline{L}^*=1$。

相似地，也可以计算 P_2 的一个安全策略；但是决策的顺序必须交换。这并不容易形象描述，除非在树中交换局中人的顺序。如果 P_2 首先行动，那么所产生的树如图 10.15 所示。叶子上的代价以不同的顺序出现；但是对于 P_1 和 P_2 所选择的相同行动序列，那么在对策结束时所得到的代价与图 10.14 中所得到的代价相同。为对策产生的下值为 $\underline{L}^*=1$。通过给每个 $n_2 \in N_2$ 分配能够最大化它孩子的代价值的行动来定义所产生的安全规划。如果 P_1 理性地响应 P_2 的安全规划，那么 P_1 执行的行动将是 (L,L)，P_2 执行的行动将是 (R,L)。注意到这与执行 P_1 的安全策略所得到的行动相同，即使它们在树中是不同的，因为交换了局中人的顺序。但是在一些情况下会得到不同的行动序列。

就像单阶段对策的情况，$\underline{L}^*=\overline{L}^*$ 意味着对策有一个确定的鞍点，并且序贯对策的值为 $L^*=\underline{L}^*=\overline{L}^*$。这种独特的对策有唯一一个确定的鞍点。这为局中人产生了相同的、可预测的选择，即使它们各自进行最坏情况分析。

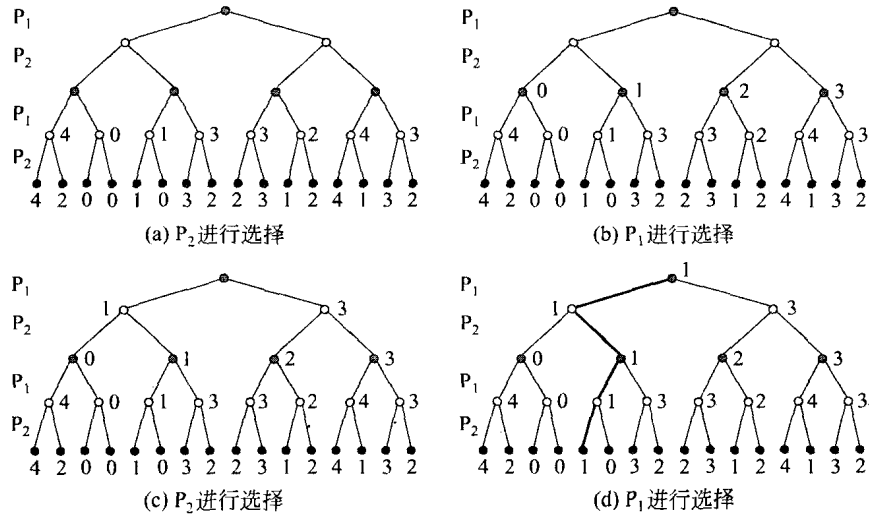


图 10.14　从叶子向上传播代价就可以确定 P_1 的安全规划；安全规划中所涉及的选择被显示在最后一个图中；得到对策的一个上值

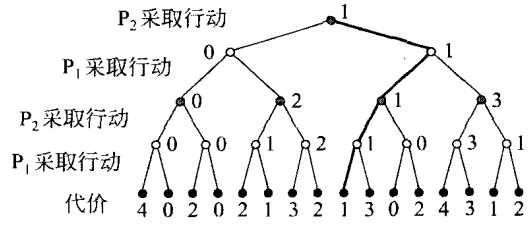


图 10.15　交换 P_1 和 P_2 的序贯就可以发现 P_2 的安全策略（叶子上的代价的顺序也被重新安排）

如果能够认识到什么时候不需要探索树的某些部分，那么就可以大大减小计算安全策略所需要的开销，因为探索那些部分并不能使代价得到改进。这种思路在人工智能中称为 alpha-beta 裁剪（关于这方面的参考以及简要历史见文献[842]的第 186 页和第 187 页）。假设以深度优先的顺序进行树的搜索来确定 P_1 的安全策略。在 P_1 的某决策顶点，假设已经确定：如果应用一个特别行动 u，那么代价 c 将得到确保；但是还有其他行动，对于这些行动来说，它们不知道什么代价将得到确保。考虑为这些剩下的行动之一（用 u' 表示）而确定能够得到确保的代价。这要求计算 P_2 是如何最大化代价来响应 u' 的。当 P_2 有至少一个选择使得代价 c' 比 c 大，那么 P_2 的其他孩子就不需要探索了。为什么？这是因为如果 P_2 的响应所导致的代价比 P_1 选择 u 已经确保的代价更高，那么 P_1 将不会选择 u'。图 10.16 显示了一个简单的例子。这种情形可能发生在树的任何层上，并且当一个行动不需要考虑时，整个子树都可以去除。在其他情形下，P_1 的孩子可以去除，因为 P_2 做出的选择不允许 P_1 改进比 P_2 为它自己所确保的值更低的代价。

10.5.1.2　计算鞍点

P_1 的安全规划组成了在交替对策模型下对策的一个有效解。在每个阶段 P_2 不得不选择对 P_1 的规划的一个最优响应。在逐阶段模型中，“解”的概念是一个鞍点，它发生在上值与下值相等的情况下。刚才描述的过程能够用来确定值及其规划；但是当值不相等时会发生

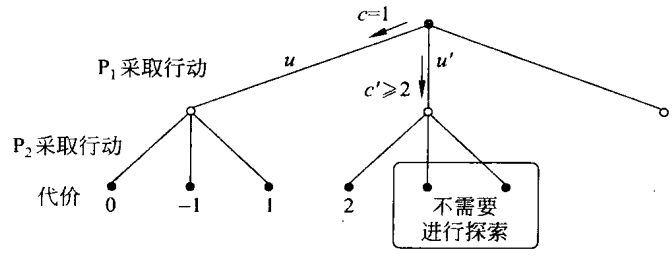


图 10.16 如果以深度优先的顺序来进行探索，那么有些孩子（及其整个子树）不需要探索；这是消除 P_2 的孩子的一个例子；也存在另外一种情况，其中将消除 P_1 的孩子

什么？在这种情况下，应该为局中人研究随机安全规划（randomized security plans）。就像单阶段对策中的情况，得到随机上值 $\overline{\mathcal{L}}^*$ 和随机下值 $\underline{\mathcal{L}}^*$。在随机规划空间中，鞍点总是存在的。这就是说对策总有随机值 $\mathcal{L}^* = \underline{\mathcal{L}}^* = \overline{\mathcal{L}}^*$。这个鞍点可以从下向上进行计算，计算的方法类似于计算安全规划所用的方法。

回到图 10.13 中的例子。就像以前所指出的那样，该对策实际上有一个确定的鞍点。该例是对这种方法的一个非常有用的阐述，因为任何确定性的规划都可以解释为随机规划的一个特殊情况（所有概率质量放在单个行动上）。考虑图 10.13 的下面四个子树，只使用决策顶点的最后两级就可以得到它们。在每个情况下，P_1 和 P_2 必须平行进行行动来结束序贯对策。每个子树都可以认为是一个矩阵对策，因为两个局中人行动后立刻可以得到代价值。

这导致了刻画图 10.13 中对策的另外一种方法，如图 10.17 所示。决策顶点底部两层用矩阵对策来代替。现在计算每个对策的随机值，并将它放在相应的叶子顶点上，如图 10.18 所示。在该例中，仅剩两层决策顶点。这可以阐述为对策：

$$
\begin{array}{c|cc}
 & V & \\
U & 0 & 1 \\
 & 2 & 3
\end{array}
\tag{10.106}
$$

它有一个值 1，出现在如果 P_1 应用 L 和 P_2 应用 R 时。因此原始序贯对策的解已经通过求解矩阵对策来加以确定，这可以作为得到安全规划的另外一种方法。新的方法的益处在于如果任何矩阵都没有确定的鞍点，那么可以用计算随机值来代替。如果在执行过程中到达相应的决策顶点，局中人必须进行随机策略。

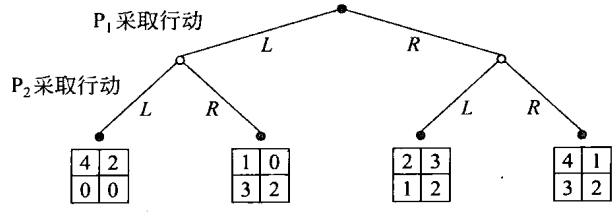


图 10.17 在逐阶段模型下，图 10.13 中的对策可以阐述为一个树，其中每个局中人行动一次，然后它们进行一个矩阵对策来确定代价

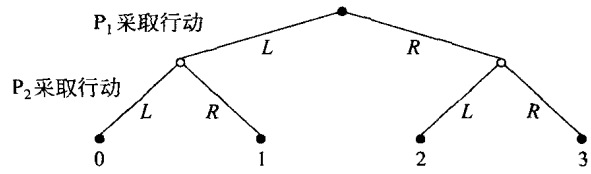


图 10.18　图 10.17 中的每个矩阵都可以用它的随机的值来代替；这里从原始树中剪切了其中的一层；对于这个独特的例子，随机的值也是一个确定的值；注意到，有一个精确的代价值出现在图 10.14（c）中；这是因为每个矩阵对策都有一个确定的值，如果它们没有，那么代价将不吻合

10.5.1.3　将决策树转换到单阶段对策

到目前为止，为交替对策和逐阶段对策模型进行了求解。开环模型还没有解决。在这种情况下，在局中人之间没有信息交换，直到对策结束并得到它们相应的代价。因此假设局中人所进行的这种序贯对策等价于进行一个大的单阶段对策。回顾到开环模型的规划是一个 $\mathcal{K}$ 上的函数。令 Π_1 和 Π_2 分别表示 P_1 和 P_2 可能的规划的集合。对于图 10.13 中的对策来说，Π_i 是每个局中人 4 个可能规划的集合，它用如下顺序进行指定：(L,L)、(L,R)、(R,L)、(R,R)。它们可以被安排在一个 4×4 的矩阵中：

$$\Pi_1 \backslash \Pi_2$$

4	2	1	0
0	0	3	2
2	3	4	1
1	2	3	2

（10.107）

这个矩阵对策没有确定的鞍点。非常遗憾，必须解决四维线性规划问题来发现随机值和平衡点。这与其他两种信息模型所得到的解有很大不同。

可以为逐阶段模型下的序贯对策产生矩阵对策的形式。但是在这种情况下，规划空间非常大。对于图 10.13 中的例子，每个局中人有 32 个可能的规划（每个局中人有 5 个决策顶点，在每个顶点上有两种不同的行动；因此 $|\Pi_i|=2^5$，$i=1$ 和 $i=2$）。这导致了 32×32 维的矩阵对策。这种对策的鞍点解应该与已经得到的鞍点解相同。采用树的优势在于将巨大的对策分解成许多小的矩阵对策。问题逐个解决，那么可以节省大量的计算。这是因为将序贯对策分解为小的矩阵对策这种基于树的计算方法中隐含地用到动态规划原理。下一节将详细介绍这种方法与以前的动态规划方法之间的联系，那里考虑的是状态空间上的序贯对策。

10.5.2　状态空间上的序贯对策

前面所介绍的方法存在一个明显的问题是顶点数随阶段数呈指数增加。在某些对策中可能有多个导致相同状态的行动序列。在许多诸如象棋、跳棋这些平常的游戏中，这种情况是存在的。在这种情况下，如果定义的状态空间能够获取唯一对策的完备集合，那将是

非常方便的。然后局中人的行动对状态进行变换。如果有导致相同状态的不同行动序列，那么不需要用多个分离的顶点。通过声明表示相同状态的顶点是等价的，就可以将对策树转换为对策图。对策图在许多方面与第 10.1 节讨论的转移图相似，那里进行的是与大自然之间的序贯对策。当有对立的局中人时，可以采用相同的思路。

通过将对策树缩减为对策图，我们将介绍在状态空间上所进行的序贯对策。我们也允许发生在任意转移边上更加一般的代价情况，这与原始对策树只有叶子这种情况不同。这里只对对策树的逐阶段模型进行推广。其他信息模型的推广将在第 11.7 节进行。在如下的表述中，P_2 可以看作是表述 10.1 中的大自然。新的表述也是对表述 9.7 单阶段零和对策的推广。为了使概念更加简单，假定所有的空间都是有限的，表述如下所示。

表述 10.4（状态空间上的序贯零和对策）

1. 两个局中人 P_1 和 P_2。

2. 一个有限的非空状态空间 X。

3. 对于每个状态 $x \in X$，P_1 有一个有限非空行动空间 $U(x)$。

4. 对于每个状态 $x \in X$，P_2 有一个有限非空行动空间 $V(x)$。为了能够对第 10.5.1 节中的交替对策模型进行扩展，也可以定义 $V(x,u)$，这使得 P_2 的可用行动集合依赖于 P_1 的行动 $u \in U(x)$。

5. 对于每个 $x \in X$、$u \in U(x)$ 和 $v \in V(x)$，有一个状态转移函数 f 产生状态 $f(x,u,v)$。

6. 用 k 表示的 K 个阶段的集合 $\mathcal{K}$，它从 $k=1$ 开始，到 $k=K$ 结束。令 $F=K+1$ 表示应用最后一个行动之后的最后阶段。

7. 初始状态 $x_I \in X$。对于某些问题，可能没有指定初始状态，在这种情况下规划解必须从所有初始状态来发现。

8. 随阶段相加的代价泛函 L。令 $\tilde{v}_K$ 表示到阶段 K 为止的 P_2 历史行动。代价泛函可以应用于状态和行动历史的任何组合来产生：

$$L(\tilde{x}_F,\tilde{u}_K,\tilde{v}_K)=\sum_{k=1}^{K} l(x_k,u_k,v_k)+l_F(x_F) \tag{10.108}$$

假设两个局中人总知道目前的状态。注意到在表述中没有终止行动。每个局中人行动 K 次以后对策终止。也没有对目标集合的直接描述。终止行动和目标集合很容易加入到表述中，但是这里没有考虑它们。行动集合很容易扩展到依赖于阶段，这样产生 $U(x,k)$ 和 $V(x,k)$。本节提出的方法都可以使用。但是为了使得符号更简单，这里没有这样做。

为每个局中人定义规划 每个局中人现在必须有它自己的规划。因为可能不清楚其他决策者将会采取什么行动，所以最好定义一个规划作为从状态到行动的映射。在第 10.1 节中，其他决策者是指大自然，这里则是指一个理性的对手。令 π_1 和 π_2 分别表示 P_1 和 P_2 规划。因为表述 10.4 中阶段数是固定的，形式为 $\pi_1 : X \times \mathcal{K} \to U$ 和 $\pi_2 : X \times \mathcal{K} \to V$ 的与阶段有关的规划是适宜的（回忆第 10.1.3 节中定义的依赖阶段的规划）。它们分别产生一个行动 $\pi_1(x,k) \in U(x)$ 和 $\pi_2(x,k) \in V(x)$。

现在考虑表述 10.4 的不同的解。对于 P_1，确定的规划是一个函数 $\pi_1 : X \times \mathcal{K} \to U$，它为每个状态 $x \in X$ 和阶段 $k \in \mathcal{K}$ 产生一个行动 $u=\pi(x) \in U(x)$。对于 P_2，函数为 $\pi_2 : X \times \mathcal{K} \to V$，它为每个状态 $x \in X$ 和阶段 $k \in \mathcal{K}$ 产生一个行动 $v=\pi(x) \in V(x)$。现在考虑

定义一个随机规划。令 $W(x)$ 和 $Z(x)$ 分别表示 $U(x)$ 和 $V(x)$ 上所有概率分布的集合。P_1 的随机规划为每个状态 $x \in X$ 和阶段 $k \in \mathcal{K}$ 产生某 $w \in W(x)$。同样地，P_2 的随机规划为每个状态 $x \in X$ 和阶段 $k \in \mathcal{K}$ 产生某 $z \in Z(x)$。

序贯对策中的鞍点　通过为每个局中人定义安全策略将会得到鞍点。每个局中人将其他局中人当作大自然，并且如果得到相同的最坏情况值，那么结果就是对策的鞍点。如果值不同，那么需要随机规划来缩小上值和下值之间的差别。

上值和下值现在依赖于初始条件 $x_1 \in X$。这与第 10.5.1 节中所介绍的不同，因为对策树的根是唯一的开始点。

如果从 x_1 应用行动序列 $\tilde{u}_K$ 和 $\tilde{v}_K$，那么重复地使用状态转移函数 f 就可以产生状态历史 $\tilde{x}_K$。从 x_1 而来的上值被定义为：

$$\overline{L}^*(x_1) = \min_{u_1} \max_{v_1} \min_{u_2} \max_{v_2} \cdots \min_{u_K} \max_{v_K} \left\{ L(\tilde{x}_F, \tilde{u}_K, \tilde{v}_K) \right\} \tag{10.109}$$

如果用大自然代替 P_2，那么它将与式（10.33）相同。式（10.109）将式（9.44）推广到多阶段。从 x_1 而来的下值是对式（9.46）的推广，它为：

$$\underline{L}^*(x_1) = \max_{v_1} \min_{u_1} \max_{v_2} \min_{u_2} \cdots \max_{v_K} \min_{u_K} \left\{ L(\tilde{x}_F, \tilde{u}_K, \tilde{v}_K) \right\} \tag{10.110}$$

如果 $\overline{L}^*(x_1) = \underline{L}^*(x_1)$，那么从 x_1 存在一个确定的鞍点。这意味着在每个阶段内 max 和 min 的顺序可以交换。

值迭代　将应用于式（10.33）的推导过程修改应用于式（10.109）就可以得到值迭代方法。这产生了动态规划递归：

$$\overline{L}_k^*(x_k) = \min_{u_k \in U(x_k)} \left\{ \max_{v_k \in V(x_k)} \left\{ l(x_k, u_k, v_k) + \overline{L}_{k+1}^*(x_{k+1}) \right\} \right\} \tag{10.111}$$

它与式（10.39）相似。用它迭代地计算 P_1 的安全规划。用下式计算 P_2 的安全规划：

$$\underline{L}_k^*(x_k) = \max_{v_k \in V(x_k)} \left\{ \min_{u_k \in U(x_k)} \left\{ l(x_k, u_k, v_k) + \underline{L}_{k+1}^*(x_{k+1}) \right\} \right\} \tag{10.112}$$

它是从式（10.110）产生的动态规划方程。

对于最后阶段 F，可以直接从代价函数确定上值和下值：

$$\overline{L}_F^*(x_F) = \underline{L}_F^*(x_F) = L_F(x_F) \tag{10.113}$$

现在计算 $\overline{L}_K^*$ 和 $\underline{L}_K^*$。从每个阶段 x_K，计算式（10.111）和式（10.112）来确定是否 $\overline{L}_K^*(x_K) = \underline{L}_K^*(x_K)$。如果是，那么 $L_K^*(x_K) = \overline{L}_K^*(x_K) = \underline{L}_K^*(x_K)$ 是第 K 阶段的 x_K 的对策值。如果确信对于任意特定 $x_K \in X$，上值和下值都不相等，那么从 x_K 就没有确定的鞍点。而且这将影响更早阶段其他状态的确定鞍点的存在性，在后来的值迭代中将会遇到这些情况。随机规划可以避免发生这种问题，但是优化过程更加复杂，因为需要重复进行线性规划。

现在假设对于所有 $x_K \in X$，$\overline{L}_K^*(x_K) = \underline{L}_K^*(x_K)$。值迭代过程用通常的方式从 $k = K$ 直到 $k = 1$。假设在每个阶段对于所有 $x_k \in X$，有 $\overline{L}_k^*(x_k) = \underline{L}_k^*(x_k)$。注意到 $\overline{L}_{k+1}^*$ 可以写在式（10.111）和式（10.112）中 $\overline{L}_{k+1}^*$ 和 $\underline{L}_{k+1}^*$ 的位置上，因为已经假设上值和下值相等。如果它们不相等，那么该方法将失效，因为需要用随机规划来得到一个随机鞍点。

一旦从每个 $x_1 \in X_1$ 计算了所产生的值，对每个 $k \in \mathcal{K}$ 和状态 $x_k \in X$，P_1 的安全规划 π_1^* 定义为满足式（10.111）中 min 的任意行动 u。同样地，满足式（10.112）中 max 的任意行

动 v 都可以被定义为 P_2 的安全规划 π_2^*。

现在假设一个或多个初始状态不存在确定的鞍点。为了避免遗憾，必须研究随机安全规划。这些问题可以直接从第 9.3.3 节的随机安全策略扩展而来。这里用向量 w 和 z 表示 $U(x)$ 和 $V(x)$ 上的概率分布。概率向量从 $W(x)$ 和 $Z(x)$ 中选择，它们分别对应于 $U(x)$ 和 $V(x)$ 上的所有概率分布的集合。为了简化符号，假设 $U(x)=\{1,\cdots,m(x)\}$ 和 $V(x)=\{1,\cdots,n(x)\}$，其中 $m(x)$ 和 $n(x)$ 是正整数。

式（9.61）和式（9.62）定义了单阶段对策的随机上值和随机下值。这里将该思路推广到序贯对策的随机上值和下值。它们的定义与式（10.109）和式（10.110）相似，除了如下两点不同以外：（1）交替进行的 min 和 max 被行动空间上的概率分布代替；（2）使用期望代价。

应用动态规划原理，产生随机上值：

$$\overline{\mathcal{L}}_k^*(x_k)=\min_{w\in W(x_k)}\left\{\max_{z\in Z(x_k)}\left\{\sum_{i=1}^{m(x_k)}\sum_{j=1}^{n(x_k)}\left(l(x_k,i,j)+\overline{\mathcal{L}}_{k+1}^*(x_{k+1})\right)w_iz_j\right\}\right\} \tag{10.114}$$

其中 $x_{k+1}=f(x_k,i,j)$。相似地，得到随机下值为：

$$\underline{\mathcal{L}}_k^*(x_k)=\max_{z\in Z(x_k)}\left\{\min_{w\in W(x_k)}\left\{\sum_{i=1}^{m(x_k)}\sum_{j=1}^{n(x_k)}\left(l(x_k,i,j)+\underline{\mathcal{L}}_{k+1}^*(x_{k+1})\right)w_iz_j\right\}\right\} \tag{10.115}$$

在许多对策中，代价项可能只依赖于状态：对于所有 $x\in X$、$u\in U(x)$ 以及 $v\in V(x)$，有 $l(x,u,v)=l(x)$。在这种情况下，式（10.114）和式（10.115）简化为：

$$\overline{\mathcal{L}}_k^*(x_k)=\min_{w\in W(x_k)}\left\{\max_{z\in Z(x_k)}\left\{l(x_k)+\sum_{i=1}^{m(x_k)}\sum_{j=1}^{n(x_k)}\overline{\mathcal{L}}_{k+1}^*(x_{k+1})w_iz_j\right\}\right\} \tag{10.116}$$

$$\underline{\mathcal{L}}_k^*(x_k)=\max_{z\in Z(x_k)}\left\{\min_{w\in W(x_k)}\left\{l(x_k)+\sum_{i=1}^{m(x_k)}\sum_{j=1}^{n(x_k)}\underline{\mathcal{L}}_{k+1}^*(x_{k+1})w_iz_j\right\}\right\} \tag{10.117}$$

这与式（10.46）相似，在那里假设 θ_k 没有出现在代价项中。求和本质上是将式（9.57）推广到多阶段情况。如果需要，这还可以写为矩阵乘法，如第 9.3.3 节所述。

值迭代可以在上述方程上进行，这样得到序贯对策的随机值。因为上值和下值总是相同的，所以没有必要检查它们的差异。实际上，在每次计算式（10.114）和式（10.115）（或者它们的简化形式）时最好首先检查 x_k 是否存在确定的鞍点。当它们其中一个不存在时，必须解决第 9.3.3 节中所述的线性规划问题来确定每个局中人的值以及最好随机规划。如果从目前的状态和阶段存在确定的鞍点，那么就不需要进行该过程。

10.5.3 其他序贯对策

到目前为止的第 10.5 节中大多数思路都可以很自然地扩展到其他序贯对策问题中。下面简要介绍这些内容。

序贯对策中的 Nash 平衡点 表述 10.3 和表述 10.4 可以扩展到序贯非零和对策。在对策树中，代价向量写在每个叶子上，向量的一个元素对应一个局中人。在逐阶段模型下，确定的和随机的 Nash 平衡点可以用第 10.5.1 节的方法来计算。这可计算单个 Nash 平衡点。

很难表示所有 Nash 平衡点。通常将对策树分解为许多矩阵对策；但是在每种情况下，必须发现和记录所有 Nash 平衡点以及它们相应的代价，并不是传播树顶的单个代价，而是自下而上传播代价向量的集合以及与每个代价向量相关的行动。就像单阶段对策那样，非允许的 Nash 平衡点可以不予考虑。因此从在计算中所遇到的每个矩阵对策，只有可允许的 Nash 平衡点以及它们的代价应该向上传播。

为 P_1 和 P_2 分别引入代价函数 L_1 和 L_2 就可以对表述 10.4 扩展进行。与对策树的扩展方式相似，值迭代方法也可以进行扩展。为状态和阶段的每个组合必须维持多值向量及其相应行动。它们对应于可允许的 Nash 平衡点。

Nash 平衡点的非唯一性在序贯对策中会引起很大困难。典型地，序贯对策比单阶段对策有更多 Nash 平衡点。因此在规划方法中 Nash 平衡点不太有用。例如在对复杂经济系统的可能结果进行建模时可能更有用。文献[60]对该方面的内容进行了全面论述。

引入大自然　大自然局中人很容易被引入到对策中。例如假设将大自然引入到零和对策中。这样有 3 个局中人：P_1、P_2 和大自然。图 10.19 显示了一个涉及大自然的单阶段零和对策的对策树。假设这 3 个局中人同时行动，这与逐阶段模型相吻合。也可以用其他一些信息模型。假设用概率不确定性来建模大自然，并假设大自然以 1/3 的概率选择左分支、以 2/3 的概率选择右分支。根据大自然所选择的分支，P_1 和 P_2 将进行一个 2×2 矩阵对策。以 1/3 的概率，代价矩阵为：

U \ V	
3	−2
−6	3

（10.118）

以 2/3 的概率，代价矩阵为：

U \ V	
3	−1
6	0

（10.119）

但是 P_1 和 P_2 并不知道它们正在进行的是哪个矩阵对策。如果在多次独立试验中进行对策，那么从期望的意义上可以消除遗憾。令 A_1 和 A_2 分别表示式（10.118）和式（10.119）。定义一个新的代价矩阵 $A=(1/3)A_1+(2/3)A_2$。那么所产生的矩阵为：

U \ V	
3	0
2	1

（10.120）

该矩阵对策有一个确定的鞍点，在该鞍点上 P_1 选择 L（第二行），P_2 选择 R（第一列），这样所产生的代价为 2。这意味着如果对策在许多次试验上进行平均，那么它们可以进行确定的策略来得到期望代价 2。如果矩阵不允许确定的鞍点，那么需要随机策略。但这个

例子不需要随机选择，即使P_1和P_2都与大自然和智能对手进行对策。

也可能有其他一些变化形式。如果大自然建模为非确定的，那么可以形成最坏情况遗憾的矩阵来确定是否可能消除遗憾。可以考虑图10.19中其中一个对策的序贯版本。在每个阶段有3个子阶段，每个子阶段中大自然、P_1和P_2都进行行动。第10.5.1节的从下而上的方法可以用来将树分解为许多单阶段对策。它们的代价可以用相同的方法向上传播到根部得到一个平衡点解。

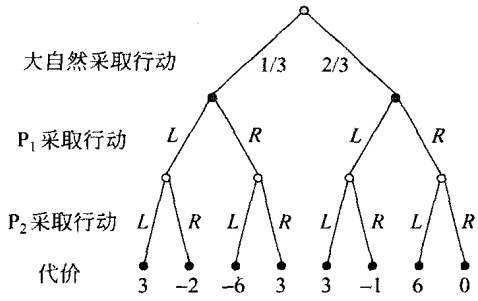


图10.19 这是一个涉及大自然的单阶段、零和对策；它假设所有局中人同时行动

扩展表述10.4来包括状态空间上对策中的大自然。对于每个x，大自然行动集合定义为$\Theta(x)$。状态转移方程定义为：

$$x_{k+1} = f(x_k, u_k, v_k, \theta_k) \tag{10.121}$$

也就是说，下一个状态除了与目前状态有关以外，还取决于3个局中人的行动。通过考虑大自然在动态规划中的影响，那么值迭代方法也可以进行扩展来解决这类问题。例如在概率情况下，在每次迭代中需要在大自然的影响上进行数学期望。所产生的序贯对策通常称为马尔可夫对策[777]。

引入多个局中人 引入更多局中人并不困难。例如，假设n个局中人$P_1, P_2, \cdots, P_n$的集合，它们轮流进行对策。考虑用对策树表示。现在阶段分成n个子阶段，其中每个局中人单独进行行动。假设从P_1开始，然后是P_2，如此进行直到P_n。P_n行动以后，开始下一个阶段，又从P_1开始。局中人循环进行对策直到对策过程结束。可能还有许多不同的信息模型。例如在逐阶段模型中，在目前阶段每个局中人不知道其他$n-1$个局中人所选择的行动。至下而上的计算方法可以用来计算Nash平衡点，但是还会面临Nash平衡点不唯一性的问题。

为每个局中人P_i和状态$x \in X$引入行动集合$U^i(x)$，就可以将表述10.4推广到状态空间情况。令u_k^i表示第k阶段P_i所选择的行动。状态转移方程变成：

$$x_{k+1} = f(x_k, u_k^1, u_k^2, \cdots, u_k^n) \tag{10.122}$$

每个P_i还有一个代价函数L_i。值迭代经过适应性修改来保持多个平衡点和代价向量，那么就可以用来计算Nash平衡点。

10.6 连续状态空间

实际上本章所介绍的所有概念都可以扩展到连续状态空间。这使得它们至少理论上可以应用于位形空间。因此涉及不确定性或者非合作机器人的运动规划问题都可用本章的概念来建模。这种问题也继承了第8章的反馈概念。本节介绍存在大自然所引起的不确定性情况下的反馈运动规划问题。在特定情况下，可以扩展第8.4节和第8.5节的某些方法。反馈规划必须保证在大自然的影响下依然能够到达目标。在第8章的方法中，最容易推广的是第8.5.2节的带插值的值迭代方法。因此它是本节的重点。对于连续状态空间中的对策，参见第13.5节。

扩展值迭代方法

采用第 8.5.2 节的表示方法，从离散问题开始，将各个组成部分连续化。从表述 10.1 开始，令 X 为 $\mathbb{R}^n$ 的一个有界、开子集。假设 $U(x)$ 和 $\Theta(x,u)$ 是有限的。通过第 8.5.2 节的插值就可以直接应用第 10.2.1 节的值迭代方法来计算 X 上的尚需代价。在非确定的情况下，递归为式（10.39），其中 G^*_{k+1} 是在一个有限采样集合 $S \subset X$ 上表示的，并在由插值而成的 $R(S)$ 中其他点上进行计算（在第 8.5.2 节中，$R(S)$ 是 S 的插值区域）。在概率情况下，可使用式（10.45）或式（10.46），但是 G^*_{k+1} 用插值进行计算。

如果 $U(x)$ 是连续的，那么在每次递归中对它进行采样来计算 min，参见第 8.5.2 节。现在假设 $\Theta(x,u)$ 是连续的。在非确定的情况下，可以对 $\Theta(x,u)$ 进行采样来计算式（10.39）中的 max，也可能直接在 $\Theta(x,u)$ 上采用一般优化技术。在概率情况下，期望必须建立在连续概率空间上。概率密度函数 $p(\theta|x,u)$ 可刻画大自然的行动。从这里可以产生概率的状态转移密度函数 $p(x_{k+1}|x_k,u_k)$。使用这些密度函数，式（10.45）和式（10.46）连续版本分别变成：

$$G^*_k(x_k) = \min_{u_k \in U(x_k)} \left\{ \int_{\Theta(x_k,u_k)} \left(l(x_k,u_k,\theta_k) + G^*_{k+1}(f(x_k,u_k,\theta_k)) \right) p\left(\theta_k | x_k,u_k\right) \mathrm{d}\theta_k \right\} \tag{10.123}$$

$$G^*_k(x_k) = \min_{u_k \in U(x_k)} \left\{ l(x_k,u_k) + \int_X G^*_{k+1}(x_{k+1}) p\left(x_{k+1} | x_k,u_k\right) \mathrm{d}x_{k+1} \right\} \tag{10.124}$$

可以用采样来计算积分。一个直接的方法是用离散分布来近似 $p(\theta|x,u)$。例如在一维情况下，这可以通过将 $\Theta(x,u)$ 分割为区间来完成，其中每个区间都被声明为一个离散的大自然行动。与离散的大自然行动有关的概率只是 $p(\theta|x,u)$ 在相关区间上的积分。

第 8.5.2 节描述了连续空间的类 Dijkstra 算法。在那里后向投影式（8.66）判断哪些采样点因远离活动的集合而不能改变它们的（代价）值。同样的原理可以应用于目前情形，但是必须使用弱后向投影式（10.20）。采用弱后向投影，当删除所有那些不在活动集合中的采样点时，可以应用通常的值迭代方法。但是对于许多问题而言，活动集合的大小可能很快变得难以管理，因为弱后向投影所引起的传播过程通常比原始后向投影更快。第 10.2.3 节中的类 Dijkstra 算法也可以推广到连续状态。但是在每次迭代中要求附加条件，必须通过使下一个状态落在 $R(D)$ 中来扩展 D，不管大自然影响与否。

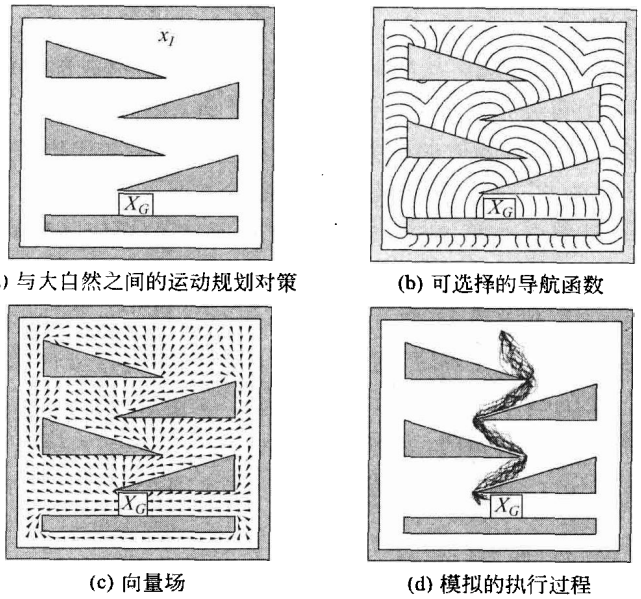
存在大自然影响的运动规划

第 8.5.2 节的带插值的值迭代可以应用于由离散时间近似的运动规划问题。可以将大自然引入离散时间近似中。例如式（8.62）可以用下式代替：

$$x(t+\Delta t) = x(t) + \Delta t(u+\theta) \tag{10.125}$$

其中 θ 从有界集合 $\Theta(x,u)$ 中进行选择。使用式（10.125）就可以进行值迭代。图 10.20 显示了这种模型下采用概率不确定性的一个二维运动规划问题。有趣的是，当从一个固定的初始状态开始执行规划时，每次得到不同的轨迹。但是多次执行后的平均代价接近于期望最优值。

可以构造一个混合系统，其中只允许大自然对模式进行干扰。回忆第 7.3 节的表述 7.3。可以加入大自然来产生如下表述。



(a) 与大自然之间的运动规划对策　(b) 可选择的导航函数

(c) 向量场　(d) 模拟的执行过程

图 10.20 （a）显示了一个二维规划问题，其中大自然是概率的（在角度的一个区间上的均匀分布密度）并且能够干扰运动的方向；在该问题中允许与障碍物发生接触；（b）计算的最优尚需代价（导航）函数的水平集；（c）导航函数产生的向量场；（d）多次试验结果重叠显示在一起[608]

表述 10.5（存在大自然影响的混合系统运动规划）

1. 表述 7.3 中的所有定义，除了转移函数 f_m 和 f。状态表示为 $x=(q,m)$。

2. 对于每个 $x\in X$ 和 $u\in U(x)$，有一个有限的大自然行动空间 $\Theta(x,u)$。

3. 一个模式转换函数 f_m，它为每个 $x\in X$、$u\in U(x)$ 和 $\theta\in\Theta(x,u)$ 产生一个状态 $f_m(x,u,\theta)$。

4. 状态转移函数 f，它是通过改变模式和保持位形固定而从 f_m 产生的。因此 $f((q,m),u)=(q,f_m(q,m,\theta))$（与表述 7.3 不同之处在于已经包括 θ）。

5. 一个无界时间区间 $T=[0,\infty)$。

6. 连续时间代价泛函：

$$L(\tilde{x}_{t_F},\tilde{u}_{t_F})=\int_0^{t_F} l\big(x(t),u(t)\big)\mathrm{d}t+l_F(x(t_F)) \tag{10.126}$$

对于这种混合系统，值迭代过程以相同的方式进行。只需要在位形空间上进行插值。不需要沿着模式“轴”进行插值，因为模式集合已经是有限的。所产生的计算时间随模式的数量线性增长。文献[616]中一个点机器人的二维运动规划如图 10.21 和图 10.22 所示。其中有一个门，这个门被建模为一个平稳马尔可夫过程。位形空间用 40×40 栅格进行采样。有两种模式：门开着或者闭着。因此位形空间有两层，一个模式对应一层。机器人希望最小化到达目标的期望时间。每层的导航函数不能独立计算，因为需要考虑模式的转换概率。例如，如果门几乎总是开着的，那么它的规划将不同于门总是闭着的情况。如果门几乎总是开着的，那么机器人应该向门走去，即使目前门是闭着的，因为稍后它很可能会打开。关于这个例子有许多的变化形式。例如增加更多模式，还可能有其他描述，诸如危险区域

和掩蔽（模式可以被假设为在下雨时，机器人必须走向掩蔽处躲雨）或者要求递送物体等[616,872,873]。

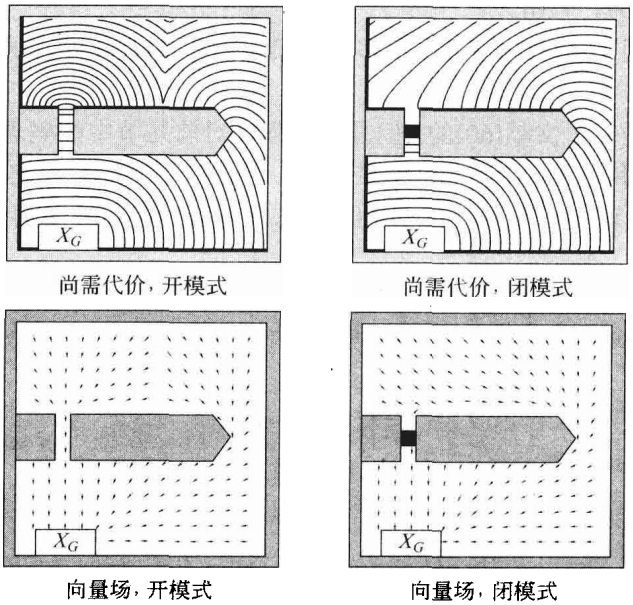


图 10.21　显示了一个随机的混合运动规划问题的最优导航函数水平集及其所产生的向量场；有两种模式，它们分别对应于门是否是关着的；目标是到达底部左边的矩形[616]

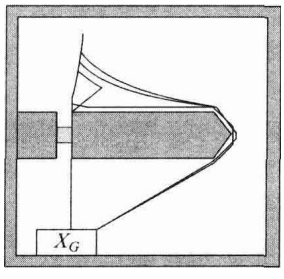


图 10.22　从同一个初始状态开始的几种执行过程；因为门的开闭状态不同，所以每次产生不同的轨迹

进一步阅读

因为本章考虑了单阶段决策问题的序贯版本，所以第 9 章后面所提到的阅读资料也和这里介绍的内容有关。第 10.1 节中的概率表述是随机控制理论的基础问题[97，567]。该框架在人工智能中也很常见[81，269，474，842]。关于随机控制的一个早期的而又很有影响力的工作，见文献[111]，其中涉及与大自然之间的序贯对策的概念。在控制理论中，前向投影和后向投影并不常见，它们是人们从文献[283，315，662]中受到启发而得来的。从表述中除去概率就得到非确定的表述，最坏情况分析也广泛地出现在控制理论中[58，59，303]。在机器人学中采用随机策略的情况见文献[316]。

第 10.2 节是基于经典动态规划，但重点是强调随机最短路径问题。更多值迭代和策略

迭代的内容参见文献[97]。第 10.2.3 节是基于对 Dijkstra 算法的扩展。关于由于对连续问题的近似而产生的收敛性问题，参见文献[94，570，723]。与大自然之间的对策所产生的复杂结果，参见文献[767，770]。

第 10.3 节是受到文献[97]的启发。更多关于第 10.4 节的强化学习的内容，参见文献[19，76，99，896]。

第 10.5 节内容是基于文献[60]的，但是着重强调对前几节中的概念进行统一。文献[60]还论述了连续空间和连续时间上的序贯对策。在连续时间下，它们被称为微分对策（differential games），这将在第 13.5 节介绍。动态规划原理不但扩展到对策论，并且还扩展到 Pareto 优化[244]。

第 10.6 节主要考虑连续状态空间的运动规划。关于将随机最优控制和运动规划结合起来的工作并不多。这里所提及的资料主要是基于文献[602，608，616，869，870]的。

习　题

1. 证明 $SB(S,u)$ 不能表示为所有 $SB(x,u)$ 的并集，$x \in S$。

2. 证明对于任意 $S \subset X$ 和状态转移方程 $x' = f(x,u,\theta)$，它服从 $SB(S) \subseteq WB(S)$。

3. 将第 10.1.2 节的强后向投影和弱后向投影推广到多阶段。

4. 假设用非确定的不确定性，对阶段数没有限制。给定所应用的 π，试确定任意 $k>1$ 阶段的前向投影的表达式。

5. 给出一种计算非确定的前向投影的算法，它使用有二进制项的矩阵。那么你的算法的渐近运行时间和空间是多少？

6. 设计图 10.6 中算法的一种变化形式，它基于可能完成目标，而不是保证完成目标。

7. 用第 10.2.1 节中介绍的推导方法，为非确定的不确定性设计值迭代的一个前向版本。

8. 用第 10.2.1 节中介绍的推导方法，为概率不确定性设计值迭代的一个前向版本。

9. 给出一种直接从基于规划的状态转移图 $\mathcal{G}_\pi$ 来计算概率前向投影的算法。

10. 从某些状态开始，目标可能是可达的，但是不保证可达，扩展第 10.2.1 节的非确定的值迭代方法来检测和处理这些状态。

11. 将式（10.39）推广到在非确定的不确定性下，依赖于阶段的状态转移方程 $x_{k+1} = f(x_k,u_k,\theta_k,k)$ 和代价项 $l(x_k,u_k,\theta_k,k)$ 的情况。

12. 重做习题 11，针对概率不确定性。

13. 将图 10.4 的策略迭代方法扩展到依赖于大自然的代价项 $l(x_k,u_k,\theta_k)$ 这种更一般的情况。

14. 推导一种策略迭代方法，它类似于图 10.4 中的非确定的情形。假设代价项不依赖于大自然。

15. 策略迭代是否能够应用于解决表述 2.3 下的不涉及不确定性的问题？解释在这种情况下会发生什么情况。

16. 显示按照尚需代价，在代价折扣模型下的概率无限范围问题等价于一种特殊的随机最短路径问题（在表述 10.1 下）。[提示：参见文献[97]的第 378 页]

17. 在代价折扣模型和非确定的不确定性下，推导无限范围问题的一种值迭代方法。该方法应该可以计算式（10.71）所给出的尚需代价。

18. 图 10.23 显示了一个用对策树表示的两阶段、零和对策。计算该序贯对策的随机化的值，并给出每个局中人的相应的随机安全规划。

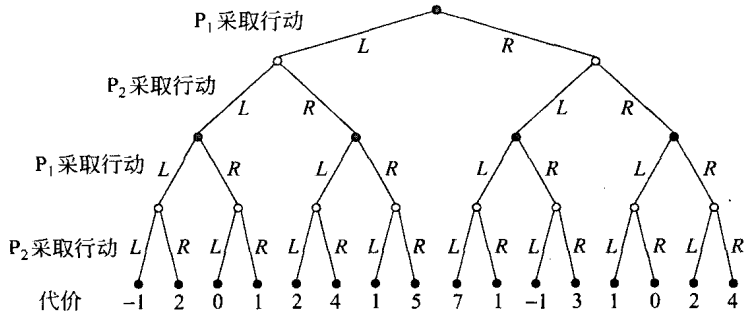


图 10.23　用一个对策树表示的两人两阶段对策

19. 推广对对策树所进行的 alpha-beta 裁剪以使它能够被应用于定义在状态空间上、从一个固定初始状态开始的序贯对策中。

20. 推导式（10.111）和式（10.112）。

21. 将表述 2.4 推广到允许非确定的不确定性。这可以通过指定算子可能效果的集合来完成。

22. 将表述 2.4 推广到允许概率不确定性。对于这种情况，指定算子可能效果的概率。

实现

23. 实现概率后向值迭代，研究图 10.3 的收敛问题。对于那些状态转移图中有许多环的问题来说，这是如何影响问题的性能的？性能是怎样依赖于特定代价和转移概率的？

24. 实现 Dijkstra 算法的非确定的版本并通过一些例子进行检验。

25. 实现并检验 Dijkstra 的概率版本。确保能够满足第 10.2.3 节的 $G_\pi(x_{k+1}) < G_\pi(x_k)$ 条件。研究算法对于那些几乎不满足条件的问题的性能。

26. 用式（10.102）迭代的基于模拟的版本来进行试验。对于某些简单例子，刻画性能是如何依赖于 ρ 的选择的？

27. 实现一个递推算法，它用动态规划来确定一个用逐阶段模型下的一个对策树来表达的序贯对策的上值和下值。

第 11 章　传感器与信息空间

到现在为止我们假设目前的状态都是已知的。如果状态未知，那么会如何？在这种情况下，关于状态的信息是在规划执行过程中由传感器获得的。这种情况在许多涉及与物理世界有相互影响的应用中出现。例如在机器人领域，对机器人来说准确知道它的状态实际上是不可能的，除了在某种受限情况下状态可能已知以外。如果有关于状态的受限信息，那么应该做些什么？一种经典的方法是收集所有可用信息并对状态进行估计。在机器人学中，状态可能包括机器人环境的地图和机器人的位形。如果估计足够可靠，那么我们可以安全地认为没有状态信息的不确定性。这使得许多目前的规划方法不需要任何修正或者较少修正就能够得以应用。

更加有趣的情况发生在当避免进行状态估计的时候。这可能是令人惊奇的，但是可以定义并解决一些重要的任务，而不要求感测特定的状态，即使为规划问题定义了状态空间。为了完成它，规划问题将用信息空间（information space）来表达。感测问题的信息空间与第 4 章的位形空间的意义相同，位形空间是为涉及几何变换的问题而定义的。每个信息空间表示存在感测不确定性的地方。成功地表述和解决这种问题取决于我们操纵、简化和控制信息空间的能力。在某些情况下存在非常好的解，而在其他一些情况下目前可能没有解。在一般情况下有许多与信息空间和感测不确定性有关的问题亟待解决。

图 11.1 所刻画的情形是第 1.4 节所示的问题。假设环境状态未知。关于状态信息，有三种来源：

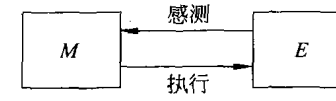


图 11.1　环境的状态未知；能够用来进行推论的信息只有传感器观测历史、已经进行的行动和初始条件；历史变成信息状态

（1）在采取任何行动之前初始条件可以提供有力信息。它甚至可能是给定初始状态的情况。在其他极端状况，初始条件可能不包含信息。

（2）传感器的观测可以提供在执行过程中关于状态的测量。这些测量通常是不完整的或含有对测量的扰动。

（3）在规划中已经执行的行动能够提供关于状态的有价值的信息。例如，如果一个机器人向东移动（除了状态未知以外没有其他不确定性），那么可以预料它的状态比以前的状态更加靠东。因此所应用的行动为推断可能的状态提供了重要线索。

在一般情况下有两种方法使用信息空间：

（1）收集所有可用信息来估计状态。这是经典的方法。认为不再有任何状态不确定性，但是可以证实（或者希望）所产生的规划运行在合理的估计误差下。一个规划一般表示为 $\pi: X \rightarrow U$。

（2）用信息空间来解决完成整个任务。有些任务可能在不知道精确状态的情况下进行。目标和分析都用信息空间来表示，不需要实现特定状态。对一些问题来说，这使得问题得到简化。对一个信息空间 $\mathcal{I}$ 来说，一个规划一般表示为 $\pi: \mathrm{I} \rightarrow U$。

第一种方法具有某种传统的特征，并且一旦定义了合适的估计技术，它可以用第 8 章中的概念来处理。本章关注第二种方法，它代表了表达和解决规划问题的一种强有力的方法。

为简单起见，在许多术语中“信息”用“I”代替。因此信息空间和信息状态分别变成 *I*-空间和 *I*-状态。这类似于将位形空间简写为 *C*-空间。

第 11.1 节～第 11.3 节首先介绍离散状态空间的信息空间。这比连续空间的信息空间更容易表述。第 11.4 节～第 11.6 节将离散状态空间扩展到连续状态空间。在进行连续情况之前，对离散情况的理解是非常重要的。第 11.7 节将信息空间的表述扩展到对策论，其中多个局中人在相同的状态空间上相互影响。在这种情况下，每个局中人都有自己的信息空间，它们的决策就在这些空间上进行。

11.1　离散状态空间

11.1.1　传感器

传感器（sensor）用来感测状态。本节都假设状态空间 X 是有限的或者可数无限的，如表述 2.1 和表述 2.3 所述。一个传感器由两个组成部分：（1）观测空间：传感器可能读数的集合；（2）传感器映射：它刻画读数的特征，如果给定目前状态或者其他信息，那么可以预料将会产生这些读数。需要注意的是在规划模型中，状态并没有给定；而是在对传感器进行建模之后才假设给定状态。这里给出的感测模型是对第 9.2.3 节给出的模型的一般化。在那里，传感器提供关于 θ 的信息，而不是 x，因为第 9 章不需要状态空间。

令 Y 表示观测空间，它是有限的或者可数无限的。令 h 表示传感器映射。将考虑 3 种不同类型的传感器映射，每种都比前一种更复杂和更一般化。

（1）状态传感器映射：在这种情况下，$h: X \to Y$，即给定状态，观测就可以完全确定。

（2）状态-大自然传感器映射：在这种情况下，为每个 $x \in X$ 定义大自然感测行动（nature sensing actions）的一个有限集合 $\Psi(x)$。每个大自然感测行动 $\psi \in \Psi(x)$ 影响传感器的观测。因此状态-大自然映射 h 为每个 $x \in X$ 和 $\psi \in \Psi(x)$ 产生一个观测 $y = h(x, \psi) \in Y$。在规划及其执行过程中，不知道大自然选定的 ψ。但是它作为感测模型的一部分而被指定。

（3）基于历史的传感器映射：在这种情况下，观测可以基于目前状态或者以前的状态。并且可以用大自然感测行动。假设目前阶段为 k。大自然感测行动集合用 $\Psi_k(x)$ 表示，特定的大自然感测行动为 $\psi_k \in \Psi_k(x)$。这产生了非常一般的传感器映射：

$$y_k = h_k(x_1, \cdots, x_k, \psi_k) \tag{11.1}$$

其中 y_k 表示在第 k 阶段得到的观测。映射用 h_k 表示是因为对于每个 k 来说，映射的定义域是不同的。一般情况下，如果需要，任何传感器映射都可能与阶段有关。

这里给出一些传感器的例子。它们用来阐述定义，并为后续 *I*-空间的例子提供基本模块。例 11.1～例 11.6 都涉及状态传感器映射。

例 11.1（奇/偶传感器）　令整数集合 $X = \mathbb{Z}$，$Y = \{0,1\}$。传感器映射为：

$$y = h(x) = \begin{cases} 0 & \text{当 } x \text{ 为偶数时} \\ 1 & \text{当 } x \text{ 为奇数时} \end{cases} \tag{11.2}$$

该传感器的局限性是它仅能够反映 $x \in X$ 是奇数还是偶数。当与其他信息进行组合时，对状态的推断可能足够了，但是一般情况下它提供的信息并不完整。■

例 11.2（取模传感器） 当 x 用某固定整数 k 除时，很容易对例 11.1 进行推广以产生余数。令 $X = \mathbb{Z}$，$Y = \{0,1,\cdots,k-1\}$。那么传感器映射为：

$$y = h(x) = x \bmod k \tag{11.3}$$

■

例 11.3（符号传感器） 令 $X = \mathbb{Z}$，$Y = \{-1,0,1\}$。传感器映射为：

$$y = h(x) = \operatorname{sgn} x \tag{11.4}$$

该传感器将提供非常有限的信息，因为它仅指出状态可能处在边界 $x = 0$ 的哪一边。但是它能够精确确定 x 是否等于 0。■

例 11.4（选择传感器） 令 $X = \mathbb{Z} \times \mathbb{Z}$，$(i,j) \in X$ 表示一个状态，其中 $i,j \in \mathbb{Z}$。假设仅能够观测 (i,j) 的第一个分量。这产生的传感器映射为：

$$y = h(i,j) = i \tag{11.5}$$

显然可以扩展到由笛卡儿乘积所形成的任何状态空间。传感器可能反映一个或者多个分量的值，其余的不可见。■

例 11.5（双射传感器） 令 X 为任意状态空间，令 $Y = X$。令传感器映射为任意双射函数 $h: X \to Y$。这种传感器提供的信息等价于知道状态。因为 h 是双射的，它可以变换为 $h^{-1}: Y \to X$。对于任何 $y \in Y$，用 $x = h^{-1}(y)$ 就可以确定状态。

双射传感器的一种特殊情况是恒等传感器，其中 h 是恒等函数。对于以前章节中所介绍的所有规划问题来说都假设存在这种传感器，因为它能够立即产生状态。任意双射传感器的目的都是如此。■

例 11.6（零传感器） 令 X 为任意状态空间，令 $Y = \{0\}$。定义传感器映射 $h(x) = 0$ 就可以得到零传感器。传感器的数据保持固定，因此不提供关于状态的任何信息。■

这些例子使人们有兴趣基于传感器观测对 X 进行划分。假设一般情况下，h 不是双射的，令 $H(y)$ 表示 X 的如下子集：

$$H(y) = \{x \in X \,|\, y = h(x)\} \tag{11.6}$$

它是 y 的原像。每个 $y \in Y$ 有一个原像，原像的集合形成了对 X 的一种划分。从某些意义上，这表示了传感器的“分辨率”。一个双射传感器将 X 划分为单元素集合，因为它包含完美信息。在其他极端情况下，零传感器将 X 划分为单个集合，也就是 X 自己。符号传感器看起来更有意义，因为它将 X 划分为三个集合：$H(1) = \{1,2,\cdots\}$、$H(-1) = \{\cdots,-2,-1\}$ 和 $H(0) = \{0\}$。选择传感器的原像特别有趣。对于每个 $i \in \mathbb{Z}$，$h(i) = \mathbb{Z}$。原像所导致的划分可能使那些有代数背景的人想起通过同态构造商群的过程[772]。

下面考虑某些涉及状态-行动传感器映射的例子。关于大自然感测行动，有两种不同的模型：

（1）非确定的：在这种情况下，没有关于将选择哪个 $\psi \in \Psi(x)$ 的额外信息。

（2）概率的：已知概率分布。在这种情况下，对每个 $\psi \in \Psi(x)$，已知选择 ψ 的概率 $P(\psi|x)$。

对于影响状态转移方程的大自然行动来说，这两种情况已出现在第 10.1.1 节中。

有时对于给定状态，将状态-行动传感器模型作为 Y 上的概率分布是非常有用的。回忆式（9.28）从 $P(\psi|\theta)$ 到 $P(y|\theta)$ 的转换。这里可以采用相同的思路，即用 X 代替 Θ。如果 h 的定义域限制在某些 $x \in X$，它形成了从 Ψ 到 Y 的一个单射（一对一的）。在这种情况下：

$$P(y|x) = \begin{cases} P(\psi|x) & (\text{唯一}\psi\text{使得}y = h(x,\psi)) \\ 0 & (\text{其他情况}) \end{cases} \tag{11.7}$$

如果去掉单射假设，那么 $P(\psi|x)$ 用所有 ψ 上的和来代替，其中 $y = h(x,\psi)$。

例 11.7（传感器扰动） 令 $X = \mathbb{Z}$，$Y = \mathbb{Z}$ 以及 $\Psi = \{-1,0,1\}$，思路是构造这样一个传感器：如果它不与大自然相互影响，那么它就是恒等传感器。传感器映射：

$$y = h(x,\psi) = x + \psi \tag{11.8}$$

总是知道 $|x-y| \leqslant 1$。因此如果收到传感器的读数 y，下面之一必须为真：$x = y-1$、$x = y$ 或者 $x = y+1$。■

例 11.8（受扰动的符号传感器） 令 $X = \mathbb{Z}$，$Y = \mathbb{Z}$ 以及 $\Psi = \{-1,0,1\}$。令传感器映射：

$$y = h(x,\psi) = \operatorname{sgn}(x+\psi) \tag{11.9}$$

在这种情况下，如果 $y = 0$，那么就不再确切知道是否 $x = 0$。可能 $x = -1$ 或者 $x = 1$。如果 $x = 0$，那么传感器的读数可能为 −1、0 或者 1。■

例 11.9（受扰动的奇偶传感器） 不难构造这样的例子：来自于大自然的适度的干扰破坏了所有的信息。令 $X = \mathbb{Z}$，$Y = \{0,1\}$ 以及 $\Psi = \{0,1\}$。令传感器映射：

$$y = h(x,\psi) = \begin{cases} 0 & (x+\psi\text{是偶数}) \\ 1 & (x+\psi\text{是奇数}) \end{cases} \tag{11.10}$$

若大自然感测行动采用非确定模型，则传感器不能提供关于状态的有用信息。无论如何观测，它都不知道 x 是奇还是偶。但在概率模型下，传感器可能提供一些有用信息。■

假设原像能够得到丰富信息。对于状态-行动传感器映射，原像为：

$$H(y) = \{x \in X | \exists \psi \in \Psi(x), y = h(x,\psi)\} \tag{11.11}$$

与状态传感器比较而言，对于状态-行动传感器映射来说，原像集合更大，并且它们一般不形成 X 的划分。例如例 11.8 的原像为 $H(1) = \{0,1,\cdots\}$、$H(0) = \{-1,0,1\}$、$H(-1) = \{\cdots -2,-1,0\}$。这不是一个划分，因为每个原像都包含 0。如果愿意，可以为每个 $y \in Y$ 直接定义 $H(y)$，而不是明确定义大自然的感测。

最后，给出一个基于历史的传感器映射的例子。

例 11.10（延时观测传感器） 令 $X = Y = \mathbb{Z}$。可以为某固定正整数 i 定义一个延时观测传感器 $(y_k = x_{k-i})$。它指出 i 阶段之前的状态。在这种情况下，它给出了对以前状态值的一个完美测量。也可能有其他形式的延时观测传感器。例如它可能仅给出 i 阶段之前状态的符号。■

11.1.2 历史信息空间

本节定义最基本、最自然的 I-空间。从它可以产生一些其他的信息空间，这是第 11.2 节的内容。假设已经如表述 10.1 那样定义了 X、U 和 f，并且阶段的符号像表述 2.2 那样也进行了定义。那么在规划执行过程中就产生了状态序列 $(x_1,x_2,\cdots)$ 以及行动序列 $(u_1,u_2,\cdots)$。但是在目前情形下，状态序列未知。取而代之的是每个阶段上所得到的观测 y_k。图 11.2 刻画了该过程。

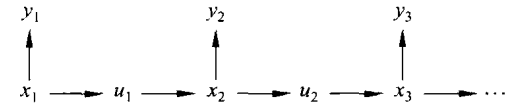


图 11.2 在每个阶段 k，得到观测量 $y_k \in Y$，并且应用行动 $u_k \in U$；但是对于决策者来说，状态 x_k 是隐藏的

在以前的表述中，一般情况下允许行动空间 $U(x)$ 依赖于 x。因为在目前情形下 x 未知，那么行动依赖于 x 似乎很奇怪。这意味着只需要告知哪一个行动可用就能够对状态进行推测[1]。默认情况下假设对于所有 $x \in X$ 来说 U 是固定的。但是在某些特殊情形下，可能允许 $U(x)$ 发生变化。

初始条件 初始条件是提供关于状态信息的 3 个来源之一。因此有 3 种可选的初始条件。

（1）已知状态：给定一个初始状态 $x_1 \subset X$。这就用完美的状态信息对问题进行了初始化。假设大自然行动干扰状态转移方程 f，那么目前状态会存在不确定性。

（2）非确定的：给定一个状态集合 $X_1 \subset X$。在这种情况下，只知道初始状态落在 X 的一个特定子集里。这可以认为是对第一种类型的推广，第一种类型是单元素子集。

（3）概率的：给定 X 上的概率分布 $P(x_1)$。

一般情况下，令 η_0 表示初始状态，它可以是三种类型之一。

历史 假设已经经过 k 个阶段。什么信息是可用的？假设在每个阶段都进行传感器观测。这就产生了感测历史（sensing history）：

$$\tilde{y}_k = (y_1, y_2, \cdots, y_k) \tag{11.12}$$

在每个阶段也可以应用行动，这就产生行动历史（action history）：

$$\tilde{u}_{k-1} = (u_1, u_2, \cdots, u_{k-1}) \tag{11.13}$$

注意到行动历史只进行到 u_{k-1}；如果应用 u_k，那么得到第 $k+1$ 阶段及其状态 x_{k+1}，这都超出目前阶段 k。将感测历史和行动历史进行组合，那么第 k 阶段的历史为 $(\tilde{u}_{k-1}, \tilde{y}_k)$。

历史信息状态 历史 $(\tilde{u}_{k-1}, \tilde{y}_k)$ 与初始条件 η_0 组合在一起形成历史 I-状态，用 η_k 表示。这对应于到第 k 阶段为止所知道的所有信息。尽管状态 $x_1, \cdots, x_k$ 可能未知，但是历史 I-状态总是已知的，因为它们是直接根据可用的信息来定义的。因此历史 I-状态为：

$$\eta_k = (\eta_0, \tilde{u}_{k-1}, \tilde{y}_k) \tag{11.14}$$

当表示 I-空间时，一般情况下我们都忽略嵌套圆括号的问题。例如将式（11.14）当作单个序列来对待，而不是包含两个序列的序列。这对决策来说是无关紧要的。

历史 I-状态 η_k 也可以被表示为：

$$\eta_k = (\eta_{k-1}, u_{k-1}, y_k) \tag{11.15}$$

---

1 研究这个问题非常有趣，但是这里没有予以考虑。

第 k 阶段的历史 I-状态包含第 $k-1$ 阶段所有的历史 I-状态。其中新的信息是最近的行动 u_{k-1} 和目前的传感器观测 y_k。

历史信息空间　历史 I-状态只是所有可能的历史 I-状态的集合。尽管历史 I-状态十分复杂，但是可以将它们抽象地视为一个新空间中的点。为了定义所有可能的历史 I-状态的集合，那么所有初始条件、行动以及观测的集合都必须被精确定义。

所有历史观测的集合用 $\tilde{Y}_k$ 表示，它是从观测空间的 k 个复制的笛卡儿乘积来得到的：

$$\tilde{Y}_k = \underbrace{Y \times Y \times \cdots \times Y}_{k} \tag{11.16}$$

同样地，所有行动历史的集合为 $\tilde{U}_{k-1}$，它是行动空间 U 的 $k-1$ 个复制的笛卡儿乘积。

定义所有初始条件的集合会稍微复杂些，因为可能有 3 种不同类型的初始条件。令 $\mathcal{I}_0$ 表示初始条件空间。根据所采用的 3 种不同类型的初始条件，采用 $\mathcal{I}_0$ 的如下 3 个定义中的一个。

（1）状态已知：如果给定初始状态 x_1，那么 $\mathcal{I}_0 \subseteq X$。在典型情况下，$\mathcal{I}_0 = X$；但是在某些情况下知道不可能有确定的初始状态。因此通常记为 $\mathcal{I}_0 \subseteq X$。

（2）非确定的：如果给定 X_1，那么 $\mathcal{I}_0 \subseteq \text{pow}(X)$（$X$ 的幂集）。在典型情况下，$\mathcal{I}_0 = \text{pow}(x)$；但是也可能知道 X 的特定子集是不可能作为初始条件的。

（3）概率的：如果给定 $P(x)$，那么 $\mathcal{I}_0 \subseteq \mathcal{P}(X)$，其中 $\mathcal{P}(x)$ 是 X 上所有概率分布的集合。

第 k 阶段的历史 I-状态表示为：

$$\mathcal{I}_k = \mathcal{I}_0 \times \tilde{U}_{k-1} \times \tilde{Y}_k \tag{11.17}$$

每个 $\eta_k \in \mathcal{I}_k$ 产生一个初始条件、一个行动历史和一个观测历史。可以认为 I-空间不依赖于 k。这用并集来进行定义（不要错误地认为这是笛卡儿乘积）。如果有 K 个阶段，那么历史 I-空间为：

$$\mathcal{I}_{hist} = \mathcal{I}_0 \cup \mathcal{I}_1 \cup \mathcal{I}_2 \cup \cdots \cup \mathcal{I}_K \tag{11.18}$$

通常，阶段数并不固定。在这种情况下，$\mathcal{I}_{hist}$ 定义为所有 $k \in \{0\} \cup \mathbb{N}$ 上的 $\mathcal{I}_k$ 的并集：

$$\mathcal{I}_{hist} = \mathcal{I}_0 \cup \mathcal{I}_1 \cup \mathcal{I}_2 \cup \cdots \tag{11.19}$$

这种情况与第 7.1 节中的时变运动规划的状态空间有关。历史 I-空间与阶段有关，因为信息随时间累计。在离散模型下，时间隐含在阶段中。因此定义的是依赖阶段的 I-空间。所有这些的并集与第 7.1 节中形成的状态空间相似，在那里将时间作为状态空间的一个轴。对于历史 I-空间 $\mathcal{I}_{hist}$ 来说，阶段索引 k 可以被想象为一个“轴”。

关于历史 I-空间 $\mathcal{I}_{hist}$ 的一个问题是它的 I-状态可能是任意长的，因为历史随阶段线性增长。对现在来说，可以将 $\mathcal{I}_{hist}$ 抽象地想象为另外一种类型的状态空间，而不去关心每个复杂的 $\eta \in \mathcal{I}_{hist}$ 可能是如何表示的。在许多情形下，有两种方法来简化 I-空间，这将在第 11.2 节进行介绍。

11.1.3　定义规划问题

规划问题将直接被定义在历史 I-空间上，这使得 I-空间作为一个平常的状态空间而出现。但它是从另外一个不能得到完整的状态观测的状态空间中产生的。在第 10.1 节，反馈

规划被定义为状态的一个函数。而这里的反馈规划是 I-状态的函数。决策不能基于状态，因为在规划执行过程中状态在一般情况下是未知的。但是 I-状态总是已知的，因此决策基于 I-状态是合理的。

令 π_K 表示一个 K 步信息反馈规划，它是 K 个函数 $\pi_K : \mathcal{I}_k \to U$ 的一个序列 $(\pi_1, \pi_2, \cdots, \pi_K)$。因此在每个阶段 k，I-状态 $\eta_k \in \mathcal{I}_k$ 用作选择行动 $u_K = \pi_K(\eta_k)$ 的基础。由于大自然通过状态转移方程和传感器映射而产生影响，所以由规划 π_K 产生的行动序列 $(u_1, \cdots, u_K)$ 直到规划结束才知道。

如表述 2.3 所示，假设 U 包含了终止行动 u_T。如果在第 k 阶段应用 u_T，那么它永远重复应用。再次假设在终止条件应用之后状态 x_k 保持固定。但是记住 x_k 一般情况下仍然是未知的，它保持固定但是未知。从技术上讲，基于历史 I-空间的定义，在应用 u_T 以后 I-状态必须改变，因为历史增加了。但是可以忽略这些变化，因为在应用 u_T 以后，没有进行新的决策。使用终止条件的规划可以指定为 $\pi = (\pi_1, \pi_2, \cdots)$，因为阶段数在每次规划执行时可能发生改变。使用式（11.19）的历史 I-空间的定义，一个信息反馈规划可表达为：

$$\pi : \mathcal{I} \to U \tag{11.20}$$

定义规划问题还需要代价泛函。就像第 10.1 节所介绍的，代价依赖于状态 $\tilde{x}$ 和行动 $\tilde{u}$ 的历史。规划的表述涉及如下组成部分以及对第 11.1 节介绍的概念的总结（见表述 11.1）。

表述 11.1（离散信息空间规划）

1. 非空状态空间 X，它是有限的，或者可数无限的。

2. 非空、有限行动空间 U。假设 U 包含一个特殊的终止行动，其作用与表述 2.3 中的相同。

3. 对于每个 $x \in X$ 和 $u \in U$，一个有限大自然行动空间 $\Theta(x,u)$。

4. 一个状态转移函数 f，它为每个 $x \in X$、$u \in U$ 和 $\theta \in \Theta(x,u)$ 产生一个状态 $f(x,u,\theta)$。

5. 一个有限的或者可数无限的观测空间 Y。

6. 对于每个 $x \in X$，有一个有限的大自然感测行动空间 $\Psi(x)$。

7. 传感器映射 h，它为每个 $x \in X$ 和 $\psi \in \Psi(x)$ 产生一个观测 $y = h(x,\psi)$。这里的定义给出的是一个状态-大自然传感器映射。也可以使用第 11.1.1 节的状态传感器映射或者基于历史的传感器映射。

8. 用 k 表示的阶段集合，它从 $k=1$ 开始，无限进行下去。

9. 初始条件 η_0，它是初始条件空间 $\mathcal{I}_0$ 的一个元素。

10. 历史 I-空间 $\mathcal{I}_{hist}$，对于每个阶段 $k \in \mathbb{N}$，它是 $\mathcal{I}_0$ 和 $\mathcal{I}_k = \mathcal{I}_0 \times \tilde{U}_{k-1} \times \tilde{Y}_k$ 的并集。

11. 令 L 表示逐阶段相加的代价泛函：

$$L(\tilde{x}_{K+1}, \tilde{u}_K) = \sum_{k=1}^{K} l(x_K, u_K) + l_F(x_{K+1}) \tag{11.21}$$

如果在某阶段 k 应用终止行动 u_T，那么对于所有 $i \geq k$，有 $u_i = u_T$，$x_i = x_k$ 和 $l(x_i, u_T) = 0$。那么可以定义可行规划或者最优规划问题，就像表述 11.1 所介绍的那样，但是这里的规划指定为 $\pi : \mathcal{I} \to U$。

目标集合可以被定义为 $X_G \subset X$。目标也可以表示为历史 I-状态的一个集合。在第 11.2 节之后读者将会看到目标可以用从历史产生出来的 I-状态来表达。

可以对表述 11.1 进行某种扩展，但是我们不这样做是为了简化即将到来的概念的符号。一种扩展是对每个 $x \in X$，允许不同的行动集合 $U(x)$。但是需要小心，因为如果给定行动集合 $U(x)$，就可以推断关于目前状态的信息，并且它随 x 而变化。另外一种扩展是允许代价依赖于大自然，这样得到 $l(x_k, u_k, \theta_k)$，而不是式（11.21）中的 $l(x_k, u_k)$。

一个规划的代价　接下来将第 10.1.3 节中固定规划下，尚需代价的定义扩展到不完美状态信息的情况。考虑评估规划的质量以便能够选择“最好的”一个。假设用非确定的不确定性来建模大自然，并且假设给定非确定的初始条件。如果固定一个规划 π，那么某些状态和行动轨迹是可能的，而另外一些则是不可能的。一般情况下不可能知道将会发生什么历史，但是规划对历史进行了约束。令 $\mathcal{H}(\pi, \eta_0)$ 表示 π 应用于初始条件 η_0 所产生的状态-行动历史的集合。

从初始条件 η_0 应用一个规划 π 后所得到的代价可用最坏情况分析来确定：

$$G_\pi(\eta_0) = \max_{(\tilde{x}, \tilde{u}) \in \mathcal{H}(\pi, \eta_0)} \{L(\tilde{x}, \tilde{u})\} \tag{11.22}$$

$\tilde{x}$ 包含 x_1，但是 $\tilde{x}$ 通常未知。只知道它落在 X_1 中，由 η_0 指定。令 Π 表示所有可能规划的集合。用最坏情况分析所得到的一个最优规划是在所有 $\pi \in \Pi$ 和 $\eta_0 \in \mathcal{I}_0$ 上使式（11.22）最小化的任何规划。如果想得到可行规划，那么通常有大量可选的等价规划。

在概率不确定性下，一个规划的代价可以用期望情况分析来确定：

$$G_\pi(\eta_0) = E_{\mathrm{H}(\pi, \eta_0)}[L(\tilde{x}, \tilde{u})] \tag{11.23}$$

其中 E 表示代价的数学期望，它用 $\mathcal{H}(\pi, \eta_0)$ 上的概率分布。目的是发现一个规划 $\pi \in \Pi$ 最小化式（11.23）。

信息空间只是另外一种状态空间　在本章和第 12 章，重要的是将 I-空间看作一个平常的状态空间。它只是有些特殊，因为它是从另外一个状态空间产生的，但是一旦忽略这些，那么它将展示出平常状态空间的许多特性。其中一个好的特性是这个特殊空间中的状态总是已知的。因此通过将普通状态空间转换到 I-空间，我们也就从拥有不完美状态信息转换到状态总是已知的状况，尽管处在一个大的状态空间中。

那么状态转移方程可以提升到 I-空间来得到信息转移函数 $f_{\mathcal{I}}$。假设没有传感器，因此也就没有观测。在这种情况下，未来的 I-状态是可以预测的，即：

$$\eta_{k+1} = f_{\mathcal{I}}(\eta_k, u_k) \tag{11.24}$$

通过将 u_k 连接到 η_k，函数 $f_{\mathcal{I}}$ 产生 η_{k+1}。

现在假设有观测，但是一般情况下观测是不可预测的。在第 10.1 节中，大自然的行动 $\theta_k \in \Theta(x, u)$ 用来建模不可预测性。按照信息转移方程，y_{k+1} 服务于相同的目的。当决定采用 u_k 时，观测量 y_{k+1} 还未知（就像第 10.1 节中 θ_k 未知一样）。在与大自然之间所进行的有完美状态信息的序贯对策中，x_{k+1} 在下一个阶段直接观测到。对于信息转移方程来说，观测到的是 y_{k+1}，并且可以确定 η_{k+1}。使用历史 I-状态式（11.14）只是将 u_k 和 y_{k+1} 连接到 η_k 中的历史上来得到 η_{k+1}。信息转移方程表达为：

$$\eta_{k+1} = f_{\mathcal{I}}(\eta_k, u_k, y_{k+1}) \tag{11.25}$$

在这里 y_{k+1} 所起的作用与完美状态信息和不可预测未来状态情况下 θ_k 的作用相同。即使大自然使得未来 I-状态不可预测，目前的 I-状态却总是已知的。如果 I-空间被看作一个状态空间，那么一个规划 $\pi : \mathcal{I} \to U$ 现在就是一个状态反馈规划。转移都由 $f_{\mathcal{I}}$ 指定。

新的状态空间中的代价可以从原始代价泛函产生，但是需要在给定当前信息的情况下在所有可能的状态上取最大值或者期望，这将在第 12.1 节介绍。

11.2 衍生的信息空间

历史 I-空间十分复杂。每个 I-状态对应于行动和观测的一个历史。非常遗憾，I-状态序列的长度随阶段数线性增加。为了克服这种困难，通常可以将历史 I-状态映射到更简单的空间。在许多应用中，这对于发现实际规划解来说是非常关键的。在某些情况下，这种简化完全保护了历史 I-空间，也就是说没有损失完备性和最优性。在其他情况下，我们愿意容忍这种简化所带来的对历史 I-空间结构的一些破坏。因为这能够减小 I-空间的大小。

11.2.1 信息映射

假设有一个函数，它将历史 I-空间映射到一个更容易进行管理的空间。正式地，令 $\kappa: \mathcal{I}_{hist} \rightarrow \mathcal{I}_{der}$ 表示从历史 I-空间 $\mathcal{I}_{hist}$ 到衍生 I-空间 $\mathcal{I}_{der}$ 的函数。函数 κ 被称为信息映射或者 I-映射。衍生 I-空间可能是任意集合，因此在定义 I-映射时有很大的灵活性[1]。图 11.3 阐述了这种思路。开始位置是 $\mathcal{I}_{hist}$，可以映射到各种各样的衍生 I-状态。该图显示了一些普通映射 κ_1、κ_2 和 κ_3 以及一些非常重要的类型 $\mathcal{I}_{est}$、$\mathcal{I}_{ndet}$ 和 $\mathcal{I}_{prob}$。最后两个分别是第 11.2.2 节和第 11.2.3 节的内容。还有一个重要的 I-映射是 κ_{est}，它利用历史来估计状态，但是衍生的 I-空间为 X（见例 11.11）。一般情况下，I-映射可以将任何衍生 I-空间映射到另外一个 I-空间，这为任意 I-空间 $\mathcal{I}_{der}$ 和 $\mathcal{I}'_{der}$ 产生了 $\kappa: \mathcal{I}_{der} \rightarrow \mathcal{I}'_{der}$。$I$-映射的合成可以产生一个新的 I-映射。图 11.3 中的衍生 I-空间 $\mathcal{I}_2$ 和 $\mathcal{I}_3$ 是通过合成得到的。

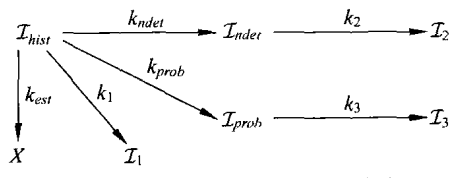


图 11.3 可以提出许多可选的信息映射；每个都能够产生一个衍生信息空间

产生更小的信息反馈规划 I-映射的原始作用是简化规划的描述。在第 11.1.3 节，规划定义为历史 I-空间 $\mathcal{I}_{hist}$ 上的一个函数。假设引入 I-映射 κ，它将 $\mathcal{I}_{hist}$ 映射到 $\mathcal{I}_{der}$。$\mathcal{I}_{der}$ 上的反馈规划定义为 $\pi: \mathcal{I}_{der} \rightarrow U$。为了执行定义在 $\mathcal{I}_{der}$ 上的一个规划，通过将 κ 应用于 η_k 得到 $\kappa(\eta_k) \in \mathcal{I}_{der}$，这样在每个阶段 κ 上计算衍生的 I-空间。π 所选择的行动为 $\pi\big(\kappa(\eta_k)\big) \in U$。

为了理解用 $\mathcal{I}_{der}$ 代替 $\mathcal{I}_{hist}$ 作为 π 的定义域所造成的影响，考虑 $\mathcal{I}_{der}$ 上的规划集合。令 Π_{hist} 和 Π_{der} 分别为 $\mathcal{I}_{hist}$ 和 Π_{der} 上所有规划的集合。任意 $\pi \in \Pi_{der}$ 都可以转换为如下等价规划 $\pi' \in \Pi_{hist}$：对于每个 $\eta \in \mathcal{I}_{hist}$，定义 $\pi'(\eta) = \pi\big(\kappa(\eta)\big)$。

但是并不总能够从某个 $\pi' \in \Pi_{hist}$ 构造规划 $\pi \in \Pi_{der}$。问题是可能存在某 $\eta_1, \eta_2 \in \mathcal{I}_{hist}$，使得 $\pi'(\eta_1) \neq \pi'(\eta_2)$ 和 $\kappa(\eta_1) = \kappa(\eta_2)$。这意味着 Π_{hist} 中的规划要求两个历史引起不同的行动，但是在衍生 I-空间中，无法区分历史。对于 Π_{der} 中的规划，这两个历史必须产生相同行动。

通过产生 $\mathcal{I}_{hist}$ 的一个划分，一个 I-映射 κ 有将 $\mathcal{I}_{hist}$ 收缩为一个更小 I-空间的潜力。对

1 理想情况下，应该映射到 $\mathcal{I}_{der}$，但是为了使某些定义更加容易，这里不做这种要求。

于每个 $\eta_{der} \in \mathcal{I}_{der}$，令原像 $\kappa^{-1}(\eta_{der})$ 定义为：

$$\kappa^{-1}(\eta_{der}) = \{\eta \in \mathcal{I}_{hist} \,|\, \eta_{der} = \kappa(\eta)\} \tag{11.26}$$

这产生了映射到 η_{der} 的历史 I-状态的集合。划分可以直观地认为是“分辨率”，历史 I-空间就是在其上进行刻画的。如果式（11.26）中的集合很大，那么 I-空间能够被大量减小。目标是选择 κ 来使得划分中的集合尽可能大；但是必须小心以避免 I-空间收缩太多，否则问题就不能够解决。

例 11.11（状态估计） 在本例中，I-映射是大量应用中很方便采用的经典方法。假设已经有一种技术，它使用历史 I-状态 $\eta \in \mathcal{I}_{hist}$ 来对目前状态进行估计。在这种情况下，I-映射为 $\kappa_{est} : \mathcal{I}_{hist} \to X$。这种情况下衍生的 I-空间正好是 X。也就是说指定规划为 $\pi : X \to U$，它就是一个状态反馈规划。

考虑由 κ_{est} 引起的 $\mathcal{I}_{hist}$ 的划分。对于每个 $x \in X$，像式（11.26）所定义的集合 $\kappa_{der}^{-1}(x)$ 是所有导致相同状态估计的历史集合。X 上的规划不再能够区分导致相同状态估计的各种历史。也就是说失去了对状态估计中不确定性的编码能力。例如最好使行动依赖于 x 的估计的协方差，但却是不可能的，因为决策仅基于估计本身。 ■

例 11.12（阶段索引） 考虑 I-映射 κ_{stage}，它只返回目前阶段的索引。因此 $\kappa_{stage}(\eta_k) = k$。衍生的 I-空间是阶段的集合，它是 $\mathbb{N}$。衍生 I-空间上的一个反馈规划指定为 $\pi : \mathbb{N} \to U$。这等价于指定一个规划作为行动序列 $(u_1, u_2, \cdots)$，如第 2.3.2 节所述。因为反馈微不足道，这几乎就是无反馈的原始情况，所以也称之为开环规划。 ■

构造衍生信息转移方程　就像以前所提及的，需要全部历史 I-状态来确定衍生 I-状态。但是更可取的是放弃历史而全部在衍生 I-状态中进行工作。机器或者机器人上不存储历史，那么需要研究衍生信息转移方程。在这种情况下，重要要求如下：

如果 η_k 被 $\kappa(\eta_k)$ 代替，那么 $\kappa(\eta_{k+1})$ 必须仅用 $\kappa(\eta_k)$、u_k 和 y_{k+1} 来确定。

是否满足该要求依赖于特定的 I-映射。另外一种表达这种要求的方法是如果给定 $\kappa(\eta_k)$，那么全部历史 η 不包含能够进一步约束 $\kappa(\eta_{k+1})$ 的任何信息。κ 所提供的信息足够确定下一个衍生 I-状态。这相似于充分统计量（sufficient statistic）的概念，它出现在决策论中[91]。如果满足要求，那么 κ 称为充分 I-映射。特点是充分性是相对于 $\mathcal{I}_{der}$ 的，而不是某种意义上绝对的。例如任意映射到 $\mathcal{I}_{der} = \{0\}$ 的 I-映射是充分的，因为 $\kappa(\eta_{k+1})$ 总是已知的（它在 0 处保持固定）。因此充分性的要求在很大程度上依赖于特定衍生 I-空间。

对于一个充分 I-映射，衍生信息转移方程为：

$$\kappa(\eta_{k+1}) = f_{\mathcal{I}_{der}}(\kappa(\eta_k), u_k, y_{k+1}) \tag{11.27}$$

其含义为 $\mathcal{I}_{der}$ 是一个新的 I-空间，问题就“生活”在该空间中。对决策者来说没有理由还考虑历史。该思路对许多规划算法的成功来说是非常关键的。第 11.2.2 节和第 11.2.3 节将介绍非确定的 I-空间和概率 I-空间，它们是两个最重要的衍生 I-空间，它们是从充分 I-映射而得到的。例 11.12 中 I-映射 κ_{stage} 也是充分的。例 11.11 的估计 I-映射通常不是充分的，因为还需要某些历史来提供更好的估计。

图 11.4（a）指出得到充分 I-映射的问题。图的顶部显示在引入 I-映射之前历史 I-状态的转移。图的底部显示试图衍生的信息转移方程 $f_{\mathcal{I}_{der}}$。要求就是在右下方得到的衍生 I-状态必须是相同的，不管从左上方沿着哪条路径。$f_{\mathcal{I}}$ 可以应用于 η，然后是应用 κ；或者 κ

应用于 η，然后是应用某 $f_{\mathcal{I}_{der}}$。即使存在 $f_{\mathcal{I}_{der}}$，问题是 κ 通常不是可颠倒的。某衍生 I-状态 $\eta_{der} \in \mathcal{I}_{der}$ 的原像 $\kappa^{-1}(\eta_{der})$ 产生了 $\mathcal{I}_{hist}$ 中历史的一个集合。将 $f_{\mathcal{I}}$ 应用到所有这些就会产生下一个阶段可能的历史 I-状态的集合。因为 κ^{-1} 所带来的模糊性，所以应用 κ 会产生衍生 I-状态的一个集合。映射的链如图 11.4（b）所示。如果在所有环境下只得到一个单元素集合，那么这就产生了所要求的 $f_{\mathcal{I}_{der}}$ 值。否则，关于目前的衍生 I-状态会出现新的不确定性。这可以通过在信息空间上定义一个信息空间来处理，但是这里没有考虑这种情况。

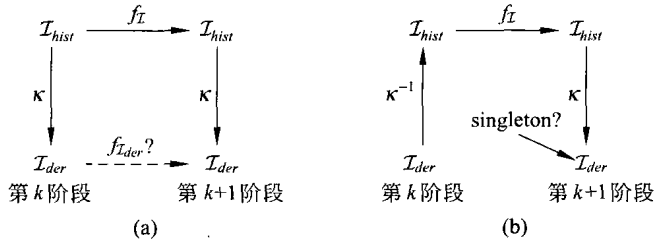


图 11.4 （a）为了使一个 I-映射成为充分的，必须到达右下方的相同结果，不管从左上方采用什么路径；（b）问题是 κ 个像可能包含许多历史，这些历史最终都将映射到多个衍生 I-状态

因为可以定义一个从任意衍生 I-空间到另外一个的 I-映射，本节提出的概念不必要求 $\mathcal{I}_{hist}$ 是开始点。例如映射 $\kappa: \mathcal{I}_{der} \rightarrow \mathcal{I}'_{der}$ 可称为关于 $\mathcal{I}_{der}$ 是充分的，而不是关于 $\mathcal{I}_{hist}$。

11.2.2 非确定的信息空间

本节通过图 11.3 定义 I-映射 κ_{ndet}，它将每个历史 I-状态转换到 X 的一个子集，该子集对应于所有可能的目前状态。大自然建模为非确定的，即除了只知道大自然将从 Θ 和 Ψ 选择行动以外，没有关于大自然将采取什么行动的任何其他信息。假设采用第 11.1.1 节的状态-行动传感器映射。考虑从一个历史 I-状态 $\eta_k = (\eta_0, \tilde{u}_{k-1}, \tilde{y}_k)$ 可以推导出什么。因为模型不涉及概率，令 η_0 表示集合 $X_1 \subseteq X$。令 $\kappa_{ndet}(\eta_k)$ 为 X 的最小子集，在给定 η_k 的情况下已知 x_k 落入该子集。该子集称为非确定的 I-状态。为提醒你 $\kappa_{ndet}(\eta_k)$ 是 X 的一个子集，它用 $X_k(\eta_k)$ 表示。重要的是 $X_k(\eta_k)$ 在尽可能小的同时与 η_k 保持一致。

回忆式（11.6），对于每个观测 y_k，可以推断出 x_k 的可能值的集合 $H(y_k) \subseteq X$。这可以作为对非确定的 I-状态的粗略估计。已知 $X_k(\eta_k) \subseteq H(y_k)$，否则 x_k 将与目前传感器观测不一致。如果我们从初始条件向前发展，同时应用状态转移方程进行约束，那么将会得到 $H(y_k)$ 的适当子集。

从状态转移函数 f，定义一个集值函数 F，它为每个 $x \in X$ 和 $u \in U(x)$ 产生 X 的一个子集为：

$$F(x,u) = \{x' \in X \,|\, \exists \theta \in \Theta(x,u), x' = f(x,u,\theta)\} \tag{11.28}$$

注意到 F 和 H 都是没有直接出现大自然行动的集值函数。大自然的影响在这些函数应用时所得到的集合中进行考虑。这对于计算非确定的 I-状态非常方便。

现在描述一个产生计算非确定的 I-状态 $X_k(\eta_k)$ 的归纳过程。$k=1$ 时：

$$X_1(\eta_1) = X_1(\eta_0, y_1) = X_1 \cap H(y_1) \tag{11.29}$$

方程首先用 (η_0, y_1) 代替 η_1。历史中没有任何行动。然后应用交集来使得两部分信息一致：（1）初始状态落入初始条件 X_1；（2）在给定观测 y_1 时，$H(y_1)$ 中可能有多个状态。

根据归纳法，现在假设已经计算 $X_k(\eta_k) \subseteq X$，接下来计算 $X_{k+1}(\eta_{k+1})$。从式（11.15），有 $\eta_{k+1} = (\eta_k, u_k, y_{k+1})$。因此新的信息只有 u_k 和观测 y_{k+1}。它们被认为是同一时刻的量。

考虑计算 $X_{k+1}(\eta_k, u_k)$。如果已知 x_k，那么应用 u_k 以后，状态可以落入用式（11.28）所得到的 $F(x_k, u_k)$ 内的任何位置。尽管实际上 x_k 未知，但是至少知道 $x_k \in X_k(\eta_k)$。因此：

$$X_{k+1}(\eta_k, u_k) = \bigcup_{x_k \in X_k(\eta_k)} F(x_k, u_k) \tag{11.30}$$

这可以认为是从 $X_k(\eta_k)$ 中某状态开始、应用任何行动 $u_k \in U(x)$ 和 $\theta_k \in \Theta(x_k, u_k)$ 所能够到达的所有状态的集合（见图 11.5）。

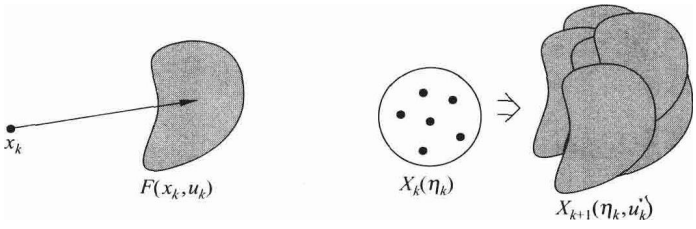


图 11.5　计算非确定的 *I*-状态的第一步是在所有可能的 $x_k \in X_k(\eta_k)$ 上取 $F(x_k, u_k)$ 的并集

下一步是考虑观测 y_{k+1}。该信息指示 x_{k+1} 落在 $H(y_{k+1})$ 中。因此用交集来得到非确定的 *I*-状态：

$$X_{k+1}(\eta_{k+1}) = X_{k+1}(\eta_k, u_k, y_{k+1}) = X_{k+1}(\eta_k, u_k) \cap H(y_{k+1}) \tag{11.31}$$

这实现了从 $X_k(\eta_k)$ 计算 $X_{k+1}(\eta_{k+1})$ 的过程。从式（11.29）开始，式（11.30）和式（11.31）经过所需要次数的迭代，可以计算任何阶段的非确定的 *I*-状态。

因为非确定的 *I*-状态总是 X 的子集，那么可以得到非确定的 *I*-空间 $\mathcal{I}_{ndet} = \mathrm{pow}(X)$（见图 11.3）。如果 X 是有限的，那么 $\mathcal{I}_{ndet}$ 也是有限的，$\mathcal{I}_{hist}$ 却不是如此，因为历史随阶段数连续增加。因此如果阶段数是无界的或者比 X 更大，那么采用非确定的 *I*-状态是更可取的。κ_{ndet} 是一个充分 *I*-映射。这意味着规划问题完全可以用 $\mathcal{I}_{ndet}$ 来表达，而不需要保留历史。目标区域 X_G 直接表示为一个非确定的 *I*-状态。那么规划任务就是要到达一个非确定的 *I*-状态 $X_k(\eta_k)$，其中 $X_k(\eta_k) \subseteq X_G$。

式（11.30）和式（11.31）说明 $X_{k+1}(\eta_{k+1})$ 可以从 $X_k(\eta_k)$、u_k 和 y_{k+1} 计算而来，所以 κ_{ndet} 是充分的。这意味着可以形成衍生信息转移方程。非确定的 *I*-空间也可以当作“另外一个状态空间”来对待。尽管许多历史 *I*-状态可能映射到相同的非确定的 *I*-状态，但是对于决策来说一旦给定 $X_k(\eta_k)$，那么特定历史是无关紧要的。

例 11.13 和例 11.14 很简单，便于阐述概念。

例 11.13（三状态例子） 对于所有 $x \in X$ 和 $u \in U$，令 $X = \{0,1,2\}$，$U = \{-1,0,1\}$，$\Theta(x,u) = \{0,1\}$。给定状态转移为 $f(x,u,\theta) = x + u + \theta$ 模 3。关于感测，对于所有 $x \in X$，有 $Y = \{0,1,2,3,4\}$ 和 $\Psi(x) = \{0,1,2\}$。传感器映射是 $y = h(x,\psi) = x + \psi$。

该例子只涉及 3 个状态，其非确定的 *I*-空间为：

$$I_{ndet} = \{\phi, \{0\}, \{1\}, \{2\}, \{0,1\}, \{1,2\}, \{0,2\}, \{0,1,2\}\} \tag{11.32}$$

它是 $X=\{0,1,2\}$ 的幂集。空集 ϕ 通常可以从 $\mathcal{I}_{ndet}$ 中删除[1]。假设初始条件为 $X_1=\{0,2\}$，并假设初始状态为 $x_1=0$。初始状态对决策者来说是未知的，但是在例中需要确保进行有效观测。

现在考虑在多个阶段上的执行过程。假设第一次观测为 $y_1=2$。基于传感器映射，那么有 $H(y_1)=H(2)=\{0,1,2\}$，这并不很有用，因为 $H(2)=X$。应用式（11.29）产生 $X_1(\eta_1)=\{0,2\}$。现在假设决策者应用行动 $u_1=1$，并且大自然应用 $\theta_1=1$。使用 f，产生 $x_2=2$。决策者不知道 θ_1，所以它必须考虑大自然可能采取的任何行动。它使用式（11.30）推断到：

$$X_2(\eta_1,u_1)=F(2,1)\cup F(0,1)=\{0,1\}\cup\{1,2\}=\{0,1,2\} \tag{11.33}$$

现在假设 $y_2=3$。从传感器映射，那么 $H(3)=\{1,2\}$。应用式（11.31）产生：

$$X_2(\eta_2)=X_2(\eta_1,u_1)\cap H(y_2)=\{0,1,2\}\cap\{1,2\}=\{1,2\} \tag{11.34}$$

过程重复多次。通过访问非确定的 I-状态的一个序列就可以产生 $\mathcal{I}_{ndet}$ 中的一条路径。如果得到观测 $y_k=4$，那么立刻就知道状态 x_k 了，因为 $H(4)=\{2\}$。 ■

11.2.3 概率信息空间

从图 11.3 定义 I-映射 κ_{prob}，它将每个历史 I-状态转换成 X 上的概率分布。一个马尔可夫概率模型建立在大自然行动只依赖于目前的状态和行动这种意义上，而不是状态和行动历史。式（11.30）和式（11.31）的并集和交集在本节分别被边缘化和贝叶斯公式所代替。从某种意义上，这些与并集和交集是概率等价的。可以将本节的表达与第 11.2.2 节的表达进行比较。

采用标准概率符号 $P(x|\eta)$，而不是 $\kappa_{prob}(\eta)$。本节 $P(x_{k+1}|\cdot)$ 形式的表达式对应于第 11.2.2 节中的 $X_k(\cdot)$。

第一步是构造 H 和 F 的概率版本。它们分别是 $P(x_k|y_k)$ 和 $P(x_{k+1}|x_k,u_k)$。其中 $P(x_{k+1}|x_k,u_k)$ 在第 10.1.1 节中给出过。从第 11.1.1 节可以知道 $P(y_k|x_k)$ 容易从 $P(\psi_k|x_k)$ 产生，那么为了得到 $P(x_k|y_k)$，应用贝叶斯公式：

$$P(x_k|y_k)=\frac{P(y_k|x_k)P(x_k)}{P(y_k)}=\frac{P(y_k|x_k)P(x_k)}{\sum_{x_k\in X}P(y_k|x_k)P(x_k)} \tag{11.35}$$

在最后一步，$P(y_k)$ 用式（9.8）的边缘化来表示。在这种情况下 x_k 以和的形式出现；因此分母只是 y_k 的一个函数。贝叶斯公式要求知道先验信息 $P(x_k)$。在接下来的表达中，这将用概率 I-状态来代替。

现在定义概率 I-状态。每个概率 I-状态都是 X 上的一个概率分布，记为 $P(x_k|\eta_k)$。初始条件产生 $P(x_1)$。就像非确定的情况，概率 I-状态可以用归纳法来计算。对于 $k=1$，新的信息只有 y_1。因此概率 I-状态 $P(x_1|\eta_1)$ 为 $P(x_1|y_1)$。令式（11.35）中的 $k=1$，那么产生：

$$P(x_1|\eta_1)=P(x_1|y_1)=\frac{P(y_1|x_1)P(x_1)}{\sum_{x_1\in X}P(y_1|x_1)P(x_1)} \tag{11.36}$$

---

1 一个著名的例外是在非确定的有限自动机理论中，其中有可能机器的所有副本都停止了，并且没有目前的状态[892]。

假设给定 $P(x_k|\eta_k)$，现在进行归纳。任务是确定 $P(x_{k+1}|\eta_{k+1})$，它等价于 $P(x_{k+1}|\eta_k,u_k,y_{k+1})$。就像第 11.2.2 节中介绍的，这将分两部分进行，首先考虑 u_k 的影响，然后考虑 y_{k+1} 的影响。第一步是从 $P(x_k|\eta_k)$ 确定 $P(x_{k+1}|\eta_k,u_k)$。首先注意到：

$$P(x_{k+1}|\eta_k,x_k,u_k)=P(x_{k+1}|x_k,u_k) \tag{11.37}$$

因为一旦给定 x_k，那么 η_k 将不包含对 x_{k+1} 的预测信息。可以用式（9.8）的边缘化来消除 $P(x_{k+1}|x_k,u_k)$ 中的 x_k。而且必须消除它，因为它没有给定。将这些步骤综合在一起产生了：

$$P(x_{k+1}|\eta_k,u_k)=\sum_{x_k\in X}P(x_{k+1}|x_k,u_k,\eta_k)P(x_k|\eta_k)=\sum_{x_k\in X}P(x_{k+1}|x_k,u_k)P(x_k|\eta_k) \tag{11.38}$$

它用给定的量来表达 $P(x_{k+1}|\eta_k,u_k)$。方程（11.38）可以认为是式（11.30）的概率版本。

接下来考虑观测 y_{k+1}。这由式（11.35）来完成，用的是到目前所积累的信息 η_k 和 u_k，并且用 $k+1$ 代替 k。结果为：

$$P(x_{k+1}|y_{k+1},\eta_k,u_k)=\frac{P(y_{k+1}|x_{k+1},\eta_k,u_k)P(x_{k+1}|\eta_k,u_k)}{\sum_{x_{k+1}\in X}P(y_{k+1}|x_{k+1},\eta_k,u_k)P(x_{k+1}|\eta_k,u_k)} \tag{11.39}$$

这可以被认为是式（11.31）的概率版本。式（11.39）的左边等价于 $P(x_{k+1}|\eta_{k+1})$，它是第 $k+1$ 阶段的概率 I-状态。在右边有两种不同类型的项。$P(x_{k+1}|\eta_k,u_k)$ 由式（11.38）给出。因此只剩下 $P(y_{k+1}|x_{k+1},\eta_k,u_k)$ 需要计算。注意到：

$$P(y_{k+1}|x_{k+1},\eta_k,u_k)=P(y_{k+1}|x_{k+1}) \tag{11.40}$$

因为传感器映射仅依赖于状态（和大自然感测行动的概率模型，该模型也只依赖于状态）。因为 $P(y_{k+1}|x_{k+1})$ 作为传感器模型的一部分来指定，那么我们现在已经确定如何从 $P(x_k|\eta_k)$、u_k 和 y_{k+1} 来得到 $P(x_{k+1}|\eta_{k+1})$。因此 $\mathcal{I}_{prob}$ 是可以当作一个状态空间来对待的另外一个 I-空间。

概率 I-空间 $\mathcal{I}_{prob}$（见图 11.3）是 X 上所有概率分布的集合。表达式（11.38）和表达式（11.39）使得 I-映射是充分的，这意味着规划问题可以完全用 $\mathcal{I}_{prob}$ 来表达，而不需要保留历史。一个目标区域可以指定为概率上的约束。例如从某特定 $x\in X$，目标可能是到达使得 $P(x_k|\eta_k)>1/2$ 的任何概率 I-状态。

例 11.14（再次用三状态的例子） 现在回到例 11.13，但是这次用的是概率模型。对于一个概率 I-状态，令 p_i 表示目前状态是 $i\in X$ 的概率。任何概率 I-状态可以表示为 $(p_0,p_1,p_2)\in\mathbb{R}^3$。这也就是说 I-空间可以很好地嵌入 $\mathbb{R}^3$ 中。根据概率性质，那么 $p_0+p_1+p_2=1$，这可以阐述为 $\mathbb{R}^3$ 中的一个平面方程，它将 $\mathcal{I}_{prob}$ 限制在一个二维集合中。对每个 $i\in\{0,1,2\}$，有 $0\leqslant p_i\leqslant 1$。这意味着 $\mathcal{I}_{prob}$ 被限制在 $\mathbb{R}^3$ 中的一个三角区域。该区域的顶点为 (0,0,1)、(0,1,0) 和 (1,0,0)，它们对应于 3 种具有完美状态信息的方法。从某种意义上，与这些点之间的距离对应于状态中的不确定性。均匀概率分布 (1/3,1/3,1/3) 与这 3 个顶点等距。三角区域在 $\mathbb{R}^2$ 上的投影如图 11.6 所示。这种情况下 p_0 和 p_1 指定了

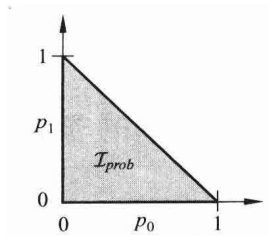


图 11.6　三状态例子的概率 I-空间是嵌入 $\mathbb{R}^3$ 的一个 2-复形，这个复形可以投影到 $\mathbb{R}^2$ 来产生 $\mathbb{R}^2$ 中三角区域

$\mathbb{R}^2$ 上一个点，p_2 从 $p_2 = 1 - p_0 - p_1$ 得到。

即使对于一个固定的初始条件来说历史 I-空间是可数无限的，$\mathbb{R}^3$ 中三角区域也是一个不可数无限集合。这似乎很奇怪，但是没有错误，因为对于一个固定初始条件来说，一般不可能到达 $\mathcal{I}_{prob}$ 中的所有点。如果初始条件可以是 $\mathcal{I}_{prob}$ 中的任意点，那么所有概率 I-空间被覆盖，因为 $\mathcal{I}_0 = \mathcal{I}_{prob}$，其中 $\mathcal{I}_0$ 为初始条件空间。 ■

11.2.4 有限记忆信息空间

限制记忆的数量提供了一种减小历史 I-空间的方法。除了特殊情况以外，这通常不保持原始问题的可用性或者最优性。尽管如此，当没有其他方法减小 I-空间时，这种 I-映射在实际中非常有用，并且它们偶尔也能保持所希望的可用性，有时甚至是最优性。

前 i 个阶段 在这种模型下，历史 I-空间被裁剪。从记忆中删除 i 个阶段以前所得到的任何行动或观测。对于任意整数 $i > 0$ 和 $k > i$，一个 I-映射 κ_i 定义为：

$$\kappa_i(\eta_k) = (u_{k-i}, \cdots, u_{k-1}, y_{k-i-1}, \cdots, y_k) \tag{11.41}$$

如果 $i \leqslant k$，那么衍生 I-状态是全部历史 I-状态，见式（11.41）。如果该方法能够产生解，那么它的优越性在于 I-状态的长度不再随阶段而增长。如果 X 和 U 是有限的，那么衍生 I-空间也是有限的。注意到 κ_i 在第 11.2.1 节所定义的意义上是充分的，因为经过了足够多的历史阶段来确定衍生 I-状态。

传感器反馈 从历史 I-状态中删掉除最后一个传感器观测以外的所有信息就可以得到一个 I-映射 $\kappa_{sf} : \mathcal{I}_{hist} \to Y$，它定义为 $\kappa_{sf}(\eta_k) = y_k$，称为传感器反馈。在这种情况下，所有决策直接用目前传感器的观测来构造。衍生 I-空间是 Y，并且衍生 I-空间上的规划为 $\pi : Y \to U$，称为传感器反馈规划。在某些文献中也称其为纯反应规划。一些在历史 I-空间中有解的问题不能用传感器反馈来解决。忽视历史数据能够防止状态过于复杂。在某种意义上，传感器反馈有些眼光短浅，这不可避免会造成无限重复同样的错误。但在确定对于某特殊问题这种传感器反馈规划解是否存在时它是有价值的。这种规划在实际情况下实现起来更加简单，因为行动可以直接与传感器输出相连。当然，如果一个问题的传感器反馈规划解存在，并且只关心可行解，那么按照历史 I-空间或者某更大的衍生 I-空间来设计和实现一个规划是没有意义的。注意到这种 I-映射是充分的，即使它忽视了整个历史数据。

11.3 离散状态空间的实例

11.3.1 基本的非确定性的实例

首先我们考虑一个简单例子，它使用例 11.3 中的符号传感器。

例 11.15（使用符号传感器） 该例与例 10.1 相似，只不过它涉及的是传感器不确定性，而不是预测不确定性。令 $X = \mathbb{Z}$，$U = \{-1, 1, u_T\}$，$Y = \{-1, 0, 1\}$ 和 $y = h(x) = \operatorname{sgn} x$。对于状态转移方程 $x_{k+1} = f(x_k, u_k) = x_k + u_k$。没有大自然行动干扰状态转移方程或者传感器映射。因此将来的历史 I-状态是可以预测的。信息转移方程为 $\eta_{k+1} = f_{\mathcal{I}}(\eta_k, u_k)$。假设初始时 $\eta_0 = X$，

也就是说任何初始状态都是可能的。目标是到达 $0 \in X$。

第 k 阶段的历史 I-状态的一般表达式为：

$$\eta_k = (X, u_1, \cdots, u_{k-1}, y_1, \cdots, y_k) \tag{11.42}$$

一个可能的 I-状态是 $\eta_5 = (X, -1, 1, 1, -1, 1, 1, 1, 1, 0)$。使用第 11.2.2 节中的非确定的 I-空间 $\mathcal{I}_{ndet} = \mathrm{pow}(X)$，它是不可数无限的。但是通过仔细观察可以发现：大多数非确定的 I-状态都不是可达的。如果 $y_k = 0$，那么可以知道 $x_k = 0$；因此 $X_k(\eta_k) = \{0\}$。如果 $y_k = 1$，那么将总是 $X_k(\eta_k) = \{1, 2, \cdots\}$，除非观察到 0。如果 $y_k = -1$，那么 $X_k(\eta_k) = \{\cdots, -2, -1\}$。由此可以在上面提及的三个非确定的 I-状态上指定一个规划 π。对于第一个，有 $\pi(X_k(\eta_k)) = u_T$。对于另外两个，有 $\pi(X_k(\eta_k)) = -1$ 和 $\pi(X_k(\eta_k)) = 1$。基于符号，规划试图向 0 移动。如果有不同的初始条件，那么可以到达更多的非确定的 I-状态，但是当定义问题时并不要求这些。注意到最优长度解由规划产生。

解甚至可以用传感器反馈来实现，因为行动仅依赖于目前传感器的值。令 $\pi : Y \to U$ 定义为：

$$\pi(y) = \begin{cases} -1 & (y=1) \\ 1 & (y=-1) \\ u_T & (y=0) \end{cases} \tag{11.43}$$

这在 $\mathcal{I}_{hist}$ 上定义规划时大量节省存储空间。 ■

接下来的例子简单阐述了在不知道目前状态的情况下解决问题的方法。这产生了目标可辨识性（goal recognizability）的问题[662]（见第 12.5.1 节）。

例 11.16（目标可辨识性） 令 $X = \mathbb{Z}$，$U = \{-1, 1, u_T\}$ 和 $Y = \mathbb{Z}$。对于状态转移方程 $x_{k+1} = f(x_k, u_k) = x_k + u_k$。现在假设采用例 11.7 的变化形式来对感测进行建模：$y = h(x, \psi) = x + \psi$ 和 $\Psi = \{-5, -4, \cdots, 5\}$。仍然假设 $\eta_0 = X$。在这种情况下，不可能保证到达目标 $X_G = \{0\}$，因为存在目标可辨识性问题。传感器映射中的扰动使得没有足够精确的状态测量来推断精确的状态。如果目标区域 X_G 扩大到 $\{-5, -4, \cdots, 5\}$，那么问题可以得到解决。由于存在扰动，非确定的 I-状态总是 11 个状态的一个连续序列的子集。得到一个规划来移动该区间直到非确定的 I-状态变成 X_G 的一个子集，这很简单。当这发生时，规划应用 u_T。在解决该问题时，从来不需要知道精确的状态。 ■

图 11.7 中显示的问题有两个目的。首先它是一个无传感器的规划[323,396]，也就是说没有观测（见第 11.5.4 节和第 12.5.2 节）。即它没有得到关于状态的信息。与直觉相反，对于该例以及其他一些可以设计规划的情况，可以进行估计状态。第二个目的是阐述 I-空间用第 11.2.1 节的 I-映射是如何收缩的。该例的标准非确定的 I-空间包括 2^{19} 个 I-状态，但是它可以映射到一个更小的只包含一些元素的衍生 I-空间。

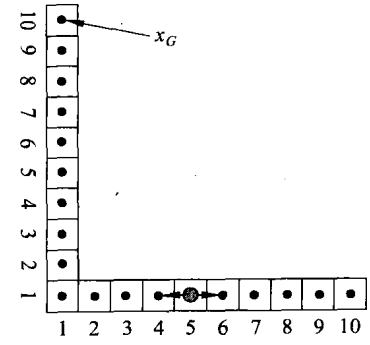


图 11.7　涉及 19 个状态的例子；这里没有传感器观测，但是选择那些使得状态能够被估计的行动；例子阐述了通过 I-映射可减小 I-空间

例 11.17（在 L 形走廊中移动） 图 11.7 所示的状态空间 X 中有 19 个状态，每个状态对应于白色格片上的一个位置。为方便起见，令每个状态用 (i, j) 表示。

有 10 个底部状态 $(1,1),(2,1),\cdots,(10,1)$，和 10 个左边状态 $(1,1),(1,2),\cdots,(1,10)$。因为 $(1,1)$ 既是底部状态又是左边状态，它称为角状态。

该问题没有传感器观测。但是大自然对状态转移有影响，这导致一种非确定的不确定性。如果应用的行动试图只前进一步，大自然可能引起进行两步或者三步行动。这可以这样来进行建模。令：

$$U=\{(1,0),(-1,0),(0,1),(0,-1)\} \tag{11.44}$$

并令 $\Theta=\{1,2,3\}$。当运动没有被（终端）阻挡时，状态转移方程定义为 $f(x,u,\theta)=x+\theta u$。例如，如果 $x=(5,1)$、$u=(-1,0)$ 和 $\theta=2$，那么下一个状态为 $(5,1)+2(-1,0)=(3,1)$。如果可能有阻挡，状态只可能变化到被阻挡为止。由于阻挡，甚至可能 $f(x,u,\theta)=x$。

因为没有传感器观测，第 k 阶段的历史 I-状态为：

$$\eta_k=(\eta_0,u_1,\cdots,u_{k-1}) \tag{11.45}$$

现在使用非确定的 I-空间 $\mathcal{I}_{ndet}=\mathrm{pow}(X)$。给定初始状态 $x_1=(10,1)$，这意味着初始 I-状态 η_0 为 $\{(10,1)\}$。目标是到达 I-状态 $\{(1,10)\}$，那么任务是设计一个规划，它从右下运动到左上。

如果有完美信息，那么问题就很简单；但是如果没有传感器，那么不确定性可能增长得很快。例如从初始状态应用行动 $u_1=(-1,0)$，非确定的 I-状态变成 $\{(7,1),(8,1),(9,1)\}$。应用 $u_2=(-1,0)$ 以后变成 $\{(4,1),\cdots,(8,1)\}$。但是该问题的一个好的特性是不用感测也可以减小不确定性。假设对于 100 个阶段，我们重复应用 $u_k=(-1,0)$。那么所产生的 I-空间是什么？当接近角状态时，不确定性减小，因为大自然不会使状态进一步发生变化。每个行动 $u_k=(-1,0)$ 至少减小 X 轴坐标一次。因此 9 个或更多阶段以后，$\eta_k=\{(1,1)\}$。一旦知道了这些，那么就可以应用行动（0,1）。当状态移动穿过左边状态的集合时，不确定性将会增加。如果应用（0,1）9 次或者 9 次以上，那么可以确切知道 $x_k=(1,10)$，它是所要求的目标状态。

现在已经得到一个成功的规划：（1）应用 $(-1,0)$ 9 次；（2）然后应用（0,1）9 次。这种规划可以定义在 $\mathcal{I}_{ndet}$，但是更简单的是使用例 11.12 的 I-映射 κ_{stage} 来定义规划 $\pi:\mathbb{N}\to U$。对于 $1\leqslant k\leqslant 9$ 的 k，$\pi(k)=(-1,0)$。对于 $10\leqslant k\leqslant 18$ 的 k，$\pi(k)=(0,1)$。对于 $k>18$ 的 k，$\pi(k)=u_T$。即使初始条件为 X 的任意子集，规划也可如此进行。从该点向前，假设给定任意子集作为初始条件。

现在通过从 $\mathcal{I}_{ndet}$ 构造其他衍生 I-空间来考虑一些可选的规划。令 κ_3 为从 $\mathcal{I}_{ndet}$ 到 3 个衍生 I-状态的集合 $\mathcal{I}_3$ 的一个 I-映射。令 $\mathcal{I}_3=\{g,l,a\}$，其中 g 表示“目标”，l 表示“左”，a 表示“任意”。I-映射 κ_3 为：

$$X(\eta)=\begin{cases} g & X(\eta)=\{(1,10)\} \\ l & X(\eta)\text{剩余状态集的一个子集} \\ a & \text{其他情况} \end{cases} \tag{11.46}$$

基于所描述的成功规划，$\mathcal{I}_3$ 上的解定义为 $\pi(g)=u_T$，$\pi(l)=(0,1)$ 和 $\pi(a)=(-1,0)$。这种规划比 $\mathbb{N}$ 上的规划更容易表示，但是它有一个缺点，即 I-映射 κ_3 不是充分的。也就是说在规划执行中需要保留更多的非确定的 I-状态。否则没法知道什么时候会发生状态转移。例如，如果从 a 开始应用（−1,0），那么机器人如何确定下一个阶段到达 l 还是 a 呢？这可以从非确定的 I-状态来确定。

为了讨论该问题，考虑一个新的 I-映射 $\kappa_{19}:\mathcal{I}_{ndet}\to\mathcal{I}_{19}$，该映射是充分的。有 19 个衍生 I-状态，它们包括以前定义的 g、对于 $1\leqslant i\leqslant 9$ 的 l_i 和对于 $2\leqslant i\leqslant 10$ 的 a_i。I-映射定义为 $\kappa_{19}(X(\eta))=g$，如果 $X(\eta)=\{(1,10)\}$。否则对于最小的 i 值，有 $\kappa_{19}(X(\eta))=l_i$ 使得 $X(\eta)$ 为 $\{(1,i),\cdots,(1,10)\}$ 的一个子集。如果对于 i 不存在这种值，那么对于最小的 i 值，有 $\kappa_{19}(X(\eta))=a_i$ 使得 $X(\eta)$ 为 $\{(1,1),\cdots,(1,10),(2,1),\cdots,(i,1)\}$ 的一个子集。现在规划定义为 $\pi(g)=u_T$，$\pi(l_i)=(0,1)$ 和 $\pi(a_i)=(-1,0)$。尽管规划很大，但是在执行过程中机器人不需要表示全部非确定的 I-状态。正确的转移发生。例如，如果在 a_5 应用 $u_k=(-1,0)$，那么得到 a_4。如果在 a_2 应用 $u_k=(-1,0)$，那么得到 l_1。从那里应用 $u=(0,1)$ 就产生 l_2。这些行动可以重复进行直到最后到达 l_9 和 g。但是所产生的规划并没有对原始开环规划有所改进提高。 ■

11.3.2　非确定的有限自动机

本章的思路与有限自动机理论之间有一个有趣的联系，这是计算理论的一部分（见文献[465,892]）。在第 2.1 节曾经提到过：确定是否存在为 DFA 所接受的某串的过程等价于一个离散可行规划问题。如果模型中引入不可预测性，那么得到一个非确定的有限自动机（NFA），如图 11.8 所示。这代表了理论计算机科学中非决定论的一个最简单例子。这种非确定的模型对定义计算模型及其相关的复杂度类型来说是非常有力的工具。这些模型产生信息空间的有趣的例子。

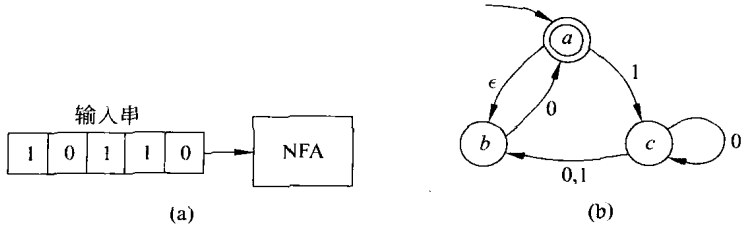


图 11.8　（a）一个非确定的有限自动机（NFA）是一个状态机，它读出一个输入字符串并决定是否接受它；（b）NFA 的一个图的描述

典型地，一个 NFA 用方向图来描述，如图 11.8（b）所示，并且被认为是一种特殊类型的有限状态机。图上的每个顶点表示一个状态，边表示可能的转移。机器读入有限长度的输入串。典型地，输入串是 0 和 1 的二进制序列。初始状态用向内的箭头指定，它没有源顶点，如图 11.8（b）中的指向状态 a。机器从该状态开始读入输入串的第一个符号。基于它的值，它进行相应的转移。对于 DFA 来说，必须为每个状态中两个输入 0 和 1 之一指定下一个状态。从 NFA 中的一个状态，可能有任意数量的往外去的边（包括 0），这些边对应于单个符号。例如从状态 c 读入 0，那么有两条往外的边（从 c 到 b 的箭头实际上对应于两条有向边，一条对应于 0，另外一条对应于 1）。也有用符号 ϵ 指定的边。如果状态有一个往外的边 ϵ，那么状态可能沿着边立刻进行转移，而不需要读入另外符号。对于可能遇到的任何往外的边 ϵ 来说，这可以迭代任意次，而不读入下一个输入符号。基于相同输入和 ϵ 转移有多条边这个原因，那么下一个状态有多个选择，这也就产生了非决定论的问题。没有传感器指示实际上选择的是哪个状态。

在计算理论中经常给出的解释是当有多个选择时，机器克隆它自己，一个副本运行一

个选择。好像有多个世界，其中有多个不同的大自然行动同时发生。如果对于状态和输入的一定组合没有往外的边，那么就不需要克隆。如图 11.8 所示的状态 a 这样的用双边界表示的状态被称为接受状态（accept state）。当输入串结束时，如果存在至少一个可选世界，其中的最后机器状态是一个接受状态，那么称 NFA 为接受输入串。

通常给定的 NFA 的表述似乎与离散可行规划的表述 2.1 非常接近。这里有一个典型的 NFA 表述[892]，它描述了图 11.8 所刻画的思路。

表述 11.2（非确定的有限自动机）

1. 一个有限状态空间 X。
2. 一个有限字母表 Σ，它表示可能的输入符号。令 $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$。
3. 一个转移函数 $\delta : X \times \Sigma_\epsilon \to \mathrm{pow}(X)$。对于每个状态和符号，往外的边的集合用指示所到达的状态来指定。
4. 开始状态 $x_0 \in X$。
5. 接受状态的集合 $A \subseteq X$。

例 11.18（三状态 NFA） 图 11.8 的例子可以用表述 11.2 来表示。它的组成有 $X = \{a,b,c\}$，$\Sigma = \{0,1\}$，$\Sigma_\epsilon = \{0,1,\epsilon\}$，$x_0 = a$ 和 $A = \{a\}$。状态转移方程要求为每个 $x \in X$ 和 Σ_ϵ 中的符号指定状态：

	0	1	ϵ
a	ϕ	$\{c\}$	$\{b\}$
b	$\{a\}$	ϕ	ϕ
c	$\{b,c\}$	$\{b\}$	ϕ

（11.47）

■

现在考虑将 NFA 以及串的接受重新表述为一种规划。一个输入串可以认为是使用无形反馈的一个规划；它是行动的一个固定序列。可行规划问题就是确定是否存在被 NFA 接受的串。因为没有反馈，所以也就没有感测模型。初始状态已知，但是不能测量后续状态。第 k 阶段的历史 I-状态 η_k 缩减为行动历史 $\eta_k = \tilde{u}_{k-1} = \{u_1, \cdots, u_{k-1}\}$。通过定义干扰状态转移的大自然行动就可以将非决定论考虑进来。这导致了如下表述。

表述 11.3（一个 NFA 规划问题）

1. 一个有限状态空间 X。
2. 一个行动空间 $U = \Sigma \cup \{\mathrm{u}u_T\}$。
3. 一个状态转移函数 $F : X \times U \to \mathrm{pow}(X)$。对于每个状态和符号，往外的边的集合用指示所到达的状态来指定。
4. 一个初始状态 $x_0 = x_1$。
5. 一个目标状态集合 $X_G = A$。

历史 I-空间 $\mathcal{I}_{hist}$ 用：

$$\mathcal{I}_k = \tilde{U}_{k-1} \tag{11.48}$$

来定义，其中 $k \in \mathbb{N}$，并取式（11.19）所定义的并集。假设 NFA 的初始状态是固定的；因此它不出现在 $\mathcal{I}_{hist}$ 的定义中。

为了表达规划任务，最好使用第 11.2.2 节的非确定的 I-空间 $\mathrm{I}_{ndet} = \mathrm{pow}(X)$。因此每个非确定的 I-状态 $X(\eta) \in \mathcal{I}_{ndet}$ 是 X 的子集，它对应于机器的可能目前状态。初始条件可以是 X 的任意子集，因为 ϵ 转移可以从 x_1 发生。后续的非确定的 I-状态直接从 F 产生。任务是计算如下形式的规划：

$$\pi = (u_1, u_2, \cdots, u_K, u_T) \tag{11.49}$$

这产生 $X_{K+1}(\eta_{K+1}) \in \mathcal{I}_{ndet}$，其中 $X_{K+1}(\eta_{K+1}) \cap X_G \neq \phi$。这意味着应用终止行动以后，NFA 必须至少有一个可能的状态处在 X_G 中。该条件弱于典型的规划要求。使用最坏情况分析，典型的要求是每个可能的 NFA 状态都处在 X_G 中。

由于有状态转移函数，表述 11.3 中给出的问题并不完全是表述 11.1 的特殊化。为方便起见，直接定义 F，而不是明确要求根据大自然行动 $\Theta(x,u)$ 来定义 f。这是关于这种表述的另外一个小问题。在本书考虑的规划问题中，总是假设有一个目前的状态。对于一个 NFA，对于一个输入如果没有往外的边，那么就不需要进行机器的克隆。也就是说不再存在潜在的目前状态。甚至可能所有克隆都停止，那么机器就没有目前状态。直接定义 F 就可以发生这种情况，但是规划问题必须总有一个目前的状态。为了解决该问题，我们可以扩展表述 11.3 中的 X，使之包括一个额外的不活动状态，它表示当没有往外的边时克隆停止。一个不活动状态可以不处在 X_G 中，一旦状态转移到不活动状态，那么状态将一直保持这种状态。在本节中，状态空间不用该方法进行扩展，但重要的是注意到很容易使 NFA 表述与表述 11.3 相一致。

规划模型可以比作是计算理论中 NFA 的标准使用。NFA 的语言（language）被定义为它所接受的所有输入串的集合。这里表述的规划问题确定是否存在 NFA 所接受的串（它结束于终止行动）。等价地，规划问题确定 NFA 的语言是否为空。构造所有成功规划的集合等价于确定 NFA 的语言。

例 11.19（三状态 NFA 的规划） 图 11.8 中的例子可以用表述 11.2 来表达。其组成有 $X = \{a,b,c\}$，$\Sigma = \{0,1\}$，$\Sigma_\epsilon = \{0,1,\epsilon\}$，$x_0 = a$ 和 $A = \{a\}$。函数 $F(x,u)$ 被定义为：

	0	1
a	ϕ	$\{c\}$
b	$\{a,b\}$	ϕ
c	$\{b,c\}$	$\{b\}$

（11.50）

非确定的 I-空间为：

$$X(\eta) = \{\phi, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}\} \tag{11.51}$$

其中初始条件为 $\eta_0 = \{a,b\}$，因为从 a 立即发生一个 ϵ 转移。该问题的一个规划解是 $(1,0,0,u_T,\cdots)$。这对应于给图 11.8 所刻画的 NFA 发送一个输入串“100”。在执行规划的过程中所得到的非确定 I-状态序列为：

$$\{a,b\} \xrightarrow{1} \{c\} \xrightarrow{0} \{b,c\} \xrightarrow{0} \{a,b,c\} \xrightarrow{u_T} \{a,b,c\} \tag{11.52}$$

■

有限自动机的一个基本定理论述到：对于 NFA 所接受的串的集合，存在一个 DFA（确定性的）接受相同的集合[892]。从非确定的 I-空间直接构造一个 DFA 就可以证明这一点。

每个非确定的 I-状态可以认为是一个 DFA 的一个状态。因此，如果原始 NFA 有 n 个状态，那么 DFA 有 2^n 个状态。DFA 的状态转移直接从非确定的 I-状态之间的转移中产生。当给定输入（或者行动），那么会发生从 X 的一个子集到另外一个子集的转移。在 DFA 中两个相应状态之间构造转移。这种构造阐述 I-空间是如何成为一个新的状态空间的，这个新的空间是当原始状态空间的状态未知时产生的。即使 I-空间通常比原始状态空间大，它的状态也总是已知的。因此它的行为与完美状态信息的情况一样。该思路非常具有一般性，它可以应用于 DFA 和 NFA 以外的一些其他问题，参见第 12.1.2 节。

11.3.3 概率情形：POMDPs

例 11.14 推广到 n 个状态的情况。在运算搜索和人工智能的文献中，这些通常被称为部分可观测马尔可夫决策过程或者 POMDPs（发音为 “pom dee peez”）。对于三状态情况，概率 I-空间 $\mathcal{I}_{prob}$ 是嵌入 $\mathbb{R}^3$ 的一个 2-单形。一般情况下，如果 $|X|=n$，那么 $\mathcal{I}_{prob}$ 是嵌入 $\mathbb{R}^n$ 的一个 $(n-1)$-单形。一个点的坐标表示为 $(p_0,p_1,\cdots,p_{n-1})\in\mathbb{R}^n$。由概率性质知：$p_0+\cdots+p_{n-1}=1$，这意味着 $\mathcal{I}_{prob}$ 是 $\mathbb{R}^n$ 的一个 $(n-1)$-维子空间。单形的顶点对应于 n 种情况，其中状态已知；因此它们的坐标为 $(0,0,\cdots,0,1),(0,0,\cdots,1,0),\cdots,(1,0,\cdots,0,0)$。为方便起见，通过指定 $\mathbb{R}^{n-1}$ 中的一个点使得 $p_1+\cdots+p_{n-2}\leq 1$，然后选择最后一个坐标 $p_{n-1}=1-p_0-\cdots-p_{n-2}$，这样单形就可以投影到 $\mathbb{R}^{n-1}$。第 12.1.3 节提出了 POMDPs 的规划算法。

11.4 连续状态空间

本节使用第 11.1 节和第 11.2 节给出的概念，并将它们推广到连续状态空间。这很重要，因为第 II 部分中运动规划所用的位形空间概念都基于连续状态空间。本节状态空间可能是第 4 章定义的一个位形空间 $X=C$，也可能是其他连续状态空间。因为它可能是一个位形空间，所以许多有趣的问题可以从机器人学中产生。

在介绍本节概念的过程中，可回顾离散状态空间中的相似概念。在一些情况下，表述可能是相同的。但是在其他情况下，连续情况更加复杂，但是它通常保持离散情况的某些特性。介绍第 11.5.1 节连续感测模型以后将会看到表述在连续空间中的某些问题比离散情况更加完美，也更容易理解。

11.4.1 离散阶段信息空间

这里假设有离散阶段。令 $X\subseteq\mathbb{R}^m$ 是一个 n 维流形，其中 $n\leq m$，它被称为状态空间[1]。令 $Y\subseteq\mathbb{R}^m$ 为一个 n_y 维流形，其中 $n_y\leq m$，它被称为观测空间。对于每个 $x\in X$，令 $\Psi(x)\subseteq\mathbb{R}^m$ 为一个 n_n 维流形，其中 $n_n\leq m$，它被称为大自然感测行动集合。第 11.4.1 节定

1 如果没有阅读第 4 章，并且不熟悉流形概念，那么可假设 $X=\mathbb{R}^n$，这没有什么区别。对 Y、$\Psi(x)$、U 和 $\Theta(x,u)$ 也可做相似的假设。

义的 3 种传感器映射 h 都是可能的，它们产生状态映射 $y=h(x)$、状态-大自然映射 $y=h(x,\psi)$ 或者基于历史的传感器映射 $y=h_k(x_1,\cdots,x_k,y)$。对于状态映射的情况，原像 $H(y)$ 导致了 X 的一个划分。也可以为状态-行动映射定义原像，但是它们不一定导致 X 的划分。

在连续状态空间中可以表述一些有趣的感测模型。第 11.5.1 节提供了一些传感器。如果使用大自然感测行动，那么可以选择非确定的不确定性或者概率不确定性。如果使用非确定的不确定性，那么表达与离散情况相同。概率模型用概率密度函数 $p:\Psi\rightarrow[0,\infty)$[1]来定义，其中 $p(\psi)$ 表示连续时间情况下的 $p(\psi)$。模型也可以表示为 $p(y|x)$，与离散状态空间中得到 $P(y|x)$ 的方式相同。

通常初始条件有 3 种选择：（1）给定 $x_1\in X$；（2）给定一个子集 $X_1\subseteq X$；（3）给定概率密度函数 $p(x)$。后两种情况的初始条件空间非常巨大。例如，如果 $X=[0,1]$ 和任意子集都可能作为初始条件，那么 $\mathcal{I}_0=\text{pow}(\mathbb{R})$，它比 $\mathbb{R}$ 有更高的势。如果任何概率密度函数都是可能的，那么 $\mathcal{I}_0$ 是一个函数空间[2]。

第 11.1.2 节中 I-空间的定义不变，只是需要理解到所有变量都是连续的。因此式（11.17）和式（11.19）可作为 $\mathcal{I}_k$ 和 $\mathcal{I}$ 的定义。令 $U\subseteq\mathbb{R}^m$ 为一个 n_u 维流形，其中 $n_u\leqslant m$。对于每个 $x\in X$ 和 $u\in U$，令 $\Theta(x,u)$ 为一个 n_θ 维流形，其中 $n_\theta\leqslant m$。将表述 11.1 中的每个离散变量用相应的连续变量代替，那么连续状态空间上的一个离散 I-空间规划问题很容易表述。因此这里没有给出其完整表述。

11.4.2　连续时间信息空间

现在假设存在有阶段的一个连续统。第 11.4.1 节的大多数组成部分保持不变。空间 X、Y、$\Psi(x)$、U 和 $\Theta(x,u)$ 保持不变。传感器映射也保持不变。主要的不同之处发生在状态转移方程，因为大自然的影响必须用速度来表示。这在第 10.6 节已经介绍过。在那里只有预测不确定性。在目前情况下，可能有预测不确定性和目前状态的感测不确定性。

对于离散阶段情况，历史 I-状态基于行动和观测序列。对于连续情况，历史变成时间的函数。如第 7.1.1 节所定义，令 T 表示时间区间，它可能是有界的，也可能是无界的。令 $\tilde{y}_t:[0,t]\rightarrow Y$ 被称为到 $t\in T$ 为止的观测历史。相似地，令 $\tilde{u}_t:[0,t)\rightarrow U$ 和 $\tilde{x}_t:[0,t]\rightarrow X$ 分别被称为到 $t\in T$ 为止的行动历史和状态历史。

因此在连续时间情况下有 3 种传感器映射。

（1）状态传感器映射表示为 $y(t)=h(x(t))$，其中 $x(t)$ 和 $y(t)$ 分别为 $t\in T$ 时刻的状态和观测。

（2）状态-大自然映射表示为 $y(t)=h(x(t),\psi(t))$，这意味着大自然在每个 $t\in T$ 选择某 $\psi(t)\in\Psi(x(t))$。

1　假设所有连续空间都是测度空间，并且所有概率密度函数都是这些空间上的可测函数。

2　为了认识该空间的大小，它通常可以被认为是一个无限维向量空间（回顾例 8.5）。例如考虑用级数展开式来表示每个函数。为了在[0,1]上精确表示任意解析函数，可能需要实值系数的一个无限序列。每个序列可以被认为是一个无限长向量，并且所有这种序列的集合形成了一个无限维向量空间。关于函数空间和泛函分析的更多背景知识，参见文献[348,839]。

（3）基于历史的传感器映射，它依赖于到目前所得到的所有状态。因此它依赖于整个函数 $\tilde{x}_t$。如果大自然也干扰观测，那么这可以表示为 $y(t)=h(\tilde{x}_t,\psi(t))$。

如果 $\tilde{u}_t$ 和 $\tilde{y}_t$ 与初始条件 η_0 进行组合，那么可以得到 t 时刻的历史 I-状态：

$$\eta_t=(\eta_0,\tilde{u}_t,\tilde{y}_t) \tag{11.53}$$

t 时刻的历史 I-空间是所有可能的 η_t 的集合，用 $\mathcal{I}_t$ 表示。$\mathcal{I}_t$ 是一个函数空间，因为每个 $\eta_t\in\mathcal{I}_t$ 都是时间的函数。回顾到在离散阶段情况下，使用式（11.18）或者式（11.19），每个 $\mathcal{I}_k$ 都可以组合进入到单个历史 I-空间 $\mathcal{I}_{hist}$。连续时间情况为：

$$\mathcal{I}_{hist}=\bigcup_{t\in T}\mathcal{I}_t \tag{11.54}$$

它是一个非正则化的函数集合，因为它们有不同的定义域；这种非正则性也发生在离散阶段情况，其中 $\mathcal{I}_{hist}$ 由可变长度的序列组成。

可以给出表述 11.1 中的代价泛函的连续时间版本来评估规划。令 L 表示代价泛函，它可能应用于任何状态-行动历史 $(\tilde{x}_t,\tilde{u}_t)$ 来产生：

$$L(\tilde{x}_t,\tilde{u}_t)=\int_0^t l(x(t'),u(t'))\mathrm{d}t'+l_F(x(t)) \tag{11.55}$$

其中 $l(x(t'),u(t'))$ 为瞬时代价，$l_F(x(t))$ 表示终端代价。

11.4.3 衍生的信息空间

对于连续状态空间，构造衍生 I-空间的意图甚至比离散情况更加强烈，因为 I-空间很快就变得难以处理。

11.4.3.1 离散阶段非确定 I-空间与概率 I-空间

第 11.2 节中的 I-映射和衍生 I-空间的概念直接扩展到连续空间。在非确定的情况下，κ_{ndet} 将初始条件和历史转换到 X 的一个子集。在概率情况下，κ_{prob} 产生 X 上的一个概率密度函数。首先考虑离散阶段情况。

可以得到第 11.2.2 节定义的非确定的 I-空间，除了离散集合用它的连续形式代替以外。例如式（11.28）中定义的 $F(x,u)$ 现在是一个连续集合，就像 X 和 $\Theta(x,u)$ 一样。因为概率 I-状态是概率密度函数，那么第 11.2.3 节中的推导过程需要稍微进行修改。但是概念上没有大的不同。按照第 11.2.3 节中的推导过程，并考虑需要代替哪一部分。

将式（11.35）替换为：

$$p(x_k\,|\,y_k)=\frac{p(y_k\,|\,x_k)p(x_k)}{\int_X p(y_k\,|\,x_k)p(x_k)\mathrm{d}x_k} \tag{11.56}$$

其中一部分是基于从 $p(\psi_k\,|\,x_k)$ 产生 $p(y_k\,|\,x_k)$。令式（11.56）中 $k=1$ 就可以得到归纳法的第一步，用它代替式（11.36）。按照从式（11.37）～式（11.40）所给出的解释，但采用的是概率密度函数，得到如下的更新方程：

$$\begin{aligned}p(x_{k+1}\,|\,\eta_k,u_k)&=\int_X p(x_{k+1}\,|\,x_k,u_k,\eta_k)p(x_k\,|\,\eta_k)\mathrm{d}x_k\\&=\int_X p(x_{k+1}\,|\,x_k,u_k)p(x_k\,|\,\eta_k)\mathrm{d}x_k\end{aligned} \tag{11.57}$$

$$p(x_{k+1}|y_{k+1},\eta_k,u_k)=\frac{p(y_{k+1}|x_{k+1})p(x_{k+1}|\eta_k,u_k)}{\int_X p(y_{k+1}|x_{k+1})p(x_{k+1}|\eta_k,u_k)\mathrm{d}x_{k+1}} \tag{11.58}$$

11.4.3.2　对不确定 I-空间与概率 I-空间的近似

一些其他的衍生 I-空间直接扩展到连续空间，如第 11.2.4 节和例 11.11 和例 11.12 的有限记忆模型。在目前情形下，尽可能减小 I-空间是非常有用的，因为在很多实际应用中 I-空间越来越难以处理。回忆到 I-映射 $\kappa:\mathcal{I}_{hist}\to\mathcal{I}_{der}$ 将 $\mathcal{I}_{hist}$ 划分为集合，在这些集合上必须采用常数行动。主要关心的是将规划限制到 $\mathcal{I}_{der}$ 不能影响问题的求解。

考虑构造能够近似非确定的和概率的 I-状态的衍生 I-空间。这种近似是有意义的，因为 X 通常是连续情况下的度量空间。目标是在试图避免损失关键信息的同时大量简化 I-空间。近似的质量与所产生的衍生 I-空间的大小之间需要进行折中。对于非确定的 I-状态，表述的是保守近似，它是保证包含非确定的 I-状态的那些集合。对于概率情况，提出的是基于矩的近似，它基于概率和统计的一般技术来近似概率密度。为了减少复杂性，表示仅限于离散阶段模型。

保守近似　假设采用非确定的不确定性，并且对非确定的 I-状态进行近似。定义一个 I-映射 $\kappa_{app}:\mathcal{I}_{ndet}\to\mathcal{I}_{app}$，其中 $\mathcal{I}_{app}$ 是 X 的一个特殊子集族。例如 $\mathcal{I}_{app}$ 可以表示 X 的所有球形子集的集合。如果 $X=\mathbb{R}^2$，那么球变成圆面，只需要 3 个参数 (x,y,r) 来描述 $\mathcal{I}_{app}$（x,y 为圆心，r 为半径）。这意味着 $\mathcal{I}_{app}\subset\mathbb{R}^3$，这比 $\mathcal{I}_{ndet}$ 更加简单，因为 $\mathcal{I}_{ndet}$ 是 $\mathbb{R}^2$ 中区域的一个复杂集合。为了使得 $\mathcal{I}_{app}$ 更小，那么可以要求 x、y 和 r 为整数（或者以某特定的离散度进行采样，如第 5.2.3 节中的定义）。如果 $\mathcal{I}_{app}$ 是有界的，那么衍生 I-状态的数量将变成有限的。当然这样做是有损失的，因为近似 $\mathcal{I}_{ndet}$ 的效果并不好。

对于固定行动序列 $(u_1,u_2,\cdots)$，考虑非确定的 I-状态序列：

$$X_1(\eta_1)\xrightarrow{u_1,y_2}X_2(\eta_2)\xrightarrow{u_2,y_3}X_3(\eta_3)\xrightarrow{u_3,y_4}\cdots \tag{11.59}$$

如图 11.9 所示。I-映射 $\mathcal{I}_{app}$ 必须为每个非确定的 I-状态选择一个包围区。从一个历史 I-状态 η 开始，可以首先计算非确定的 I-状态 $X_k(\eta_k)$，然后应用 $\mathcal{I}_{app}$ 来产生一个包围区。对于任意 η_k，如果有一个有效的方法计算 $X_k(\eta_k)$，那么 $\mathcal{I}_{app}$ 上的规划比 $\mathcal{I}_{ndet}$ 或者 $\mathcal{I}_{hist}$ 上的规划更加简单。

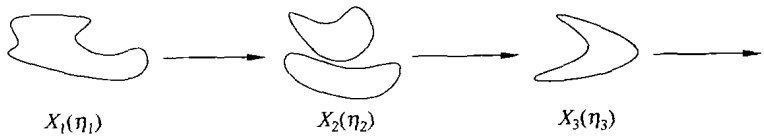


图 11.9　非确定的 I-状态可能是非常复杂的区域，它们很难或者不可能进行计算

如果计算 $X_k(\eta_k)$ 比较困难，一种解决方法是定义一个衍生信息转移方程，如第 11.2.1 节所述。但是麻烦在于 $\mathcal{I}_{app}$ 通常不是一个充分 I-映射。假设要计算 $\kappa_{app}(X_{k+1}(\eta_{k+1}))$，它是对 $X_{k+1}(\eta_{k+1})$ 的近似。这可以这样来进行：从 $X_k(\eta_k)$ 开始、应用式（11.30）和式（11.31），然后对 $X_{k+1}(\eta_{k+1})$ 应用 κ_{app}。一般情况下，这不同于从包围体 $\mathcal{I}_{app}(X_k(\eta_k))$ 开始、应用式（11.30）

和式（11.31），然后应用 κ_{app} 所产生的结果。

因此完全在 $\mathcal{I}_{app}$ 中表达这种转移不可能没有某些信息的损失。但是如果信息的损失是可以容忍的，那么这种近似方法是有用的。一般思路是为每次迭代中的非确定的 I-状态构造一个包围区。令 $\hat{X}_k$ 表示第 k 阶段的包围区。在使用这种近似时需要小心。如图 11.9 和图 11.10 所示，衍生 I-状态的序列产生分叉。简单地用 $\mathcal{I}_{app}$ 的元素来限制每个计算的 I-状态是不能得到图 11.10 中的序列的，整个序列是不同的。

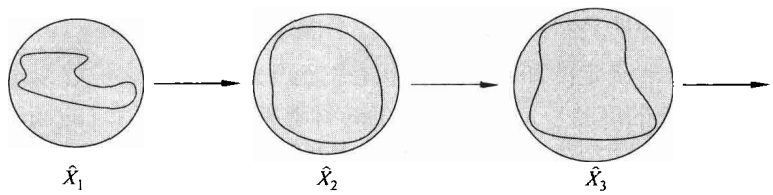


图 11.10 非确定的 I-状态可以用包围球来近似

初始情况下，选择 $\hat{X}_1$ 使得 $X_1(\eta_1) \subseteq \hat{X}_1$。在归纳中，认为 $\hat{X}_k$ 好像是真正的非确定的 I-状态（而不是一个近似）。使用式（11.30）和式（11.31），考虑 u_k 和 y_{k+1} 的更新是：

$$\hat{X}'_{k+1} = \left(\bigcup_{x_k \in \hat{X}_k} F(x_k, u_k) \right) \cap H(y_{k+1}) \tag{11.60}$$

一般情况下，$\hat{X}'_{k+1}(\eta_{k+1})$ 可能不在 $\mathcal{I}_{app}$ 中。因此必须在 $\hat{X}'_{k+1} \subseteq \hat{X}_{k+1}$ 约束下选择一个包围区 $\hat{X}_{k+1} \in \mathcal{I}_{app}$ 来近似 $\hat{X}'$。这完成了归纳，整个归纳过程产生了序列：

$$\hat{X}_1 \overset{u_1, y_2}{\rightarrow} \hat{X}_2 \overset{u_2, y_3}{\rightarrow} \hat{X}_3 \overset{u_3, y_4}{\rightarrow} \cdots \tag{11.61}$$

如图 11.10 所示。

现在可以在 $\mathcal{I}_{app}$ 上定义规划 $\pi : \mathcal{I}_{app} \rightarrow U$ 和信息转移。为了确保规划的可靠性，近似必须是保守的。如果在某迭代中，$\hat{X}_{k+1}(\eta_{k+1}) \subset \hat{X}'_{k+1}(\eta_{k+1})$，那么真正的状态可能不一定包括在近似衍生 I-状态中。例如这意味着机器人处在碰撞状态，即使衍生 I-状态指出这是不可能的。进行保守近似就可以避免这种情况。在一种极端情况下，总使 $\hat{X}_{k+1}(\eta_{k+1}) = X$ 就可以产生非常保守的近似。但是这是没有实际意义，因为在每个阶段可能的规划必须应用单个固定的行动。即使近似非常好，它也可能不能在近似 I-状态中引起转移。为了确保规划问题的解的存在，因此使得包围体尽可能接近衍生 I-状态。

第 5.3.2 节的碰撞检测中也存在对包围体的简化和使用它所付出的计算代价之间的折中。使用简单的形状可以使性能得到提高，但是在目前情形下有可能无法解决问题。使用目前的球所提供的界可能不够紧凑。使用椭球体能够提供更紧的近似，但是 $\mathcal{I}_{app}$ 的维数以 X 的维数的平方增长。球方程通常要求 $n+1$ 个参数，因此椭球体方程要求 $\binom{n}{2} + 2n$ 个参数。因此如果 X 的维数很高，那么就很难使用椭球体近似。非凸包围体可以提供更好的近似，但是参数的数量易变得无法处理，即使 $X = \mathbb{R}^2$。但是对于非常特殊的问题，有可能设计一个形状族，它们既便于处理，又紧致地近似了非确定的 I-状态。这产生了许多有趣的研究

课题。

基于矩的近似　因为概率 I-状态是函数，很自然使用函数逼近方法来近似 $\mathcal{I}_{prob}$。一种方法是使用泰勒级数展开的前 m 个系数。那么衍生的 I-空间变成可能的泰勒系数的空间。当 m 增加时近似的质量提高，但是衍生 I-空间的维数也增加了。

因为我们使用概率密度函数，所以通常可以使用矩来进行近似，而不是用泰勒级数的系数或者其他基本函数逼近方法。一阶矩和二阶矩就是我们所熟悉的均值和协方差。它们比其他近似更加可取，因为均值和协方差表示高斯密度，高斯密度是概率论中最基本的密度。因此用一阶矩和二阶矩来近似概率 I-空间等价于假设所产生的概率密度总是高斯的。由于这非常方便，所以在实际中经常使用。一般情况下，可以使用更高阶的矩来得到更高质量的近似，这当然需要更多的系数。令 $\kappa_{mom}:\mathcal{I}_{prob}\to\mathcal{I}_{mom}$ 表示基于矩的 I-映射。

对于 κ_{mom} 也有与 κ_{app} 相同的问题。在多数情况下，κ_{mom} 不是一个充分 I-映射。用于保守近似相同的方法来计算矩。为概率 I-状态而应用更新方程（11.57）和方程（11.58），但是每一步后，将 κ_{mom} 应用于所产生的概率密度函数。这限制住了 $\mathcal{I}_{mom}$ 中的衍生 I-状态。在已经应用式（11.57）和式（11.58）之一或者两者之后计算矩（可以得到不同的结果）。后一种情况根据不同的应用计算起来可能会更加困难。

首先考虑采用均值（一阶矩）来表示某概率 I-状态 $p(x|\eta)$。令 x_i 表示 x 的第 i 个坐标。关于 x_i 的均值 $\bar{x}_i$ 一般被定义为：

$$\bar{x}_i=\int_X x_i p(x|\eta)\mathrm{d}x \tag{11.62}$$

这产生了均值向量 $\bar{x}=(\bar{x}_1,\cdots,\bar{x}_n)$。假设我们只用均值来构造 $\mathcal{I}_{mom}$。因为没有关于密度的协方差的信息，计算 $\bar{x}$ 与估计状态非常相似。那么均值就作为估计，并且 $\mathcal{I}_{mom}=X$。这有助于简化 I-空间，但是无法推断与 I-状态有关的不确定性。概率质量主要集中在 $\bar{x}$，还是密度函数在 X 上非常分散？

使用二阶矩有助于解决该问题。关于两个变量 x_i 和 x_j 的协方差为：

$$\sigma_{i,j}=\int_X x_i x_j p(x|\eta)\mathrm{d}x \tag{11.63}$$

因为 $\sigma_{i,j}=\sigma_{j,i}$，那么二阶矩可以构成一个对称协方差矩阵：

$$\Sigma=\begin{pmatrix}\sigma_{1,1} & \sigma_{1,2} & \cdots & \sigma_{1,n}\\ \sigma_{2,1} & \sigma_{2,2} & \cdots & \sigma_{2,n}\\ \vdots & \vdots & & \vdots\\ \sigma_{n,1} & \sigma_{n,2} & \cdots & \sigma_{n,n}\end{pmatrix} \tag{11.64}$$

其中有 $\binom{n}{2}+n$ 个不同的元素，它们对应于每个 $x_{i,i}$ 和 x_i、x_j 对，对于不同的 i 和 j 使得 $1\leqslant i,j\leqslant n$。这意味着如果采用一阶矩、二阶矩，那么 $\mathcal{I}_{mom}$ 的维数为 $\binom{n}{2}+2n$。对于某些问题，可能所有的概率 I-状态确实是高斯分布的。在这种情况下，均值和协方差能够准确

反映概率 I-空间。这种情况下的 I-映射是充分的。这就产生了卡尔曼滤波，将在第 11.6.1 节介绍。

采用更高阶的矩可以得到更高质量的近似。r 阶矩被定义为：

$$\int_X x_{i_1} x_{i_2} \cdots x_{i_r} p(x|\eta) \mathrm{d}x \tag{11.65}$$

其中 $i_1, i_2, \cdots, i_r$ 是 r 个整数，它们用来代替 $\{1, \cdots, n\}$。

基于矩的近似方法与为非确定的不确定性而采用的保守近似方法非常相似。均值和协方差的使用与为非确定的情况所使用的椭球体非常相似。高斯密度的水平集是椭球体。这些水平集将置信区间的概念推广到置信椭球体，它描述了非确定的情形与概率情形之间非常密切的关系。高斯密度的定义域是 $\mathbb{R}^n$，它是无界的，这与非确定的情形相反。但是对于给定的置信水平，它可以近似为一个有界集合。例如可以计算一个椭球区域，使得 99.9% 的概率质量落在其中。一般情况下，有可能将矩的思路和包围体的思路组合起来为概率情形构造一个衍生 I-空间。这既考虑了概率，同时又能够保证规划的正确性。但这又增加了衍生 I-空间的维数。

11.4.3.3 连续时间的衍生 I-空间

连续时间的情况一般来说更加难以表达和计算。但是在一些特殊情况下则有一些很有效的计算方法。这些内容有些将在第 11.5 节和第 12 章进行介绍。为了完成 I-空间的框架，这里给出某些一般的表达。一般情况下，I-映射和衍生 I-空间可以用第 11.2.1 节中的思路来构造。

因为没有离散转移规则，所以衍生的 I-状态不能用简单的更新规则来表达。但是它们至少可以表示成一个函数，该函数指示从初始状态 $x(0)$ 应用 $\tilde{u}_t$ 和 $\tilde{\theta}_t$ 以后所得到的状态 $x(t)$。尽管这并未明确给出，但是这通常可以通过某种形式的积分来实现（见第 14.1 节）。一般情况下，令 $X_t(\eta_t) \subset X$ 表示 t 时刻的一个非确定的 I-状态，它代替了离散阶段情况中的 X_k。初始条件表示为 X_0，而不是离散阶段情况下所使用的 X_1。

还需要更多的定义来描述 $X_t(\eta_t)$。令 $\tilde{\theta}_t : [0,t) \to \Theta$ 表示到 t 时刻为止的大自然行动历史。相似地，令 $\tilde{\psi}_t : [0,t) \to \Psi$ 表示大自然感测行动的历史。假设初始条件为 $X_0 \subset X$。非确定的 I-状态定义为：

$$\begin{aligned} X_t(\eta_t) = \{ x \in X \,|\, & \exists x' \in X_0, \exists \tilde{\theta}_t, \quad \text{且 } \exists \tilde{\psi}_t \text{ 因而 } x = \Phi(x', \tilde{u}_t, \tilde{\theta}_t) \\ & \text{且} \forall t' \in [0,t], y(t') = h(x(t'), \psi(t')) \} \end{aligned} \tag{11.66}$$

总之，这意味着状态 $x(t)$ 处在 $X_t(\eta_t)$ 中，当且仅当存在一个初始条件 $x' \in X_0$、一个大自然历史 $\tilde{\theta}_t$、一个大自然感测行动历史 $\tilde{\psi}_t$ 使得转移方程到达 $x(t)$，并且观测历史 $\tilde{y}_t$ 在 0～t 的所有时间里都与传感器映射相符。

也可能产生一个概率 I-状态，但是这要求连续随机过程和随机微积分方程的内容。在某些情况下，所产生的表达非常合适；但是对衍生 I-状态进行一般表达是非常困难的，因为它需要对随机过程的行为进行一些假设。关于这种系统的细节，参见文献[570]。

11.5　连续状态空间的例子

11.5.1　传感器模型

在应用中会出现一种非常令人注目的感测模型，它涉及连续状态空间。本节列出的不同类型的感测模型主要来自于机器人学。将这些模型收集在一起有利于进行参考比较。某些还将在后续章节中使用，而其他一些则是为了加深对 I-空间的理解。根据是否包括大自然感测行动，每个传感器通常有两种版本的模型。

线性感测模型　主要从控制论中发展而来的线性感测模型是最平常的，也是最重要的。对于该家族中的所有传感器来说，假设 $X=Y=\mathbb{R}^n$（如果需要，非奇异线性变换可以使得传感器空间有更低的维数）。最简单的情况是恒等传感器，即 $y=x$。在这种情况下可以立刻知道状态。如果这种传感器在每个阶段都是可用的，那么 I-空间就通过 I-映射 $\kappa_{sf}:\mathcal{I}_{hist}\to X$ 收缩到 X。

现在可以用大自然感测行动来恶化这种完美的状态观测，得到 $y=h(x,\psi)=x+\psi$。假设 y 是对目前状态 x 的估计，误差限用一个常数 $r\in(0,\infty)$ 来指示。这可以通过为每个 $x\in X$ 分配一个半径为 r、中心在原点的一个封闭球来建模：

$$\Psi=\{\psi\in\mathbb{R}^n \mid \|\psi\|\leqslant r\} \tag{11.67}$$

图 11.11 阐述了所产生的非确定的感测模型，如果得到观测 y，那么就可以知道真实状态落在以 r 为半径、以 y 为中心的一个球的内部。这个球是原像 $H(y)$，如式（11.11）所定义的那样。为了构造概率模型，可以在 Ψ 上定义一个概率密度函数。例如可以假设 $p(\psi)$ 为均匀分布（尽管在一些应用中这个模型并不很现实，因为存在一个边界，在边界处概率质量突跳到零）。

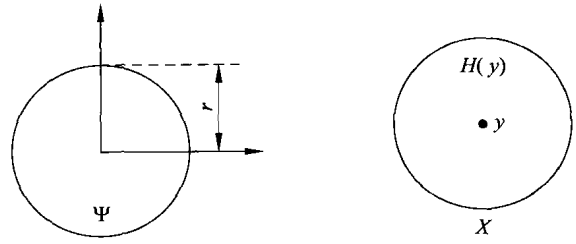


图 11.11　一个简单感测模型，它的观测误差不超过 r：（a）大自然感测行动空间；（b）观测量 y 在 X 中的原像

令 $\Psi(x)=\mathbb{R}^n$ 并在全部 $\mathbb{R}^n$ 上定义概率密度函数，这样就可以产生一个更加典型的概率感测模型（注意到这种传感器的非确定的版本是完全无用的）。最容易的一种选择是多变量高斯概率密度函数：

$$p(\psi)=\frac{1}{\sqrt{(2\pi)^n|\Sigma|}}e^{-\frac{1}{2}\psi'\Sigma\psi} \tag{11.68}$$

其中 Σ 是协方差矩阵（11.64），$|\Sigma|$ 是它的行列式，$\psi^T\Sigma\psi$ 是一个二次型，它是：

$$\psi^T\Sigma\psi=\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{i,j}\psi_i\psi_j \tag{11.69}$$

如果 $p(x)$ 是高斯的并且得到 y，那么 $p(y|x)$ 在这种模型下也一定是高斯的。这在第 11.6.1 节中变得更加重要。

目前所提出的感测模型通过线性变换可以进行一般化。例如令 C 表示一个非奇异 $n\times n$ 实值矩阵。如果传感器映射是 $y=h(x)=Cx$，那么状态可以立刻确定，因为映射 $y=Cx$ 是双射的；每个 $H(y)$ 包含 X 的一个唯一点。也可以在大自然感测行动上形成一个线性变换。令 W 表示一个 $n\times n$ 矩阵。传感器映射是：

$$y=h(x)=Cx+W\psi \tag{11.70}$$

一般情况下，C 和 W 可能是奇异的，但是仍然可以得到一个线性感测模型。假设 W=0。如果 C 是奇异的，那么就不可能从单个传感器观测中直接推断状态。一般情况下这对应于从一个 n 维状态空间到 Y 的一个子集的投影，其中子集的维数为 C 的秩。例如，如果：

$$C=\begin{pmatrix}0 & 1\\ 0 & 0\end{pmatrix} \tag{11.71}$$

那么 $y=Cx$ 产生 $y_1=x_2$ 和 $y_2=0$。每个 $(x_1,x_2)\in X$ 中只有 x_2 能够观测到，因为 C 的秩为 1。因此对于某些特殊情况，奇异矩阵可以测量某些状态变量，但也会使得其他一些状态不可观测。对于一个一般的奇异矩阵 C，对它的解释是 X 通过传感器投影到某个 k 维子空间，其中 k 是 C 的秩。如果 W 是奇异的，这就说明大自然的影响是有限的。干扰传感器观测时大自然所用的自由度就是 W 的秩。

简单投影传感器 通过观测 X 的特定坐标并且放弃其他一些不可观测的坐标，这样就可以定义几种平常的传感器模型。这就是例 11.4 中传感器的连续版本。例如，假设有一个机器人，它既能够在一个二维世界 $\mathcal{W}=\mathbb{R}^2$ 中滚动，也能够进行旋转。状态空间（或者位形空间）是 $X=\mathbb{R}^2\times\mathbb{S}^1$。为了便于形象描述，可以假设该机器人非常小以至于可以用一个点表示，这样可以避免构造第 4.3 节中复杂的位形空间[1]。令 $p=(p_1,p_2)$ 表示点的坐标，令 $s\in\mathbb{S}^1$ 表示它的方向。因此 $\mathbb{R}^2\times\mathbb{S}^1$ 中的状态指定为 (p_1,p_2,s)（而不是 (x,y,θ)，因为这可能与诸如 X、Y 和 Θ 这样的重要空间相混淆）。

假设机器人能够估计它自己的位置，但不知道它的方向。这就产生了位置传感器，定义为 $Y=\mathbb{R}^2$，其中 $y_1=p_1$ 和 $y_2=p_2$（也表示为 $y=h(x)=p$）。第三个状态变量 s 未知。当然可以加入任何以前考虑的大自然感测行动模型。例如大自然可能引起误差，该误差可以用高斯概率密度来建模。

同样地，只观测最后一个状态变量 s 可以产生一个罗盘或者方向传感器。在这种情况下，$Y=\mathbb{S}^1$ 和 $y=s$。可以包含大自然感测行动。例如感测到的方向可能是 y，但是只知道 $|s-y|\leq\epsilon$，常数 ϵ 是最大感测误差。高斯模型不能得到正确应用，因为它的定义域是无界的，而 $\mathbb{S}^1$ 是有界的。截短高斯分布或者使用更加合适的分布可以解决该问题。

位置和方向传感器可以推广到三维世界 $\mathcal{W}=\mathbb{R}^3$。回忆到第 4.2 节在这种情况下的状态

1 问题可以得到处理，但它只是给目前的讨论加入了不必要的复杂性。

空间是 $X = SE(3)$，该空间可以表示为 $\mathbb{R}^3 \times \mathbb{RP}^3$。位置传感器测量前三个坐标，方向传感器测量后三个坐标。在 $\mathbb{R}^3$ 中测量方向的一个传感器称为陀螺（gyroscope）。这通常是基于运动原理，也就是说它们包含一个转子，它能够抵抗由于角动量的缘故而发生的方向变化。对于用旋转关节连接起来的连杆组来说，状态空间中的一个点给出每个相互连接的连杆对之间的角度。关节编码器就是这样一种产生角度的传感器。

在第Ⅳ部分之前都没有考虑机械系统的动力学问题，但是了解几种传感器却是非常有价值的。在这些问题中，状态空间将会扩展到包括速度参数以及可能的加速度参数。在这种情况下，里程表可以感测速度向量或者标量速度。也有测量角速度的传感器，它指示旋转的速度。可以用加速度计来测量加速度参数。用这些模型，大自然感测行动可以用来考虑测量误差。

边界传感器　如果状态空间有一个边界，就像运动规划问题中 $\mathcal{C}_{free}$ 的情况，那么一些重要的边界传感器可以基于边界检测原理进行表述。图 11.12 显示了几种有趣的情况。

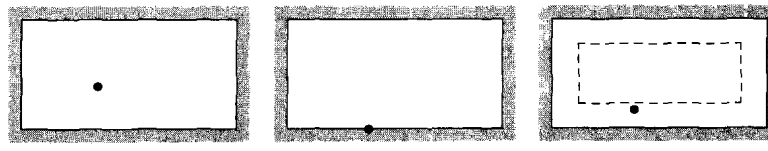


图 11.12　边界传感器指示是否与边界发生接触；在后一种情况下，一个接近传感器可以表明边界是否在一个特定的距离范围内

假设状态空间是一个定义了边界的闭集合。为了提供与运动规划之间的联系，假设 $X = \mathrm{cl}(\mathcal{C}_{free})$，即 $\mathcal{C}_{free}$ 的闭包。一个接触传感器确定是否接触到边界。在这种情况下，$Y = \{0,1\}$，如果 $x \in \partial X$，那么 h 定义为 $h(x) = 1$，否则 $h(x) = 0$。这两种情况如图 11.12（a）和图 11.12（b）所示。使用这种传感器，没有关于沿着边界在什么地方发生接触的信息。在可移动机器人学中，如果机器人与障碍物发生接触是非常糟糕的。取而代之的是通常采用接近传感器，如果状态或者位置处在 ∂X 的某个指定的常数 r 范围内时，传感器产生 $h(x) = 1$，否则 $h(x) = 0$，如图 11.12 所示。

在机器人操作、触觉接触面以及其他一些在机器和环境之间存在物理接触的应用中，会用到力传感器。除了简单的指示接触以外，力传感器还可以指示力的大小和方向。机器人模型必须从给定的状态产生预期的力的值。

地标传感器　一些重要的感测模型可以按照**地标**（landmark）来定义。地标是状态空间中一个特殊的点或者区域，传感器用某种方式可以检测到它们。可以用对地标的测量来对目前的状态进行推断。一个古老的例子是在海上利用恒星来进行导航。基于恒星相对于船的位置，可以计算出船的方向。当来到一个陌生城市的时候，你可能已经发现了地标的用处了。例如墨西哥城周边的山或者巴黎的埃菲尔铁塔都可以用来标明方向。即使巴黎的道路非常复杂，在任何可以看见埃菲尔铁塔的地方向它走去，总能够到达埃菲尔铁塔。这种模型在分析在线算法的竞争率框架中是很常见的[677]。

一般情况下，状态的集合也可以作为地标。通常的模型是将 x_G 作为一个地标。在机器人应用中，这些地标可以认为是世界 $\mathcal{W}$ 中的点。地标点可以推广到地标区域。但是为了阐

述概念，这里的思路将尽可能保持简单。以此为基础，你可以假设各种各样的推广。对于所有的地标例子，都假设 $X=\mathbb{R}^2$，并且令状态表示为 $x=(x_1,x_2)$。

例如，假设只有一个地标 $l\in X$，它的坐标 (l_1,l_2)。图 11.13 刻画了一个自引导传感器，它产生 $Y=\mathbb{S}^1$ 中的值。传感器映射是 $h(x)=\text{atan2}(l_1-x_1,l_2-x_2)$，其中 atan2 给出适当象限中的角度。

另外一种情况是 Geiger 计数传感器（辐射水平），其中 $Y=[0,\infty)$ 和 $h(x)=\|x-l\|$。在这种情况下，传感器报告与地标之间的距离，没有方向信息。



图 11.13 最基本的地标传感器只指示方向

接触传感器也可以与地标思路相结合来产生一种称为鹅卵石（pebble）的传感器。如果“接触”到这种传感器，那么它报告 1，否则报告 0。这种思路可以推广到区域。假设有一个地标区域 $X_l\subset X$，如果 $x\in X_l$，那么地标区域检测器报告 1，否则报告 0。

用目前所解释的多个地标的思路可以表述一些有趣而且有用的传感器模型。例如用 3 个不共线的自引导传感器可以重构状态。将地标区域和它们相关的检测器都植入状态空间，这样就可以产生一些有趣的问题。在可移动机器人应用领域，这可以通过将摄像机或者其他传感器放置在环境中来实现。传感器可以指示目前哪个摄像机能够看见机器人。它们也提供机器人与每个摄像机的距离信息。

深度映射传感器 在许多机器人应用中，机器人不可能有障碍物的地图。在这种情况下，用感测来学习环境和确定它在环境中的位置和方向。假设机器人放置在如图 11.14（a）所示的环境中。对于这样的问题，状态既表示机器人的位置，也表示障碍物的位置。这种情形将在第 12.3 节做更进一步的介绍。这里为这种问题给定某些传感器模型。它们与边界传感器和图 11.12 的接近传感器有关，但是当机器人没有处在边界上时候，它们将产生更多信息。

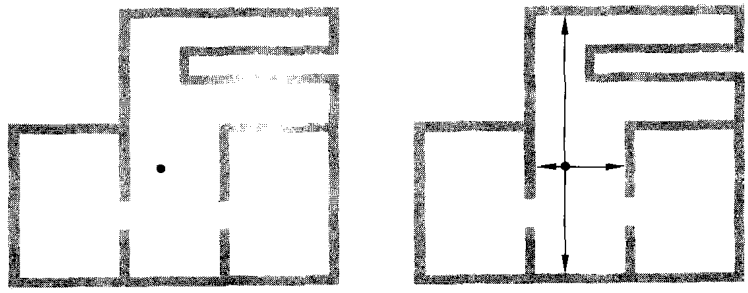
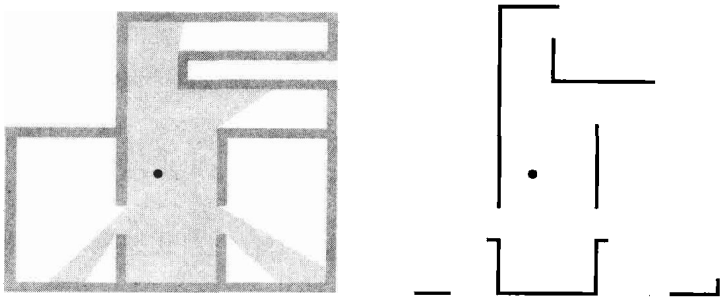


图 11.14 （a）一个可移动机器人被置于一个未知环境中；（b）用四个声纳来测量其与墙之间的距离

在可移动机器人中使用的最古老的一种传感器是声纳，它向特定方向发射一种高频声波，测量声波遇到墙后返回到声纳所用的时间（通常声纳扮演着讲话者和麦克风的角色）。基于声波的速度和飞行时间，可以估计与墙之间的距离。有时声波不会返回，这可以用大自然来建模。距离估计的误差也可以用大自然来建模。一般情况下，单个声纳的观测空间 Y 是 $[0,\infty]$，其中 ∞ 表示声波不会返回。Y 可以是飞行时间，或者它已经变换成估计的距离。如果有 k 个声纳，每个声纳指向不同的方向，那么 $Y=[0,\infty]^k$，表示从每个声纳可以读到一

个值。例如图 11.14（b）显示 4 个声纳以及它们可以测量的距离。每个观测产生 $\mathbb{R}^4$ 中的一个点。

现代激光扫描技术使得高角密度精确测距成为现实。例如 SICK LMS-200 至少每 0.5° 测量一个距离，每秒扫描全部 360° 至少 30 次。在室内环境下的测距精度通常为几个毫米的数量级。假设极限情况下有无穷多个声纳分布在 $\mathbb{S}^1$ 的每个角度。这样产生的传感器被称为测距扫描仪或者可视性传感器，它为每个 $s \in \mathbb{S}^1$ 提供一个距离测量，如图 11.15 所示。



(a)　(b)

图 11.15　一个测距扫描仪或者可视性传感器就像有无穷多个高精度声纳；为每个 $s \in \mathbb{S}^1$ 提供距离值

如果只指示深度扫描时在 $\mathbb{S}^1$ 中发生不连续的点，那么可以产生一个较弱的传感器。这里称之为缺口传感器（gap sensor），图 11.16 显示了一个例子。甚至可能是仅给定这些缺口沿着 $\mathbb{S}^1$ 的圆周顺序，而不知道它们之间的相对角度或者与每个缺口之间的距离。基于这种感测模型的规划器将在第 12.3.4 节介绍。

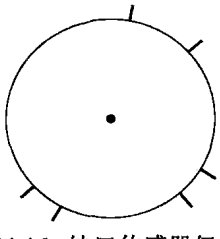


图 11.16　缺口传感器仅指示深度上发生不连续的方向，而不提供距离信息

里程表传感器　下面给出最后一种传感器，它提供了一个基于历史的传感器映射的例子，如第 11.1.1 节为离散状态空间而定义的那样。可移动机器人通常都有里程表传感器，该传感器基于轮子转过的圈数来指示机器人运动了多远。由于轮子滑动、表面缺陷以及建模误差等缘故，这种测量往往不够准确。对于一个给定的状态历史 $\tilde{x}_t$，传感器可以估计运动的整个距离。对于该模型，$Y = [0, \infty)$ 和 $y = h(\tilde{x}_t)$，其中自变量 $\tilde{x}_t$ 是到时刻 t 为止的整个状态历史。另外一种建模里程表的方法是用一个传感器指示从上一个阶段到目前所运动距离的估计值。这避免了对整个历史的依赖性，但是它很难对距离估计值的误差进行建模。

在许多文献中（例如文献[352]），行动历史 $\tilde{u}_t$ 可以被认为是里程表。这种阐述在某些应用中是合适的。例如每个行动可以对应于蹬自行车踏板完整一周。踏板旋转的圈数可以作为里程表的读数。因为这种信息已经在 η_k 中出现，所以本书没有将它作为感测过程的一部分来建模。对于自行车，可能有一个里程表传感器，它的测量是基于踏板运动以外的因素。将该传感器建模为一个基于历史的传感器是合适的。

另外一种基于历史的传感器是观测一个时钟，它指示从初始阶段到目前所经过的时间。结合诸如机器人的速度等其他信息，这可以对状态进行推断。

11.5.2 简单投影的例子

本节给出 I-空间的例子，其中的传感器映射是 $y=h(x)$，h 是一个投影，它揭示某些状态变量，同时也隐藏另外一些状态变量。这些例子都涉及连续时间，它们的注意力主要集中在非确定的 I-空间 $\mathcal{I}_{ndet}$。假设没有行动，即 $U=\phi$。但是允许大自然行动 $\Theta(x)$。因为没有机器人行动和大自然感测行动，所有的不确定性都产生于这样一个事实：h 是一个投影，而且影响状态转移方程的大自然行动未知。这是一类非常重要的问题。如果允许对行动集合 U 进行某些控制，那么会使例子更加复杂，但是这里的目的是阐述 I-空间的概念。因此没有必要使问题更加复杂化。

例 11.20（在正弦曲线上运动） 假设状态空间是平面中正弦曲线上点的集合：

$$X=\{(x_1,x_2)\in\mathbb{R}^2 \,|\, x_2=\sin x_1\} \tag{11.72}$$

令 $U=\phi$，即不产生行动历史。观测空间为 $Y=[-1,1]$，并且传感器映射产生正弦曲线上点的高度 $y=h(x)=x_2$，如图 11.17 所示。

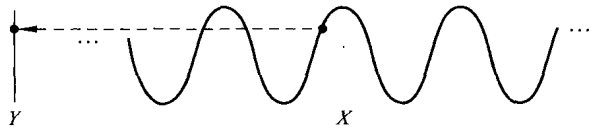


图 11.17 状态空间是 $\mathbb{R}^2$ 中正弦曲线所绘制出的点的集合

大自然行动空间是 $\Theta=\{-1,1\}$，其中 -1 表示沿着正弦曲线在 $-x_1$ 方向以单位速度运动，1 表示沿着正弦曲线在 x_1 方向以单位速度运动。因此对于某大自然行动历史 $\tilde{\theta}_t$，沿着曲线运动的点所形成的状态轨迹 $\tilde{x}_t$ 可以用积分来确定。

历史 I-状态为 $\eta_t=(X_0,\tilde{y}_t)$，它包括初始条件 $X_0\subseteq X$ 以及到 t 时刻为止的观测历史 $\tilde{y}_t$。该问题的非确定的 I-状态非常有趣。对于每个观测 y，原像 $H(y)$ 是对应于 X 与高度为 y 的水平线相交的点的一个可数无限集合，如图 11.18 所示。

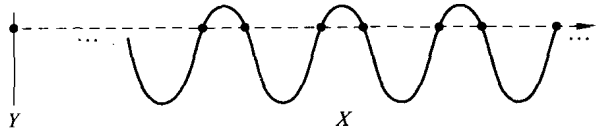


图 11.18 一个观测 y 的原像 $H(y)$是沿着 X 的点的一个可数无限集合

该问题的不确定性总可以用交点的数量来刻画，这些交点可能包含真实状态。假设 $X_0=X$。在这种情况下，没有能够减小不确定性的状态轨迹。当点沿着 X 移动时，由于有传感器，所以高度总是已知的，但是 x_1 坐标的范围只能够缩小为任意交点。

如果假设 $X_0=\{x_0\}$，其中 x_0 是沿着 X 的某个特定的点。如果 y 在从 $t=0$ 开始的某时间周期内都处在 $(0,1)$ 中，那么 $x(t)$ 是已知的，因为包含状态的正弦曲线片段是已知的。但是如果点到达一个极值，即 $y=0$ 或者 $y=1$，那么就不知道点究竟走哪条路。从该点，传感器不能区分向 $-x_1$ 方向移动还是向 x_1 移动。因此不确定性就增加了，如图 11.19 所示。得到观测 $y=1$ 后，根据当 $y=1$ 时大自然所采取的行动，目前的状态有两种可能性，因此非确定的 I-状态包含两个状态。如果继续移动到 $y=1$，那么在非确定的 I-状态中有 4 个状态。但

本例中的不确定性只能增加，无法使用传感器来减小非确定的 I-状态的数量。 ■

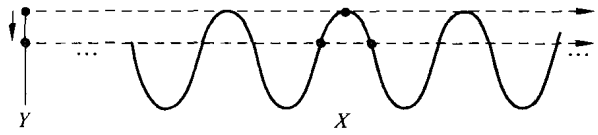


图 11.19　当得到 $y=1$ 或者 $y=-1$，那么会发生分叉；这增加了状态的不确定性

前面的例子可以推广到观测在平面拓扑图中移动的点的单个坐标，如图 11.20（a）所示。大多数模型与例 11.20 相同，只是现在的状态空间是一个图。需要对大自然行动的集合 $\Theta(x)$ 进行扩展，这样一来，如果 x 是图的一个顶点，那么每条入射边都有一个输入。这是点运动的一些可能的方向。

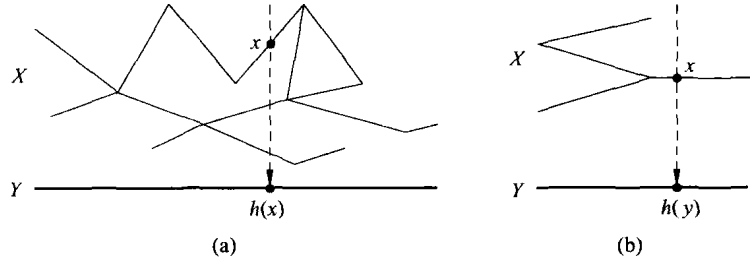


图 11.20　（a）只观测一个坐标的同时，假设试图推断平面图上一个点的位置；（b）这个例子涉及沿着一个有 4 条边的图进行运动的点；当点在最右边的边上，那么没有不确定性，但是当点沿着其他边运动时将会存在不确定性

例 11.21（观测图上的一个点）　考虑图 11.20（b）所示的图，其中有四条边[1]。当点在图的最靠右的边中移动时，状态可以由传感器来推断确定。集合 $H(y)$ 包含最靠右的边上的一个单点。如果点在其他一条边中移动，那么 $H(y)$ 包含三个点，每个点对应 y 以上的一条边。这产生了非确定的 I-状态的几种可能的情况，如图 11.21 所示。对于非确定的 I-状态来说，这些边的任意子集都是可能的，除了空集以外。

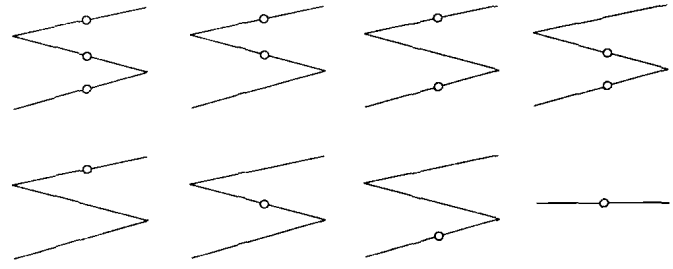


图 11.21　点所处在的边的不同集合可以得到非确定的 I-空间 $\mathcal{I}_{ndet}$ 的片段

图 11.21 所示的 $\mathcal{I}_{ndet}$ 的 8 个片段用一种非常有趣的方法连接起来。假设点在最靠右的边上并向左移动。穿越顶点以后，I-状态必须是图 11.21 右上部所示的情况，它指出：点将在两条边中的一条上。如果点正好从靠左的边的 I-状态之一经过，那么图 11.20 的右下部

1　与 Rob Ghrist 进行有益的讨论后，这个例子得到了改进提高。

显示的 I-状态总能够到达，但是不可能返回到左边相同的 I-状态。因此一般情况下，对 $\mathcal{T}_{ndet}$ 有方向约束，并且从图 11.20 的左下部的 I-状态直接向右移动不可能到达右下部的 I-状态。这是因为从图的结构知道这是不可能的。 ■

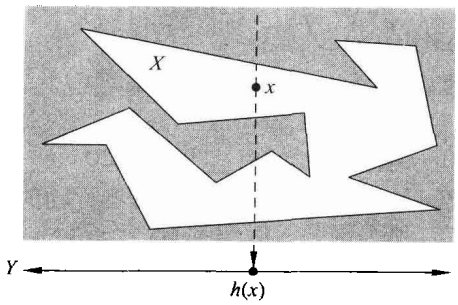


图 11.22 图可以推广到一个平面区域，并且还将得到非确定的 I-空间中的层次

对图的例子进行推广可以反映在机器人技术和其他领域中出现的一大类问题。例如图 11.22 显示了一个多边形，一个点可以在其中移动。仅观测到一个坐标，所得到的非确定的 I-空间具有与例 11.21 中的相似。这些思路可以推广到任意维数。用简单的投影传感器可以构造有趣的模型，例如第 11.5.1 节的位置传感器或者罗盘。在第 12.4 节中，这将出现在追逃对策中，该对策使用可视性传感器来发现可移动目标。

11.5.3 带大自然感测行动的例子

本节阐述大自然感测行动的影响，但是只针对非确定的情况而言。计算概率 I-状态的一般方法将在第 11.6 节介绍。

例 11.22（在圆面上面还是下面？） 本例涉及连续时间。假设任务是收集信息、确定状态轨迹是在状态空间的某指定区域上面还是在下面行进，如图 11.23 所示。

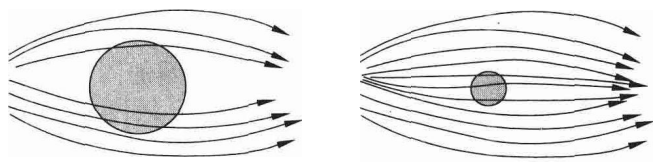


图 11.23 （a）总是可能确定状态轨迹是从指定区域的上方还是下方经过；（b）确定状态轨迹是从孔的上方还是下方经过的这种能力取决于特定的观测：在某些情况下，它是不可能的

令 $X=\mathbb{R}^2$。通过对速度 $(\dot{x},\dot{y})$ 进行积分产生运动，速度表示为 $\dot{x}=\cos(u(t)+\theta(t))$ 和 $\dot{y}=\sin(u(t)+\theta(t))$。为简单起见，假设 $u(t)=0$，这是一个向右移动的命令。大自然行动 $\theta(t)\in\Theta=[-\pi/4,\pi/4]$ 干扰输出结果。机器人试图向正 x_1 方向上移动，但是大自然的干扰使得很难预测 x_2 的方向。如果没有大自然的影响，那么 x_2 坐标应该没有变化；但是有大自然的情况下，x_2 方向的误差随时间 t 增长。图 11.24 阐述了可能产生的运动。

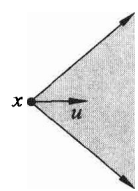


图 11.24 大自然干扰命令方向，所以真实的状态可能在圆形部分的任何地方

构造能够减小不确定性增长的圆锥体的传感器观测，使用图 11.11 感测模型和假设测量误差 r 为 1。假设在 $\mathbb{R}^2$ 中有一个半径大于 1 的圆面，如图 11.23（a）所示。因为真实的状态与测量状态之间的距离从来不会超过 1，所以总可能确定状态是从圆的上面还是下面经过。多个可能的观测历史如图 11.23（a）所示。观测历史不必是连续的，为了方便而绘制成连续的。对于一个半径小于 1 的圆来说，

可能存在一些观测历史，它们不可能确定真实状态是从圆的上面还是下面经过，如图 11.23（b）所示。对于其他观测历史，仍然可能进行这种判断。例如从图 11.23（b）所示的最上面的轨迹可以知道：真实状态当然是从圆盘的上面经过的。 ■

例 11.23（可移动机器人的一个简单模型） 假设机器人被建模为一个在 $X=\mathbb{R}^2$ 中移动的点。除了用离散阶段代替连续时间以外，感测模型与例 11.22 中的相同。可以假设每个阶段代表一个固定时间区间（例如 1 秒）。

为了控制机器人，运动命令以行动 $u_k \in U=\mathbb{S}^1$ 的形式给出。大自然用两种方式干扰运动：（1）机器人试图运动某距离 d，但是有误差 $\varepsilon_d>0$，知道真实距离 d' 满足 $|d'-d|<\varepsilon_d$；（2）机器人试图沿一个方向 u 移动，但存在误差 $\varepsilon_u>0$，知道真实方向 u' 满足 $|u'-u|<\varepsilon_u$。这两种相互无关的误差可以通过定义一个二维大自然行动集合 $\Theta(x)$ 来建模。然后定义转移方程使得前向投影 $F(x,u)$ 如图 11.25 所示。

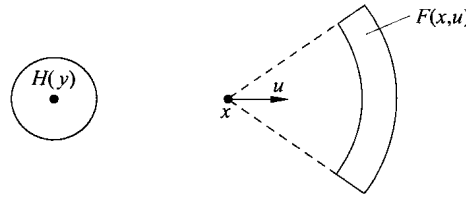


图 11.25　一个简单的可移动机器人的运动模型，其中感测模型由图 11.11 给出，那么大自然干扰运动就产生了一个不确定性区域，该区域是一个圆环

现在构造某些非确定的 I-状态。假设初始状态 x_1 已知，历史 I-状态采用如下形式：

$$\eta_k=(x_1,u_1,\cdots,u_{k-1},y_1,\cdots,y_k) \tag{11.73}$$

因为初始状态已知，所以第一个传感器观测 y_1 是无用的。应用方程（11.29）产生 $H(y_1)\cap\{x_1\}=\{x_1\}$。假设应用行动 $u_1=0$，指示机器人应该向右水平移动。方程（11.30）产生 $X_2(\eta_1,u_1)$，这看起来与图 11.25 所示的 $F(x,u)$ 是相同的。假设得到如图 11.26（a）所示的一个观测 y_2。使用该观测，那么 $H(y_2)$ 和 $X_2(\eta_1,u_1)$ 的交集就是 $X_2(\eta_2)$，如图 11.26（b）所示。

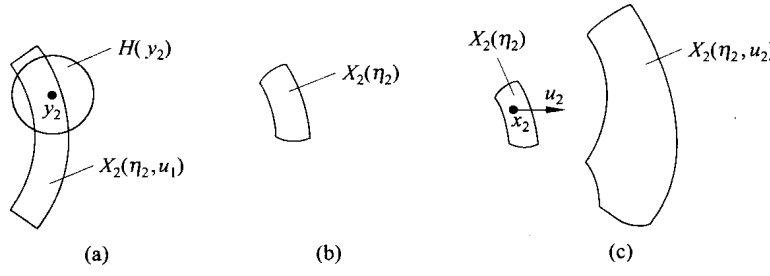


图 11.26　（a）将来自于 $X_2(\eta_1,u_1)$ 和观测 y_2 的信息进行组合；（b）必须取 $X_2(\eta_1,u_1)$ 和 $H(y_2)$ 的交集；（c）行动 u_2 产生了一个复杂的非确定的 I-状态，它是所有 $x_2\in X_2(\eta_2)$ 上 $F(x_2,u_2)$ 的并集

下一步会更加复杂。假设 $u_2=0$ 并且应用式（11.30）从 $X_2(\eta_2)$ 计算 $X_3(\eta_2,u_2)$。对于所有可能的 $x_2\in X_2(\eta_2)$，取 $F(x_2,u_2)$ 的并集就得到图 11.26（c）所示的形状。所产生的形状由圆弧和线段组成（见习题 13）。一旦得到 y_3，那么交集产生 $X_3(\eta_3)=X_3(\eta_2,u_2)\cap H(y_3)$，如图 11.27 所示。过程重复进行所需要的阶段数。图 11.26（c）中区域的复杂性为第 11.4.3 节的近似方法提供了很好的动机。例如非确定的 I-状态可以用椭球区域来近似。 ■

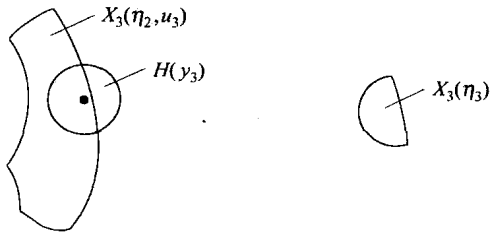


图 11.27 传感器观测 y_3 以后，必须取 $X_3(\eta_2,u_2)$ 和 $H(y_3)$ 的交集

11.5.4 在没有传感器的情况下获得信息

对于某些问题来说，不用传感器也可能减小不确定性。请回忆例 11.17。这似乎不符合我们的直觉，因为关于状态的信息只能从感测而来。但是有可能从已经执行的行动以及在状态转移中应该产生的效果中收集信息。本节提出的例子是受到关于无传感器操作规划的启发[323,398]，在第 12.5.2 节将详细介绍无传感器的操作规划。这种规划强调用 I-空间进行推理的优越性，而不是要求进行准确的状态估计。

例 11.24（托盘倾斜） 状态空间 $X \subset \mathbb{R}^2$ 指示在一个平坦表面上滚动的球的位置，如图 11.28 所示。球在图 11.28 中所示多边形区域内滚动。可以假设球在一个托盘中滚动，盘上粘有几个挡板限制了它的运动。如果盘被倾斜，那么可以想象到球会沿着重力所产生的方向滚动。

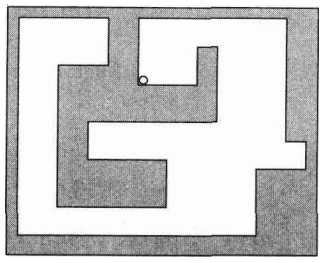


图 11.28 一个托盘的顶视图，必须倾斜它来将小球滚动到所希望的角落

托盘的倾斜可以认为是机器人所选择的一个行动。假设球的初始位置（初始状态）未知，并且没有传感器来估计状态。任务是通过倾斜将球放置到图 11.28 所示的位置，不管初始位置在哪里。

问题可以用连续时间来建模，但是这使得设计复杂化。如果托盘向一个特定的方向倾斜，那么球向一个方向滚动，可能会沿着边界，直到停止。这可以认为是一个离散阶段转移：球处在某停止状态，倾斜托盘，那么球进入另外一个停止状态。因此使用的是一个离散阶段状态转移方程 $x_{k+1}=f(x_k,u_k)$。

为了描述倾斜行动，我们从 $\mathbb{S}^2$ 的上半部选择托盘的向上法向量的方向。但是这可以减为一维集合，因为倾斜的陡度并不重要，只要球能够滚向一个新的平衡状态。因此行动集合可以认为是 $U=\mathbb{S}^1$，其中一个方向 $u \in \mathbb{S}^1$ 指示球由于重力而滚动的方向。在应用任何行动之前，假设托盘开始是水平的（它的法线平行于重力方向）。实际上，我们应该更仔细地对两个行动之间的托盘运动进行建模；在这里没有如此，因为例子仅仅是为了进行说明而已。引入可以指示托盘方向的新的状态变量或者使用连续时间行动就可以实现更多的情况。在这种情况下，行动提供所需要的状态信息，这意味着行动函数必须是连续的。但在这里仅简单地假设应用的是 $\mathbb{S}^1$ 中的行动序列。

初始条件是 $X_1=X$，并且历史 I-状态为：

$$\eta_k=(X_1,u_1,u_2,\cdots,u_{k-1}) \tag{11.74}$$

因为没有观测，所以 I-空间中的路径是可以预测的。因此对于任意所希望的 K，规划 π 仅仅是一个行动序列 $\pi=(u_1,u_2,\cdots,u_K)$。

根据非确定的 I-状态进行推理就可以解决这个任务，其中每个 I-状态对应于球的可能位置的一个集合。图 11.29 所示的 6 个行动的序列就足以保证球将停留在目标位置，不管初始位置在哪里。■

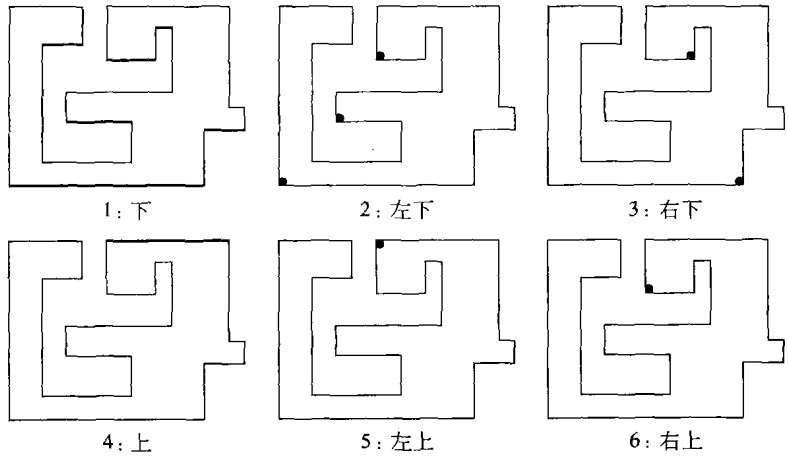


图 11.29　显示了一个规划，它通过六次倾斜来将球放置在所需要的位置上，不管它的初始位置在哪里，尽管没有传感器；深黑线和黑点指示球的可能的位置：非确定的 I-状态；在每个图片下面都写出了采取行动后球滚动的方向

11.6　计算概率信息状态

概率 I-状态在实际中可能十分复杂，因为 $\mathcal{I}_{prob}$ 的每个元素都是一个概率分布或者密度函数。因此为有效地计算概率 I-状态而做了大量的工作。本节可以认为是第 11.2.3 节（和第 11.4 节中连续状态空间的部分）的继续。第 11.6.1 节将介绍卡尔曼滤波器，它为概率 I-状态提供了一个非常好的计算方法。它是为状态转移和传感器映射都是线性的以及所有大自然行动用多变量高斯密度来建模等这类问题而设计的。第 11.6.2 节将介绍基于采样的一般规划方法，虽然这些方法是近似的，但是可以应用于很多类型的问题中。粒子滤波是这些方法之一，它在可移动机器人的定位中应用得非常广泛。

11.6.1　卡尔曼滤波

本节介绍一个衍生 I-空间的应用最成功、最广泛的例子，该空间显著地缩减历史 I-空间。在 f 和 h 都是线性函数，并且 $p(\theta)$、$p(\psi)$ 和 $p(x_1)$ 是高斯的这种特殊情况下，所有概率 I-状态都变成高斯的。这意味着概率 I-空间 $\mathcal{I}_{prob}$ 不需要表示每个可能的概率密度函数。概率 I-状态总在 $\mathcal{I}_{prob}$ 的子空间中，该子集只对应于高斯分布。子空间用 $\mathcal{I}_{gauss}$ 表示。这意味着应用 I-映射 $\kappa_{mom}:\mathcal{I}_{prob}\rightarrow\mathcal{I}_{gauss}$ 时不会失去任何信息。

该模型称为线性高斯的（或者 LG）。$\mathbb{R}^n$ 上的每个高斯密度完全由其 n 维均值向量 μ 和 $n \times n$ 对称协方差矩阵 Σ 确定。因此 $\mathcal{I}_{gauss}$ 可以认为是 $\mathbb{R}^m$ 的一个子集，其中 $m = 2n + \binom{n}{2}$。例如，如果 $X = \mathbb{R}^n$，那么 $\mathcal{I}_{gauss} \subset \mathbb{R}^5$，其中两个独立参数指定均值、3 个独立参数指定协方差矩阵（因为对称，所以不是 4 个参数）。在第 11.4.3 节曾经提到过在一般情况下可以使用基于矩的近似方法。但是对于一个 LG 模型来说，重要的是要记住：$\mathcal{I}_{gauss}$ 是 $\mathcal{I}_{prob}$ 的一个精确表示。

除了 $\mathcal{I}_{prob}$ 能够缩减以外，κ_{mom} 还是一个充分 I-映射，并且对于完全根据计算的均值和方差来递推地更新衍生 I-状态来说，表达方式也很简单。这意味着我们可以直接用 $\mathcal{I}_{gauss}$，而不需要关心原始历史或者第 11.4.1 节概率 I-状态的一般公式。这里没有对更新表达式进行完整解释，因为这很冗长乏味，但并不困难，在任何关于随机控制的教科书中都有这方面的介绍（例如文献[97,567]）。

对于卡尔曼滤波，所需要的空间都是欧氏空间，但是空间的维数是不同的。因此令 $X = \mathbb{R}^n$，$U = \Theta = \mathbb{R}^m$ 以及 $Y = \Psi = \mathbb{R}^r$。因为卡尔曼滤波依赖于线性模型，所以所有的表达式都可以用矩阵变换来表示。令 A_k、B_k、C_k、G_k 和 H_k 分别表示一个实值常数矩阵，它们可能是奇异的，也可能不是。矩阵的维数可以从它们所处的方程进行推断（维数必须满足矩阵乘法法则）。下标 k 用来指示在每个阶段可能用到不同矩阵。在许多应用中，矩阵在每个阶段都是相同的，在这种情况下它们可以被表示为 A、B、C、G 和 H。因为卡尔曼滤波能够处理更一般的情况，所以这里包含了下标。

一般情况下，状态转移方程 $x_{k+1} = f_k(x_k, u_k, \theta_k)$ 定义为：

$$x_{k+1} = A_k x_k + B_k u_k + G_k \theta_k \tag{11.75}$$

其中矩阵 A_k、B_k 和 G_k 具有相应的维数。符号 f_k 用来代替 f，因为即使 f 在每个阶段都是不同的，卡尔曼滤波器也能够正常工作。

例 11.25（线性高斯的例子） 作为式（11.75）的一个简单例子，假设 $X = \mathbb{R}^3$ 和 $U = \Theta = \mathbb{R}^2$。一个特殊的情况是：

$$x_{k+1} = \begin{pmatrix} 0 & \sqrt{2} & 1 \\ 1 & -1 & 4 \\ 2 & 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} u_k + \begin{pmatrix} 1 & 1 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \theta_k \tag{11.76}$$

■

传感器映射 $y_k = h_k(x_k, \psi_k)$ 的一般形式为：

$$y_k = C_k x_k + H_k \psi_k \tag{11.77}$$

其中矩阵 C_k 和 H_k 具有相应的维数。符号 h_k 用来代替 h，因为每个阶段可以使用不同的传感器映射。

到目前为止已经给定了模型的线性部分。接下来确定模型的高斯部分。在每个阶段，大自然行动 θ_k 和 ψ_k 建模为零均值的高斯白噪声。因此它们都有一个相应的协方差矩阵，分别用 Σ_θ 和 Σ_ψ 表示。使用给定的模型、从 X 上的一个初始高斯密度开始，产生的所有概率 I-状态都将是高斯的[567]。

$\mathcal{I}_{gauss}$ 中的每个衍生 I-状态都可以用一个均值和协方差来表示。令 μ_k 和 Σ_k 表示 $p(x_k|\eta_k)$ 的均值和协方差。本节其余部分中给出的表达式定义了一个衍生信息转移方程，该方程在给定 μ_k、Σ_k、u_k 和 y_{k+1} 情况下计算 μ_{k+1} 和 Σ_{k+1}。过程从计算初始条件下的 μ_1 和 Σ_1 开始。

假设给定的初始条件代表了 $\mathbb{R}^n$ 上的一个高斯密度。用 μ_0 和 Σ_0 表示。第一个 I-状态融合了第一个观测 y_1，I-状态的计算可通过 $\mu_1 = \mu_0 + L_1(y_1 - C_1\mu_0)$ 和：

$$\Sigma_1 = (I - L_1C_1)\Sigma_0 \tag{11.78}$$

其中 I 是单位矩阵，同时：

$$L_1 = \Sigma_0 C_1^T (C_1\Sigma_0 C_1^T + H_1\Sigma_\psi H_1^T)^{-1} \tag{11.79}$$

尽管 L_1 的表达式很复杂，但是注意到所有矩阵已经作为模型的一部分给定了。只是需要矩阵求逆，求逆有时会导致数值不稳定。能够减小这种问题的可选表达方式见文献[567]或者其他参考文献。

接下来给出从第 k 到第 $k+1$ 阶段的迭代更新。使用 μ_k，下一个阶段的均值为：

$$\mu_{k+1} = A_k\mu_k + B_ku_k + L_{k+1}(y_{k+1} - C_{k+1}(A_k\mu_k + B_ku_k)) \tag{11.80}$$

其中 L_{k+1} 将稍后给出定义。协方差的计算分两个步骤；一是基于应用 u_k，二是从 y_{k+1} 而来。因此应用 u_k 以后，协方差变成：

$$\Sigma'_{k+1} = A_k\Sigma_k A_k^T + G_k\Sigma_\theta G_k^T \tag{11.81}$$

得到 y_{k+1} 以后，协方差 Σ_{k+1} 从 Σ'_{k+1} 计算而来：

$$\Sigma_{k+1} = (I - L_{k+1}C_{k+1})\Sigma'_{k+1} \tag{11.82}$$

L_k 的表达式是：

$$L_k = \Sigma'_k C_k^T (C_k\Sigma'_k C_k^T + H_k\Sigma_\psi H_k^T)^{-1} \tag{11.83}$$

为了得到 L_{k+1}，将 $k+1$ 代替式（11.83）中的 k。注意到用式（11.80）计算 μ_{k+1}，必须首先计算 Σ'_{k+1}，因为式（11.80）依赖于 L_{k+1}，而 L_{k+1} 又依赖于 Σ'_{k+1}。

卡尔曼滤波器最平常的应用是用 μ_k 来提供状态 x_k 的可靠估计值。对于用闭环形式表达的 LG 系统，可以得到二次型代价泛函指标下的最优期望代价反馈规划，见第 15.2.2 节。这种模型通常被称为 LQG，即线性的、二次型代价和高斯的。最优反馈规划可以直接用 μ_k 表示，而不需要 Σ_k。这表示 I-空间可能缩小到 X，但是相应的 I-映射不是充分的。依然需要协方差来计算均值，就像式（11.80）和式（11.83）。因此一个最优规划可以指定为 $\pi: X \to U$，但是需要表示 $\mathcal{I}_{gauss}$ 中的衍生 I-状态以使 I-映射成为充分的。

卡尔曼滤波器给线性高斯模型提供了非常好的解决方案。它也经常能够非常成功地应用于甚至不满足这些条件的问题中。这称为扩展卡尔曼滤波。它的成功可以解释为：用基于二阶矩近似的方法，概率 I-空间可以用均值和协方差来近似。一般情况下，这种近似可能是不合适的，但是在实际情况下它的应用却十分广泛。

11.6.2　基于采样的方法

因为连续空间上概率 I-空间的计算涉及复杂的高维积分，那么使用基于采样的方法的愿望就非常强烈。如果问题是非线性的和/或者非高斯的，基于采样的方法可能是仅有的计

算概率 I-状态的实用方法。这里考虑两种方法：基于栅格的采样和粒子滤波。这里描述的最常用的一个应用就是可移动机器人的定位问题，这将在第 12.2 节进行介绍。

基于栅格的方法 数值计算概率 I-状态最直接的方法可能是在栅格上对概率密度函数进行近似，并用数值积分来计算式（11.57）和式（11.58）中的积分。

栅格可以用来计算离散概率分布，该离散分布是对连续概率密度函数的近似。例如考虑采用图 5.5（a）所示的 Sukharev 栅格，或者一个为适应状态空间而修改的相似栅格。考虑用一个有限集合 $S \subset X$ 来近似某概率密度函数 $p(x)$。包围每个点的 Voronoi 区域可以认为是一个拥有概率质量的“桶”。概率与每个采样点有关，定义为 Voronoi 区域上 $p(x)$ 的积分。这样，采样点 S 及其离散概率分布 $P(s)$ 就近似了 X 上的 $p(x)$，对于所有 $s \in S$。令 $P(s_k)$ 表示第 k 阶段栅格采样点集 S_k 上的概率分布。

在初始时，通过数值计算每个采样点的 Voronoi 区域上 $p(x_1)$ 的积分，这样就从 $p(x)$ 计算了 $P(s)$。也可以这样来进行估计：从密度 $p(x_1)$ 随机抽取采样点，然后记录落入每个 Voronoi 区域内的采样点的数量。对所有 Voronoi 区域中的数量进行归一化就可以产生概率分布 $P(s_1)$。根据所需精度的不同，有很少点或者没有点的 Voronoi 区域可以从将来的计算中排除掉。令 S_1 表示非零概率的采样点。

现在假设已经在 S_k 上计算了 $P(s_k|\eta_k)$，任务是在给定 u_k 和 y_{k+1} 的情况下，计算 $P(s_{k+1}|\eta_{k+1})$。对 $p(x_{k+1}|x_k,u_k)$ 的离散近似 $P(s_{k+1}|s_k,u_k)$ 可以用上面描述的栅格和“桶”来计算。到这里式（11.57）所需要的密度已经用离散分布来近似了。那么就可以在 S_k 上应用式（11.38）来得到 S_{k+1} 上一个基于栅格的分布（任何没有包含足够概率质量的桶都可以抛弃掉）。所产生的分布是 $P(s_{k+1}|\eta_k,u_k)$，下一步是考虑 y_{k+1}。再次计算离散分布；在这种情况下，通过使用栅格采样点可以用 $P(s_{k+1}|y_{k+1})$ 来近似 $p(x_{k+1}|y_{k+1})$。这使得式（11.58）被其离散化式（11.39）所代替，所以对采样点应用式（11.39）。所产生的分布 $P(s_{k+1}|\eta_{k+1})$ 表示近似概率 I-状态。

粒子滤波 离散分布可以用采样点来进行估计。实际上，甚至不需要仔细考虑采样点上的 Voronoi 区域。可以直接用从初始概率密度 $p(x_1)$ 进行随机抽取所得到的采样点集合来进行估计。这种方法称为**粒子滤波**（particle filtering），它在试验性的可移动机器人的应用中表现出了良好的性能。回忆图 1.7，并参见第 12.2.3 节。

令 $S \subset X$ 表示采样点的一个有限集合。在 S 上定义一个概率分布。采样点的集合和它的概率分布一起被认为是 X 上概率密度的一个近似。因为 S 用来表示概率 I-状态，所以令 P_k 表示 S_k 上的概率分布，它是用第 k 阶段的历史 I-状态 η_k 来计算的。因此在每个阶段会有一个新的采样点集 S_k 和概率分布 P_k。

计算概率 I-状态更新的一般方法过程如下。对于 m 次迭代，进行：

1. 按照分布 P_k 选择一个状态 $x_k \in S_k$。
2. 通过根据密度 $p(x_{k+1}|x_k,u_k)$ 来生成一个采样点，从而为 S_{k+1} 生成一个新的采样 x_{k+1}。
3. 分配权重 $w(x_{k+1}) = p(y_{k+1}|x_{k+1})$。

m 次迭代以后，归一化 S_{k+1} 上的权重得到一个有效的概率分布 P_{k+1}。当 m 趋近无穷时，结果能够收敛于真实概率 I-状态。也有其他收敛更快的方法[539]。应用粒子滤波的主要困难在于：对于某些问题来说，它很难保证采样点充分集中在最需要的地方。这是困扰许多

基于采样的算法的一个普遍性的问题，包括第 5 章的运动规划算法。

11.7　对策论中的信息空间

用本章的 I-空间概念来统一第 10.5 节的序贯对策论中的概念。注意力集中在对对策论中的信息进行建模。因为每个局中人有其自身的参考坐标系和 I-空间，这使得问题复杂化。诸如鞍点或者 Nash 平衡点等对策的解都依赖于进行决策时每个局中人的可用信息。并行于第 10.5 节，本节首先介绍对策树中的 I-状态，然后介绍状态空间上对策的 I-状态。本节的内容仅限于状态空间和阶段都是有限的这种情况。I-空间的表述很自然地扩展到可数无限或者连续的状态空间、行动空间和阶段[60]。

11.7.1　对策树中的信息状态

回忆第 10.5.1 节，在对策树中表述一个序贯对策的很重要的一部分内容是指定信息模型。这在表述 10.3 中的步骤 4 中进行。在第 10.5.1 节考虑了 3 种信息模型：交替对策、逐阶段对策和开环对策。这些以及其他一些信息模型都可以用 I-空间来描述。

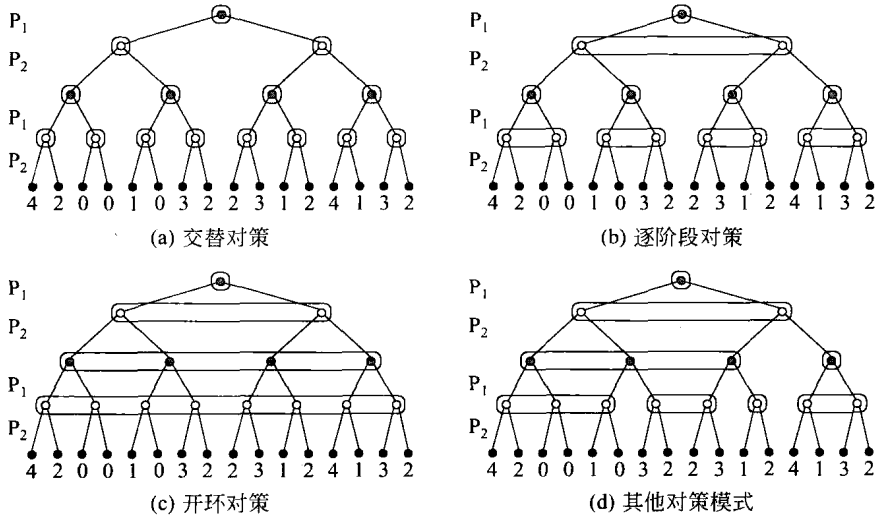
从第 11.1 节可以清楚地看到：I-空间的定义总是关于状态空间的。即使第 10.5.1 节没有正式引入状态空间，定义它也并不困难。令状态空间 X 为对策树中所有顶点的集合 N。假设两个局中人进行的是序贯零和对策。采用第 10.5.1 节中的符号，N_1 和 N_2 分别是 P_1 和 P_2 的决策顶点。考虑 N 上的非确定的 I-空间 $\mathcal{I}_{ndet}$。令 η 表示一个非确定的 I-状态，因此每个 $\eta \in \mathcal{I}_{ndet}$ 是 N 的一个子集。

当进行决策时有许多可能的情况使局中人感到困惑。例如，如果 η 包含来自于 N_1 和 N_2 的顶点，那么局中人将不知道是否已经轮到他进行决策了。如果 η 还包含某叶子顶点，那么对策结束时局中人可能还没有意识到。多数对策树的表述避免了这种奇怪的状况。通常假设局中人至少知道什么时候轮到它进行对策。也通常假设它们知道对策进行到哪个阶段。这就从 $\mathcal{I}_{ndet}$ 中排除了许多集合。

当进行对策时，每个局中人有它自己的非确定的 I-状态，因为局中人可能相互隐瞒它们的对策。令 η_1 和 η_2 分别表示 P_1 和 P_2 的非确定的 I-状态。对于每个局中人来说，已经排除掉 $\mathcal{I}_{ndet}$ 中一些集合，并且还附加一个约束 $\eta_i \subseteq N_i$，$i=1$ 和 $i=2$。当轮到某局中人进行决策时，我们只关心该局中人的 I-状态。因此非确定的 I-状态应该告诉我们：当 P_i 面临一个决策时，N_i 中的哪个决策顶点是可能的。令 $\mathcal{I}_1$ 和 $\mathcal{I}_2$ 分别表示 P_1 和 P_2 的非确定的 I-空间，其中所有不可能的 I-状态都已经排除掉。

通过围绕属于相同 I-状态的顶点，I-空间 $\mathcal{I}_1$ 和 $\mathcal{I}_2$ 通常直接定义在对策树上。它们形成了树的每个层次中顶点的划分（除了叶子以外）。实际上，$\mathcal{I}_i$ 甚至为每个局中人形成了对 N_i 的一个划分。图 11.30 显示了用这种方式为图 10.13 中的例子所指定的 4 种信息模型。前 3 个直接对应于第 10.5.1 节中的模型。在交替对策模型中，每个局中人总是知道对策顶点。这对应于完美状态信息的情况。在逐阶段对策模型中，P_1 总知道决策顶点；P_2 知道 P_1 上次决策所基于的顶点，但是它不知道所选择的分支。开环模型表示拥有最少信息的一种情况。

只有 P_1 在开始的时候知道其决策顶点。自此以后就没有关于所选行动的任何信息。实际上，局中人甚至记不住它以前的行动。图 11.30（d）显示了一个与这 3 种都不相符的信息模型。在这种模型中产生了非常奇怪的行为。如果 P_1 和 P_2 初始选择右分支，那么就知道所产生的决策顶点；但是如果 P_2 选择了左分支，那么 P_1 将会忘记它自己采取了什么行动（好像 P_2 的行动引起了 P_1 的健忘症）。这里有一个单阶段的例子：



(a) 交替对策　(b) 逐阶段对策

(c) 开环对策　(d) 其他对策模式

图 11.30　阐述图 10.13 中对策的几种不同的信息模型

例 11.26（一个不寻常的信息模型） 图 11.31 显示了一个对策，它与第 10.5.1 节中的任意信息模型都不相符。它实际上是图 10.12 所考虑的对策的一个变化形式。该对策是一种混合体，它看起来有点像交替对策模型，又有点像逐阶段对策模型。这种特殊的问题可以用通常的方法自下而上地解决。对于 P_2 进行决策的那个层次，为每个非确定的 *I*-状态计算一个值。左边的 *I*-状态的值为 5，它对应于 P_1 选择 1、P_2 的响应为 3。右边 *I*-状态的值为 4，这是从 P_1 和 P_2 之间进行的 2×3 矩阵对策中的确定性的鞍点产生而来。整个对策有一个确定性的鞍点，在鞍点处 P_1 选择 3、P_2 选择 3。这导致了对策的值为 4。 ■

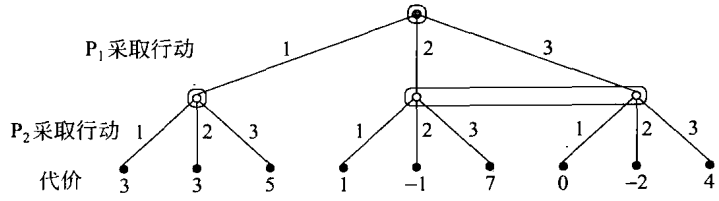


图 11.31　一个单阶段对策，它的信息模型与第 10.5.1 节中的不同

现在直接将规划定义为 *I*-空间上的函数。P_1 的一个（确定性的）规划定义为 $\mathcal{I}_1$ 上的一个函数 π_1，它为每个 $\eta_1 \in \mathcal{I}_1$ 产生一个行动 $u \in U(\eta_1)$，$U(\eta_1)$ 是可以从 *I*-状态 η_1 导出的行动集合；假设对于 η_1 中所有决策顶点来说该集合都是相同的。相似地，P_2 的一个（确定性的）规划定义为 $\mathcal{I}_2$ 上的一个函数 π_2，它为每个 $\eta_2 \in \mathcal{I}_2$ 产生一个行动 $v \in V(\eta_2)$。

根据 *I*-空间，通常有两种方法来定义随机规划。第一种是定义一个全局随机规划，它是所有确定性的规划集合上的概率分布。在执行过程中，这意味着整个确定性的规划将根

据概率分布提前被采样。另外一种情况是当在每个 I-状态需要行动时对行动进行采样。定义如下所述。对于随机情况，令 $W(\eta_1)$ 和 $Z(\eta_2)$ 分别表示 $U(\eta_1)$ 和 $V(\eta_2)$ 上所有概率分布的集合。P_1 的局部随机规划定义为一个函数，对于每个 $\eta_1 \in \mathcal{I}_1$，函数产生某 $w \in W(\eta_1)$。同样地，P_2 的局部随机规划定义为一个从 $\mathcal{I}_2$ 映射到 $Z(\eta_2)$ 的函数。表示为 I-状态的函数的局部随机规划在对策论文献中通常被称为行为策略。

对于所有序贯对策来说并不存在局部随机规划空间上的一个随机鞍点[60]。这是令人遗憾的，因为对于规划执行过程中进行决策的方式来说，这种形式的随机化似乎是最自然的。至少对于逐阶段模型来说，随机鞍点在局部随机规划空间上总是存在的。对于开环模型来说，只有用全局随机规划才能保证随机鞍点存在（实际上在第 10.5.1 节就是这样做的）。为了有助于问题的理解，假设对策树是平衡的 k 阶段二叉树（有 $2k$ 层）。对于每个局中人来说，有 2^k 种可能的确定性的规划。这意味着可以独立地分配 2^k-1 个概率值（最后一个概率使得概率和为 1）来定义确定性规划空间上的全局随机规划。定义一个局部随机规划，每个局中人有 k 个 I-状态，一个阶段对应一个 I-状态。在每个阶段，概率分布定义在行动集合上，行动集合只有两个元素。因此这些分布仅有一个独立参数。以这种方式用 $k-1$ 个独立参数来指定随机规划。因为 $k-1$ 比 2^k-1 少得多，所以有许多全局随机规划不能表示为一个局部随机规划。非常遗憾，在某些对策中，局部随机表示有可能删除了随机鞍点。

发生这种结果的主要原因是局中人忘记了信息。具有完美回忆功能的局中人能够记住它自己的行动，也从来不会忘记它以前所知道的任何信息。Kuhn 指出所有全局随机规划的空间等价于所有局部随机规划的空间，当且仅当局中人有完美记忆时[565]。因此假设对策中所有局中人都有完美记忆，那么随机鞍点在局部随机规划空间中总是存在的。对于关于局部随机规划空间上随机 Nash 平衡点的存在性这种更一般的情况，Kuhn 的结果也成立。

在涉及更多局中人的对策树中可以采用非确定的 I-状态。从而可以定义确定性的、全局随机化的和局部随机化的规划。Kuhn 的结果应用于任意数量的局中人，这可以确保局部随机策略空间上的随机 Nash 平衡点的存在性，当（且仅当）局中人有完美回忆。通常可以利用这个事实将对策树分解为更小的矩阵对策，如图 10.5.1 所示。能够进行这种操作的条件是它必须是嵌套的[60]。这意味着除了根顶点以外还有多个决策顶点，在这些顶点上：（1）进行决策的局中人知道它处在该顶点（非确定的 I-状态是一个单元素集合）；（2）非确定的 I-状态将不离开以该顶点为根的子树（当绘制对策树时，不能包围子树以外的顶点）。在这种情况下，对策树在这些特殊的决策顶点处可以进行分解，并用对策值代替。非常遗憾，仍然存在多个 Nash 平衡点的问题。

出乎预料的是，非确定的 I-状态的定义没有用历史 I-空间来产生。有可能定义一个感测模型来产生本节使用的非确定的 I-状态。在许多情况下，I-状态可以只用行动历史的一个子集来表达。令 $\tilde{u}_k$ 和 $\tilde{v}_k$ 分别表示 P_1 和 P_2 的行动历史。交替对策模型在第 k 阶段的历史 I-状态对 P_1 来说就是 $(\tilde{u}_{k-1}, \tilde{v}_{k-1})$，对 P_2 来说就是 $(\tilde{u}_k, \tilde{v}_{k-1})$。逐阶段对策模型的历史 I-状态对于两个局中人来说都是 $(\tilde{u}_{k-1}, \tilde{v}_{k-1})$。本节使用的非确定的 I-状态可以从这些历史中产生。对于诸如图 11.31 那样的其他模型来说，还需要感测模型，因为仅有关于某些行动的部分信息出现。这产生了下一节所要介绍的表述，它涉及感测模型和状态空间。

11.7.2 状态空间上对策的信息空间

I-空间的概念可以融入在状态空间上进行的序贯对策中。所产生的表述很自然地将表述 11.1 扩展到多个局中人。这种扩展并不是从两个局中人开始进行一般化推广，而是将有 n 个局中人这种最一般的情形直接呈现在眼前。本节的重点是描述这种对策的 *I*-空间，而不是解决它们。求解的方法很大程度上依赖于特定的信息模型，这里不予介绍。

就像第 11.7.1 节中介绍的那样，每个局中人有其自身的参考坐标系，因此也就有其自身的 *I*-空间。每个局中人的 *I*-状态指示其关于公共对策状态的信息。这与第 10.5 节介绍的状态相同，但是每个局中人可能有不同的观测，还可能不知道其他局中人的行动。因此对于每个决策者来说 *I*-状态是不同的。在完美状态感测的情况下，这些 *I*-空间都收缩为 X。

假设有 n 个局中人。每个局中人有它自身的行动空间 U^i。但是这里不能依赖于 x，因为状态通常可能未知。但是它可以依赖于 *I*-状态。如果大自然行动干扰状态转移方程，那么使用式（11.120）（如果有两个局中人）；否则使用式（11.121），如果给定所有局中人的行动，式（11.121）产生可预测的未来状态。当存在预测不确定性时，可以用一个大自然行动 $\theta \in \Theta(x,u^1,u^2,\cdots,u^n)$ 来建模大自然对所有局中人的影响。

第 11.1.1 节中的任何传感器模型都可以在有多个局中人的情况下进行定义。每个局中人有它自己的观测空间 Y^i 和传感器映射 h^i。对于每个局中人来说，大自然可能通过大自然感测行动 $\Psi^i(x)$ 来干扰观测。状态-行动传感器映射以 $y^i = h^i(x,\psi^i)$ 的形式出现，也可以定义状态传感器映射和基于历史的传感器映射。

现在考虑单个局中人的第 k 阶段，对策是如何出现的。为进行决策，什么信息是可用的？在最一般的情况下，每个局中人产生：（1）一个初始条件 η_0^i；（2）一个行动历史 $\tilde{u}_{k-1}^i$；（3）一个观测历史 $\tilde{y}_k^i$。还必须指明一个局中人是否知道其他局中人以前所采取的行动。对于一个局中人来说，它可能得到其他局中人的观测。如果 P_i 得到所有这种信息，那么它在第 k 阶段的历史 *I*-状态是：

$$\eta_k^i = \left(\eta_0^i, \tilde{u}_{k-1}^1, \tilde{u}_{k-1}^2, \cdots, \tilde{u}_{k-1}^n, \tilde{y}_k^1, \tilde{y}_k^2, \cdots, \tilde{y}_k^n\right) \tag{11.84}$$

但是在多数情况下，η_k^i 只包括式（11.84）中历史的一个子集。一个典型的情况是：

$$\eta_k^i = (\eta_0^i, \tilde{u}_{k-1}^i, \tilde{y}_k^i) \tag{11.85}$$

这意味着 P_i 仅知道它自己的行动和观测。另外一种可能性是所有局中人知道所有已经应用过的行动，但是它们得不到其他局中人的观测。这产生：

$$\eta_k^i = (\eta_0^i, \tilde{u}_{k-1}^1, \tilde{u}_{k-1}^2, \cdots, \tilde{u}_{k-1}^n, \tilde{y}_k^i) \tag{11.86}$$

当然，对本章的一些例子进行一般化就可以定义一些特殊情况。例如可以定义一种无传感器的对策，其中历史 *I*-状态仅包括行动。这产生：

$$\eta_k^i = (\eta_0^i, \tilde{u}_{k-1}^1, \tilde{u}_{k-1}^2, \cdots, \tilde{u}_{k-1}^n) \tag{11.87}$$

或者一种更加隐秘的对策，其中不知道其他局中人的行动：

$$\eta_k^i = (\eta_0^i, \tilde{u}_{k-1}^i) \tag{11.88}$$

一旦确定了 *I*-状态，那么每个局中人的历史 *I*-空间 $\mathcal{I}_{hist}^i$ 定义为所有历史 *I*-状态的集合。一般情况下，可以定义 *I*-映射和衍生 *I*-空间来简化每个历史 *I*-空间。

假设所有空间是有限的，到目前为止所给出的概念可以用来构成序贯对策的表述，这是表述 11.4 在非完美状态信息下的表述。

表述 11.4（用 *I*-空间的序贯对策）

1. n 个局中人 $\mathrm{P}_1,\mathrm{P}_2,\cdots,\mathrm{P}_n$ 的一个集合。

2. 一个有限的非空状态空间 X。

3. 对于每个 P_i，都有一个有限行动空间 U^i。也可以有更一般的定义，其中可行选择的集合依赖于历史 I-状态；记为 $U^i(\eta^i)$。

4. 对于每个 $x\in X$ 和 $u^i\in U^i$，$1\leqslant i\leqslant m$，有一个有限的大自然行动空间 $\Theta=(x,u^1,\cdots,u^n)$。

5. 对于每个 $x\in X$、$\theta\in\Theta(x,u)$ 和 $u^i\in U^i$，$1\leqslant i\leqslant n$，有一个状态转移函数 f，它产生状态 $f(x,u^1,\cdots,u^n,\theta)$。

6. 对于每个 P_i，有一个有限观测空间 Y^i。

7. 对于每个 P_i，有一个有限大自然感测行动空间 $\Psi^i(x)$，其中 $x\in X$。

8. 对于每个 P_i，有一个传感器映射 h^i，它产生观测 $y=h^i(x,\psi^i)$，其中 $x\in X$ 和 $\psi^i\in\Psi^i(x)$。该定义假设状态-大自然感测行动映射。如第 11.1.1 节中所定义的，也可能采用状态传感器映射或者基于历史的传感器映射。

9. 用 k 表示的 K 阶段集合，它从 $k=1$ 开始，结束于 $k=K$。令 $F=K+1$。

10. 对于每个 P_i，都有一个初始条件 η_0^i，它是初始条件空间 $\mathcal{I}_0^i$ 的一个元素。

11. 对于每个 P_i，都有一个历史 I-空间 $\mathcal{I}_{hist}^i$，它是所有历史 I-状态的集合，由行动和观测历史形成，也可能包括其他局中人的历史。

12. 对于每个 P_i，令 L^i 表示逐阶段相加的代价泛函：

$$L^i(\tilde{x}_F,\tilde{u}_K^1,\cdots,\tilde{u}_K^2)=\sum_{k=1}^{K}l(x_k,u_k^1,\cdots,u_k^n)+l_F(x_F) \tag{11.89}$$

对于一个或者多个空间是连续的这种情况，也存在扩展形式，参见文献[60]。也很容易增加目标集合和终止条件、允许阶段无限期地运行。

当所有局中人都有相同的代价函数时会产生表述 11.4 的一个有趣的特例。这并不等价于只有一个局中人的情况，因为这里的多个局中人有不同的 I-状态。例如对多个机器人来说，它们的任务可能是搜索宝藏，但是它们之间的通信很有限。这导致了不同的 I-状态。它们都愿意进行合作，但是由于不知道状态，它们不能进行合作。这种问题演变成队论（team theory）[227,453,533]。

至于表述 10.4 所考虑的对策，每个局中人都有它自己的规划。因为局中人不一定知道状态，那么决策是基于 I-状态。确定性规划、全局随机规划以及局部随机规划的定义本质上与第 11.7.1 节中的相同。仅有的不同之处在于目前定义的是更加一般的 I-空间。为表述 11.4 中的一般对策，可以定义诸如鞍点和 Nash 平衡点等各种解的概念。一般情况下，局部随机鞍点和 Nash 平衡点的存在性依赖于特定的信息模型[60]。

例 11.27（战舰游戏） 经典棋盘游戏中有许多有趣的 I-空间。这里简要阐述一个从战舰游戏而来的对策，它是交替对策模型下的一个序贯对策。两个局中人 P_1 和 P_2 都有一个舰

队，它们秘密地排列在10×10的栅格上（见图 11.32）。

状态是每个局中人的栅格上的所有战舰的精确位置。状态空间为这两个局中人产生了战舰所有可能的位置的集合。每个局中人知道它自己战舰的位置。一旦它们被放置在栅格上，那么不允许它们移动。

局中人轮流猜测用行、列表示的单个格片，怀疑该格片上有战舰。可能的观测是“击中”和“未击中”，这取决于该位置上是否有舰艇。局中人每次只进行一次猜测，轮流进行直到其中一个局中人已经观测到每个被战舰所占据的格片都被击中了一次。

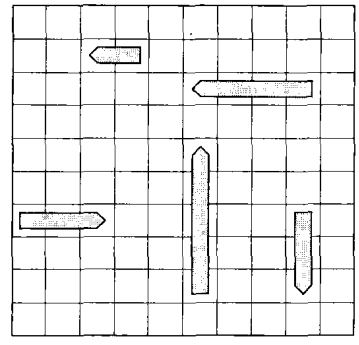


图 11.32 在战舰游戏中，每个局中人将船放置在一个栅格中，其他局中人必须通过询问一个格片是否被占据来猜测战舰的位置

这是一个有趣的游戏，因为一旦发现一个“击中”，那么一个局中人应该搜索附近的其他的“击中”，因为将有数个邻近的格片被同一个战舰所覆盖。仅有的问题是战舰的精确位置和方向未知。一个好的局中人本质上是使用非确定的 *I*-状态来提高下一次击中的机会。 ■

例 11.28（公主与妖怪） 这是一个经典的无感测的例子。一个公主和一个怪物在二维环境中移动。假设采用一个简单运动模型，例如它们在栅格上单步移动。公主试图不被怪物发现，游戏完全在黑暗中进行。当怪物和公主移动到同一个栅格点上时游戏结束。在游戏中没有任何形式的反馈，但是对于局中人来说可以构造非确定的 *I*-状态。对于大多数环境来说，不可能保证怪物能够赢，但是在某些环境下是可以保证的。这个例子可以认为是一种特殊类型的追逃对策。涉及 *I*-空间的连续时间追逃对策将在第 12.4 节介绍。 ■

进一步阅读

信息空间的基本概念可以追溯到 Kuhn 的关于对策树的工作[565]。那里的非确定的 *I*-状态被称为一个信息集合。在推广到整个对策论以后，随机控制论也借鉴这个概念（见文献[97，567]）。信息空间的概念在序贯和微分对策论中得到广泛使用[60]。关于对策论中 *I*-空间的阅读材料，见文献[60，762]。在人工智能的文献中，*I*-状态被称为 belief 状态，它在关于 POMDPs 的研究中非常重要，见第 12 章所提到的文献。控制论中的可观测性问题也导致 *I*-空间[194，310，481，913]，其中观测器用来从历史 *I*-状态中重构目前的状态。在机器人学的文献中，它们被称为超状态[398]和知识状态[317]。与 *I*-空间密切相关的概念也以概念等价类的形式[289]和 Donald 的不变信息框架的形式[288]出现。*I*-空间在文献[69，607，608]中是作为表示存在感测不确定性情况下规划的一般方法而提出的。更多关于传感器的内容，参见文献[354]。

大量书籍都详细介绍了卡尔曼滤波，参见文献[228，567，913]。原始参考文献见文献[503]。关于粒子滤波，参见文献[45，295，352，539]。

习　题

1. $\mathcal{I}_{ndet}$ 中的前向投影：

（a）从一个非确定的 I-状态 $X_k(\eta_k)$ 开始、应用一个行动 u_k，通过扩展第 10.1.2 节中的表示来为非确定的一阶段前向投影产生一个表达式。

（b）从 $X_k(\eta_k)$ 开始、应用行动 u_k 和 u_{k+1}，确定一个两阶段前向投影的表达式。

2. $\mathcal{I}_{prob}$ 中的前向投影：

（a）从一个概率 I-状态 $P(x_k|\eta_k)$ 开始、应用行动 u_k，产生概率一阶段前向投影的一个表达式。

（b）从一个概率 I-状态 $P(x_k|\eta_k)$ 开始、应用行动 u_k 和 u_{k+1}，确定两阶段前向投影的一个表达式。

3. 对于一个给定的历史 I-状态 η_k，确定 $\mathcal{I}_{hist}$ 上的强后向投影和弱后向投影。它们应该给出可能的 $\eta_{k-1} \in \mathcal{I}_{hist}$ 的集合。

4. 在第 11.3.2 节曾提到，一个等价 DFA 可以从一个 NFA 来构造。

（a）给出一个明确的 DFA，它能够接受与图 11.8（b）中 NFA 相同的串的集合。

（b）确定图 11.8（b）中的 NFA 是否能够将任意串作为用表述 2.1 所示的规划问题来接受。

5. 这个问题涉及计算例 11.14 的概率 I-状态。令初始 I-状态为：

$$P(x_1) = [1/3 \;\; 1/3 \;\; 1/3] \tag{11.90}$$

其中向量的第 i 项指示 $P(x_1 = i+1)$。令 $U = \{0,1\}$。对于每个行动，可以确定一个状态转移矩阵，它给出概率 $P(x_{k+1}|x_k,u_k)$。对于 $u=0$，令 $P(x_{k+1}|x_k,u_k=0)$ 为：

$$\begin{pmatrix} 4/5 & 1/5 & 0 \\ 1/10 & 4/5 & 1/10 \\ 0 & 1/5 & 4/5 \end{pmatrix} \tag{11.91}$$

第 i 行的第 j 个元素产生 $P(x_{k+1} = i|x_k = j, u_k = 0)$。对于 $u=1$，令 $P(x_{k+1}|x_k,u_k=1)$ 为：

$$\begin{pmatrix} 1/10 & 5/5 & 1/10 \\ 0 & 1/5 & 4/5 \\ 0 & 0 & 1 \end{pmatrix} \tag{11.92}$$

感测模型由三个向量来给定：

$$P(y_k|x_k = 0) = [4/5 \;\; 1/5] \tag{11.93}$$

$$P(y_k|x_k = 1) = [1/2 \;\; 1/2] \tag{11.94}$$

$$P(y_k|x_k = 2) = [1/5 \;\; 4/5] \tag{11.95}$$

其中第 i 个分量产生 $P(y_k = i|x_k)$。假设 $k=3$ 和到目前所得到的历史 I-状态为：

$$(\eta_0, u_1, u_2, y_1, y_2, y_3) = (\eta_0, 1, 0, 1, 0, 0) \tag{11.96}$$

任务是计算概率 I-状态。从 $P(x_1)$ 开始，计算如下分布：$P(x_1|\eta_1)$、$P(x_2|\eta_1,u_1)$、$P(x_2|\eta_2)$、$P(x_3|\eta_2,u_2)$、$P(x_3|\eta_3)$。

6. 解释例 11.7 中为什么从一个非确定的 I-状态不可能到达每个非确定的 I-状态。给出一个从初始 I-状态不能到达的一个非确定的 I-状态的例子。完整刻画从所有可能的初始条件开始的非确定的 I-状态的可达性。

7. 采用例 11.12 中的思路，考虑有一个点在如图 11.33 所示的拓扑图上运动。完全刻画 $\mathcal{I}_{ndet}$ 的连通性（可用对称性来简化答案）。

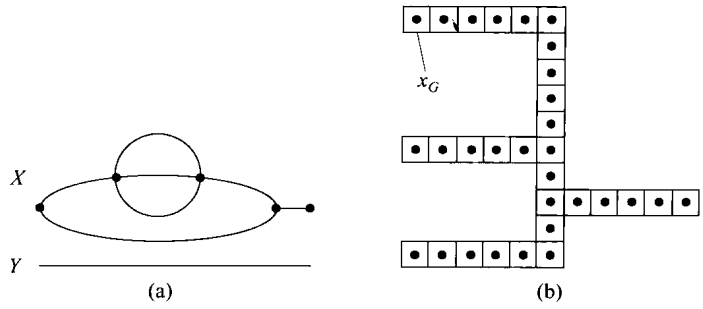


图 11.33 （a）一个拓扑图，图中有一个点在运动（注意到有两对顶点是垂直排列的）；（b）一个习题，它是例 11.17 的一种变形

8. 为例 11.17 设计一个 I-映射，它不必是充分的，但是应该导致只定义在三个衍生 I-状态上的一个规划解。

9. 考虑图 11.33（b）中的离散问题，使用例 11.17 中的感测和运动模型。

（a）设计一个充分 I-映射和一个规划解，它们用到尽可能少的衍生 I-状态。

（b）设计一个 I-映射和一个规划解，I-映射不一定是充分的，规划解用尽可能少的衍生 I-状态。

10. 假设有两个 I-映射 $\kappa_1 : \mathcal{I}_1 \to \mathcal{I}_2$ 和 $\kappa_2 : \mathcal{I}_2 \to \mathcal{I}_3$，并且给定 κ_1 关于 $\mathcal{I}_1$ 是充分的，κ_2 关于 $\mathcal{I}_2$ 是充分的。确定映射 $\kappa_1 \circ \kappa_2$ 关于 $\mathcal{I}_1$ 是否是充分的，证明你的结论。

11. 试用较少的非确定 I-状态给出例 11.16 的解。

12. 假设一个点机器人在 $\mathbb{R}^2$ 中运动，它接收三个非共线的导引信标的观察信息，信标的信息来源于已知位置。假设机器人能够标定 $\mathbb{S}^1$ 上的三个观测。

（a）证明机器人总能够恢复它在 $\mathbb{R}^2$ 中的位置。

（b）如果仅有两个信标，那么机器人能够进行什么推断？

13. 非确定的 I-状态问题：

（a）证明例 11.23 中的非确定的 I-状态总是一个单连通区域，该区域的边界仅由圆弧和线段组成。

（b）为有效地计算从一个阶段到另外一个阶段的非确定的 I-状态而设计一个算法。

14. 设计一个算法，它将一个单连通直线区域（例如一个都是直角的多边形所描述的区域）和一个目标状态作为输入，并且设计一个倾斜托盘的序列来保证球将停留在目标上。例 11.24 提供了这样一个阐述。

15. 将第 11.7.2 节的历史 I-空间的对策论表述扩展到连续时间情况。

16. 考虑“我从哪里来？”这样的问题。

（a）推导 $X_1(\eta_k)$ 的表达式。

（b）推导 $P(x_1 | \eta_k)$ 的表达式。

17. 在例 11.27 的对策中，是否存在一个点，在该点上一个局中人虽然没有观察到每个可能的“击中”，但它知道对策的状态（例如所有战舰的精确位置）？请给予解释。

18. 当在娱乐场玩扑克游戏时，一些数牌策略涉及对牌进行简单统计，而不是得到目前为止所见到的所有的牌。定义一个数牌游戏，将它作为能够大量减少 I-空间大小的例子和信息反馈规划的例子。

实现

19. 实现一个在平面中运动的机器人这种情况的卡尔曼滤波器。显示在执行过程中得到的置信椭圆体。注意数值计算问题（见文献[567]）。

20. 实现在一个二维多边形环境中运动的点机器人的概率 I-状态的计算。比较基于栅格的近似方法和粒子滤波的效率和精度。

21. 设计和实现一个算法，它用非确定的 I-状态来进行战舰对策，如例 11.27 所示。

第 12 章　存在感测不确定性条件下的规划

第 11 章的主要目的是介绍信息空间（*I*-空间）的概念和给出一些有助于理解的例子。本章将讨论存在感测不确定性时的规划，也就是在 *I*-空间中进行的规划。第 12.1 节将介绍一般算法，从中可以发现只有那些状态较少的问题才能得以解决，因为 *I*-空间呈爆炸式增长。在第 6 章已经看到一般运动规划算法只能够应用于简单问题。克服这种情况的方法是研究基于采样的技术，或者将问题集中在更加狭小的范围内。将基于采样的规划思路应用于 *I*-空间是非常令人感兴趣的，但是目前该思路还需要进一步研究和探讨。因此本章将注意力主要集中在为更加狭小范围内的问题设计规划算法。在每种情况下，研究的算法所能够解决的问题比用一般算法所能够解决的问题要复杂得多。这是因为它们利用了问题的某些特殊结构。

当处理 *I*-空间时，一个重要的原理是通过得到更简单的衍生 *I*-空间来研究尽可能减小空间大小和复杂性的 *I*-映射。由于 *I*-空间具有完全一般性，那么按照这个思路可能设计一个有效解决新问题的专用算法。该思路在本章不断出现。由第 11.2.2 节可知，最常见的衍生 *I*-空间是 $\mathcal{I}_{ndet}$，第 11.2.3 节中的 $\mathcal{I}_{prob}$ 也是经常出现的。

第 12.1 节以后，本章其余部分所考虑的问题主要是受机器人应用领域的启发。第 12.2 节将讨论定位问题，即机器人必须用感测到的信息来确定它的位置。从本质上讲，这是一个维持衍生 *I*-状态以及计算能够导致想要的衍生 *I*-空间的规划问题。第 12.3 节将定位问题推广到机器人不知道它所处的环境。在这种情况下，状态空间和 *I*-空间既考虑机器人可能所处的环境，也考虑在环境中机器人可能所处的位置。该节是机器人学的基础，因为建立机器人所处环境的精确地图是非常困难和昂贵的。通过对 *I*-空间进行仔细研究，在许多应用中可以避免对环境进行完备表示。

第 12.4 节将介绍一种追逃对策，该对策可以被认为是儿童“藏猫猫”游戏的正式版本。追赶者打着灯笼，它必须照亮一个能够以不受限的速度运动的不可预测的逃跑者。该问题的非确定的 *I*-状态描述了逃跑者的可能的位置的集合。这个问题可以通过对 $\mathcal{I}_{ndet}$ 进行胞腔剖分得到一个有限的、图搜索问题来解决。方法是基于在 *I*-空间中寻找临界曲线，这就像第 6.3.4 节中针对线段机器人的临界曲线方法。

第 12.5 节将论述在不完美状态信息下的操作规划。这不同于第 7.3.2 节中的操作规划，因为那里假设状态总是已知的。第 12.5.1 节将提出原像规划的框架，20 年前引入原像规划是为了讨论状态转移和传感器都存在有界不确定性下的操作规划问题。从该框架以及相关的文献中可以得到一些重要 *I*-空间的思路及其复杂性，因此这里予以介绍。第 12.5.2 节讨论的问题中机器人拥有非常有限的感测信息，它依赖于在与目标对象之间的相互影响中所得到的信息。在某些情况下，这些方法甚至不需要进行感测。

12.1　一 般 方 法

本节对第 11.1 节所介绍的问题提出规划方法。它们主要基于基本动态规划，而不需要利用问题的任何特定结构。因此它们的应用局限于小的状态空间；尽管如此，它们还是值得介绍的，因为它们非常具有一般性。基本的思路是用非确定的或者概率的 I-映射来表达问题，这种表达分别采用衍生的 I-空间 $\mathcal{I}_{ndet}$ 或 $\mathcal{I}_{prob}$。一旦定义了衍生的信息转移方程，那么可以假设 $\mathcal{I}_{ndet}$ 或 $\mathcal{I}_{prob}$ 就是一个状态空间，在执行过程中能够得到完美的状态信息测量（因为 I-状态总是已知的）。

12.1.1　将信息空间作为一个大的状态空间

表述 11.1 所指定的任何问题都可以用衍生 I-状态来转换为表述 10.1 所描述的问题。通过建立在第 11.1.3 节结尾所讨论内容的基础之上，这可以通过将 I-空间当作一个大的状态空间来实现，在这个大状态空间中每个状态都是原始问题的一个 I-状态。以前给出过某些组成部分，但是这里仍然给出一个完整表述。

假设问题已经由表述 11.1 给出，那么通常的组成部分有 X、U、Θ、f、Y、h、x_I、X_G 和 L。在任何充分的 I-映射中将会用到如下概念，但是叙述只限于两种重要情况：κ_{ndet} 和 κ_{prob}（它们分别产生衍生 I-空间 $\mathcal{I}_{ndet}$ 和 $\mathcal{I}_{prob}$）。

表述 10.1 的组成仍采用原始问题的组成。为了避免在两个表述之间产生混淆，新的表述的每个组成部分上被加上了一个箭头。图 12.1 简要介绍了即将进行的定义。新的状态 $\vec{X}$ 被定义为 $\vec{X}=\mathcal{I}_{der}$，状态 $\vec{x}\in\vec{X}$ 是一个衍生 I-状态 $\vec{x}=\eta_{der}$。在非确定的不确定性下，$\vec{x}_k$ 就是 $X_k(\eta_k)$，其中 η_k 是历史 I-状态。在概率不确定性下，$\vec{x}_k$ 就是 $P(x_k|\eta_k)$。行动空间依然保持不变：$\vec{U}=U$。

	符号	说明
状态	$\vec{x}=\eta_{der}$	衍生 I-状态
状态空间	$\vec{X}=\mathcal{I}_{der}$	衍生 I-空间
行动空间	$\vec{U}=U$	原始行动空间
大自然行动空间	$\vec{\Theta}\subseteq Y$	原始观测空间
状态转移方程	$\vec{f}(\vec{x},\vec{u},\vec{\theta})$	大自然的行动只是 y
初始状态	$\vec{x}_I=\eta_0$	初始 I-状态 $\eta_0\in\mathcal{I}_{der}$
目标集	$\vec{X}_G$	原始 X_G 的子集
代价泛函	$\vec{L}$	从原始 L 衍生而来

图 12.1　衍生 I-空间可以被认为是一个平常的状态空间，在该空间上可以进行有完美状态信息的规划

表述中最奇特的部分是新的大自然行动空间 $\vec{\Theta}(\vec{x},\vec{u})$。表述 11.1 中的观测很像大自然的行动，因为它们都不是机器人选择的，并且稍后将会看到，它们只是新的状态转移方程的

不可预测的部分。因此 $\vec{\Theta}(\vec{x},\vec{u})\subseteq Y$，$Y$ 是原始观测空间。一个新的大自然行动 $\vec{\theta}\in\vec{\Theta}$ 只是一个观测 $\vec{\theta}(\vec{x},\vec{u})=y$。集合 $\vec{\Theta}(\vec{x},\vec{u})$ 通常依赖于 $\vec{x}$ 和 $\vec{u}$，因为某些观测不可能从某些状态得到。例如，如果测量可移动机器人位置的一个传感器的测量误差从来不会超过 1m，那么距离真实位置超过 1m 的观测是不可能的。

一个衍生状态转移方程用 $\vec{f}(\vec{x}_k,\vec{u}_k,\vec{\theta}_k)$ 定义，它产生了一个新的状态 $\vec{x}_{k+1}$。采用原始符号，这只是一个用 $\kappa(\eta_k)$、u_k 和 y_k 计算下一个衍生 I-状态 $\kappa(\eta_{k+1})$ 的函数，这是允许的，因为我们有第 11.2.1 节所描述的充分 I-映射。

初始状态和目标集合是可选的，它们可以用新的表示方法来描述。初始 I-状态 η_0 变成新的初始状态 $\vec{x}_I=\eta_0$。根据规划是在 $\mathcal{I}_{ndet}$ 还是在 $\mathcal{I}_{prob}$ 中进行的，可以假设 η_0 是 X 的一个子集或一个概率分布。在非确定的情况下，新的目标集合 $\vec{X}_G$ 为：

$$\vec{X}_G=\left\{X(\eta)\in\mathcal{I}_{ndet}\,\middle|\,X(\eta)\subseteq X_G\right\} \tag{12.1}$$

它是衍生 I-状态的集合，这些状态保证真实状态落入 X_G。通过要求赋有非零概率 $P(x|\eta)$ 的所有状态落入 X_G，还可以得到它的概率版本。如果不用非零概率，那么也可以用门限值。例如可以要求到达目标的可能性为 98%。

表述 10.1 还包括代价泛函。我们将研究只使用状态和行动历史的代价模型。对大自然的依赖性意味着代价直接依赖于观测 $y=\vec{\theta}$，在表述 11.1 中并未做这种假设。表述 10.1 中 K 阶段代价泛函在这里被表示为：

$$\vec{L}(\vec{x}_k,\vec{u}_k)=\sum_{k=1}^{K}\vec{l}(\vec{x}_k,\vec{u}_k)+\vec{l}_F(\vec{x}_F) \tag{12.2}$$

关于终止行动，它有通常意义下的代价假设。

代价泛函 $\vec{L}$ 必须从原始问题的代价泛函 L 中产生。这用未知状态来表示。首先考虑 $\mathcal{I}_{prob}$ 的情况。第 k 阶段的状态 x_k 服从第 11.2.3 节的概率分布 $P(x_k|\eta_k)$。使用 $\vec{x}_k$，那么期望代价为：

$$\vec{l}(\vec{x}_k,\vec{u}_k)=\vec{l}(\vec{\eta}_k,\vec{u}_k)=\sum_{x_k\in X}P(x_k|\eta_k)l(x_k,u_k) \tag{12.3}$$

$$\vec{l}_F(\vec{x}_F)=\vec{l}_F(\vec{\eta}_F)=\sum_{x_F\in X}P(x_F|\eta_K)l_F(x_F) \tag{12.4}$$

理想情况下，对于 $\mathcal{I}_{ndet}$ 的情况，我们愿意构造相似的表达，但是存在一个问题。为每个阶段都表述最坏情况代价是过于悲观的。例如在相继两个阶段之间可能得到非常高的代价，但是这两个代价分别对应于 X 中如下不同的路径。没有将最坏情况分析约束到相同路径。在概率情况下没有这种问题，因为可以给路径分配概率。对于非确定的情况，可以定义一个代价泛函，但是在一般情况下破坏了动态规划所需要的逐阶段相加的性质。如果对可允许的代价进行某些限制，那么逐阶段相加的性质将得到保护。

知道第 k 阶段的状态 x_k 落在 $X_k(\eta_k)$ 中，如第 11.2.2 节所述。对于每个历史 I-状态 $\eta_k=\vec{x}_k$ 和 $u_k\in U$，假设 $l(x_k,u_k)$ 在所有 $x_k\in X_k(\eta_k)$ 上是不变的。在这种情况下：

$$\vec{l}(\vec{x}_k,\vec{u}_k)=\vec{l}(\vec{\eta}_k,\vec{u}_k)=l(x_k,u_k) \tag{12.5}$$

其中 $x_k\in X_k(\eta_k)$，并且：

$$\vec{l}_F(\vec{x}_F)=\vec{l}_F(\vec{\eta}_F)=l_F(x_F) \tag{12.6}$$

其中 $x_F \in X_F(\eta_F)$。

现在也可以将衍生 I-空间 $\mathcal{I}_{ndet}$ 或者 $\mathcal{I}_{prob}$ 上的规划认为是新的状态空间 $\vec{X}$ 上的规划。因此现在有可能进行状态反馈，但反馈是在更大的状态空间 $\vec{X}$ 上进行的，而不是在 X 上。由于有观测，行动的结果通常还是不可预测的。当没有观测时会发生一种有趣的特殊情况。在这种情况下，I-状态是可预测的，因为它只从机器人所选择的行动中产生。这样，新的表述不需要大自然行动，那么它将缩减成表述 2.3。因此，如果给定初始条件，那么就不再需要反馈。规划仍可以用一个行动序列来表达。即使原始状态不是可预测的，未来信息状态却是可预测的。这意味着在新的状态空间中的状态轨迹完全是可预测的。

12.1.2　非确定的 I-空间中的算法

既然现在 $\mathcal{I}_{ndet}$ 中的规划问题已经用表述 10.1 进行了表达，那么可以直接应用第 10.2 节中的方法。但主要局限性在于新的状态空间 $\vec{X}$ 比 X 大指数倍。如果 X 包括 n 个状态，那么 $\vec{X}$ 包括 2^n-1 个状态。因此即使第 10.2 节的某些方法可以解决涉及上百万个状态的问题，但是在原始状态空间中只有大约 20 个状态。处理巨大数量的问题要求研究针对特定应用的方法来利用其 I-空间的某些特殊结构，可能采取的方法是定义一个产生较小衍生 I-空间的 I-映射。

值迭代　第 10.2.1 节的值迭代方法不经过修改就可以直接应用。在第一步中，用式（12.6）初始化 G_F^*。使用新问题的符号，那么动态规划递归式（10.39）变为：

$$G_k^*(\vec{x}_k) = \min_{\vec{u}_k \in U} \left\{ \max_{\vec{\theta}_k} \left\{ \vec{l}(\vec{x}_k, \vec{u}_k) + G_{k+1}^*(\vec{x}_{k+1}) \right\} \right\} \tag{12.7}$$

其中 $\vec{x}_{k+1} = \vec{f}(\vec{x}_k, \vec{u}_k, \vec{\theta}_k)$。

计算式（12.7）的困难在于确定产生最大值的集合 $\vec{\Theta}(\vec{x}_k, \vec{u}_k)$。假设采用状态-行动传感器映射，如第 11.1.1 节所定义的。从 I-状态 $\vec{x}_k = X_k(\eta_k)$，应用行动 $\vec{u}_k = u_k$。这就产生了前向投影 $X_{k+1}(\eta_k, u_k)$。所有可能的观测的集合为：

$$\vec{\Theta}(\vec{x}_k, \vec{u}_k) = \left\{ y_{k+1} \in Y \middle| \exists x_{k+1} \in X_{k+1}(\eta_k, u_k), \exists \psi_{k+1} \in \Psi, y_{k+1} = h(x_{k+1}, \psi_{k+1}) \right\} \tag{12.8}$$

不用前向投影，可以得到一个更长的等价表达式：

$$\vec{\Theta}(\vec{x}_k, \vec{u}_k) = \{ y_{k+1} \in Y | \exists x_k \in X_k(\eta_k), \exists \theta_k \in \Theta, \exists \psi_{k+1} \in \Psi, y_{k+1} = h(f(x_k, u_k, \theta_k), \psi_{k+1}) \} \tag{12.9}$$

不同的感测模型也可以有其他表述形式。

策略迭代　可以应用第 10.2.2 节的策略迭代方法，但是它不可能解决具有挑战性的问题。例如，如果 $|X| = 10$，那么每次迭代将要求解决有一百万个元素的矩阵。在一些应用中，至少它们可能是稀疏的。

图搜索方法　也可以将第 10.2.3 节的基于后向投影的方法应用于该表述。这些方法必须初始设置 $S = \vec{X}_G$。如果 S 初始时非空，那么可以使用图 10.6 中的一般算法来尝试后向投影。可以用图 10.8 给出的 Dijkstra 算法来产生最坏情况下最优的规划。

无传感器的情况　如果没有传感器，那么可以应用更好的方法，因为描述已经从表述 10.1 缩减为表述 2.3。可以用第 2.3 节的更简单的值迭代或者 Dijkstra 算法来求解。如果不

要求最优性，那么可以用第 2.2 节的任何搜索方法。例如可以假设在 $\vec{X}$ 上进行一个双向搜索来将 $\vec{x}_I$ 连接到 $\vec{x}_G$。

12.1.3 概率 I-空间中的算法（POMDPs）

对于概率情况，不能应用第 10.2 节中的方法，因为 $\mathcal{I}_{prob}$ 是一个连续空间。需要用第 10.6 节介绍的连续空间的动态规划方法。主要的困难在于 $\vec{X}$ 的维数随 X 中状态数量线性增长。如果在 X 中有 n 个状态，那么 $\vec{X}$ 的维数为 $n-1$。因为第 10.6 节的方法受维数所限，一般动态规划技术局限于 X 只有较少状态的问题。

近似值迭代 通过在 $\vec{X}$ 上进行插值来确定尚需代价值，可以直接用第 10.6 节的连续空间方法来产生近似解。采样点集合上初始的尚需代价值 G_F^* 可用式（12.6）得到。根据式（10.46），动态规划递归是：

$$G_k^*(\vec{x}_k) = \min_{\vec{u}_k \in \vec{U}} \left\{ \vec{l}(\vec{x}_k, \vec{u}_k) + \sum_{\vec{x}_{k+1} \in \vec{X}} G_{k+1}^*(\vec{x}_{k+1}) P(\vec{x}_{k+1} | \vec{x}_k, \vec{u}_k) \right\} \tag{12.10}$$

如果 $\vec{\Theta}(\vec{x},\vec{u})$ 有界，那么概率质量分布在点 $y = \vec{\theta} \in \vec{\Theta}(\vec{x},\vec{u})$ 的一个有限集合上。这意味着 $P(\vec{x}_{k+1}|\vec{x}_k,\vec{u}_k)$ 也分布在 $\vec{X}$ 的一个有限子集上。这似乎有些不寻常，因为 $\vec{X}$ 是连续空间，它通常要求概率密度函数。因为将来状态的集合是有限的，这使得在式（12.10）中可以用和式来代替概率密度函数上的积分。这种技术产生 $\vec{X}$ 上的概率密度，但是该密度必须用 Dirac 函数来表达[1]。但是仍然需要进行近似，因为 x_{k+1} 点可能不是表示尚需代价函数 G_{k+1}^* 所用的采样点。

精确方法 如果阶段的总数小，那么有可能计算精确表达式。一些方法是基于观察到已付代价是分段线性的和凸的[497]。这产生了一个线性规划问题，它可以用第 9.3 节中的零和对策的随机鞍点来解决。由于有大量约束，已经有许多方法能够大大减少所需要考虑的数量（见本章结尾关于 POMDPs 的参考文献）。

一个精确的离散表示可以像下面那样进行计算。假设初始条件空间 $\mathcal{I}_0$ 由一个初始条件 η_0 组成（或者一个有限数量的初始条件），并且决策不超过 K 个阶段。因为假设 $\Theta(x,u)$ 和 $\Psi(x)$ 是有限的，那么可能有有限个最终 I-状态 $\eta_F = (\eta_0, \tilde{u}_K, \tilde{y}_F)$。对于其中每个最终状态，可以计算分布 $P(x_F|\eta_F)$，这也可以表示为 $\vec{x}_F$。那么下面用式（12.4）为每个可能的 $\vec{x}_F$ 计算 $G^*(\vec{x}_F)$。遗憾的是这些状态的数量是总阶段数的指数形式。对 $k = K$ 应用动态规划递归式（12.10）来后退一个阶段。每个可能的 $\vec{x}_{k+1}$ 将是 $\vec{X}$ 中的一个点，在该点上计算代价值。因为要为所有可能的 I-状态计算值，所以不需要进行插值。可以重复使用式（12.10）直到到达第一阶段。在每次迭代中，不需要进行插值，因为为每个可能的下一个 I-状态可计算出所需代价 G_{k+1}^*。给定数量巨大的 $\mathcal{I}$，该方法实际上只能解决很小的问题。

无传感器的情况 在没有观测的情况下，$\mathcal{I}_{prob}$ 中的路径变成可预测的。假设表述的是一个可行规划问题。例如在规划执行过程中对 X 上的概率分布有许多复杂的约束。因为 $\vec{X} = \mathcal{I}_{prob}$ 是一个连续空间，那么可以用第 5 章的运动规划技术来寻找一条成功的路径。这

1 这是赋予了非零概率质量的单点，在连续概率密度函数的构造中这是不被允许的。

些技术可能需要一些适应性修改，但是必须用行动和状态转移方程来表述它们，在第 5 章中我们没有这样做。对这些方法所进行的修改将在第 14 章介绍。它们可以应用该问题来搜索 *I*-空间，并且在满足对概率的严格约束的同时产生该空间的行动序列。

12.2 定　　位

定位是机器人学的基本问题。通过传感器，可移动机器人必须在环境地图中确定它的位置。定位分为被动定位和主动定位两种。

- **被动定位**：机器人采取行动，它的位置通过计算非确定的或者概率的 *I*-状态来确定。例如，若采用卡尔曼滤波，那么概率 *I*-状态由均值和方差来确定。均值表示对机器人位置的估计，方差表示不确定性的量。
- **主动定位**：设计的规划必须尽可能减小定位的不确定性。机器人应该如何移动才能定出它的位置呢？

本节将讨论这两种定位。

在许多应用中，定位是一个增量问题。初始位形可能已知，任务是当运动时始终保持良好估计。一种极端情况是被绑架的机器人（kidnapped-robot）问题，其中机器人在开始时不知道它自己的初始位形。每种情况都可以用相应的初始条件来建模。被绑架的机器人问题更加困难，本节在默认情况下都假设这种情况。

12.2.1 离散定位

关于定位问题的许多教程都从离散定位开始。能够解决的或者不能解决的问题可以嵌入更加复杂的、涉及连续状态空间的问题中。离散情况通常更容易理解。为了简化表示，只考虑非确定的 *I*-空间 $\mathcal{I}_{ndet}$，概率情况可参见第 12.2.3 节。

假设一个机器人在二维栅格上移动，这种情况在例 2.1 中介绍过。它有一个栅格图，但是机器人不知道它在栅格中的初始位置或者方向。图 12.2（a）显示了一个例子。

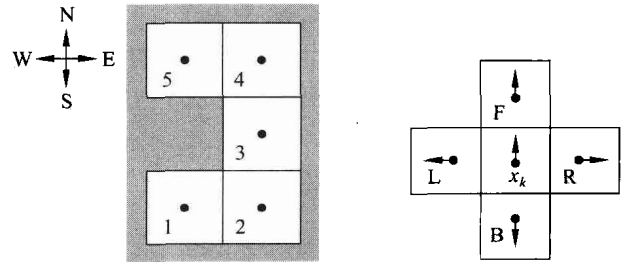


图 12.2　（a）为进行定位，给机器人一个地图；（b）4 个可能的行动，每步进行一个行动，如果可能，机器人将重新定向

为了表述该问题，我们将位置和方向包含在状态中。假设用 N、E、W、S 分别表示“北”、“东”、“西”、“南”4 个方向。尽管机器人被当作一个点，但是它的方向非常重要，因为它没有罗盘。如果它选择向一个特定方向移动，例如一直向前，那么它不必知道前进的方向。

因此状态 $x \in X$ 被记为 $x=(p,d)$，其中 p 表示一个位置，d 表示 4 个方向之一。在相同位置上的状态集合用特殊上标表示，它指向可能的方向。例如 $3^{\llcorner}$ 代表状态集合 $p=3$ 且可能的方向为北（N）和东（E），因为上标指向北向和东向。

给定机器人 4 个行动：

$$U=\{\mathrm{F,B,R,L}\} \tag{12.11}$$

分别表示向“向前”、“向后”、“向右”、“向左”移动。这些运动按照机器人目前的方向进行，方向可能是未知的（见图 12.2（b））。对于 F 行动，机器人向前移动一个栅格并保持它的方向。对于 B 行动，机器人改变方向 180°，然后向前移动一个栅格。对于 R 行动，机器人向右转 90°，然后向前移动一个栅格，L 行动与此相似。如果由于障碍而无法移动，那么假设机器人改变它的方向（B、R 或者 L 情况），但是不改变它的位置，如图 12.3（a）所示。

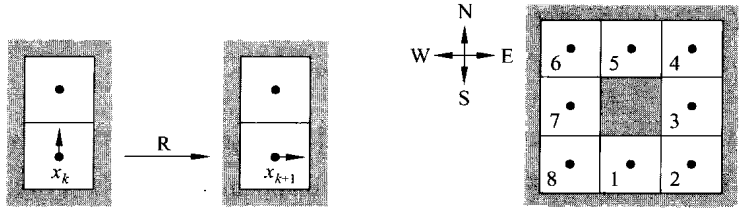


图 12.3 （a）如果一个方向受到障碍物的阻挡，那么改变方向，但是位置不变，在这个例子中，应用行动 R；（b）为定位的目的，给机器人另外一个地图，在这种情况下，机器人不能准确确定自身位置

机器人有一个简单的传感器，只能探测是否能够向它所期望的方向移动。传感器空间是 $Y=\{0,1\}$，传感器映射为 $h: X\times X\to Y$。这就产生了如下情况：如果 x_{k-1} 和 x_k 将机器人放在不同位置，那么 $y=h(x_{k-1},x_k)=1$；否则 $h(x_{k-1},x_k)=0$。因此传感器指示在应用行动以后机器人是否已经移动。

采用非确定的不确定性，并且假设初始 I-状态 η_0 总是 X（它可以扩展到允许从 X 任意非空子集开始）。第 k 阶段的历史 I-状态的一般形式为：

$$\eta_k=(X,\tilde{u}_{k-1},y_2,\cdots,y_k) \tag{12.12}$$

与式（11.14）进行比较会发现，这里进行了一个特殊的调整。没有观测 y_1，因为传感器映射要求以前的状态来报告一个值。因此观测历史从 y_2 开始。第 $k=5$ 阶段的历史 I-状态是：

$$\eta_5=(X,\mathrm{R,R,F,L},1,0,1,1) \tag{12.13}$$

其中 $\eta_0=X$，$\tilde{u}_4=(\mathrm{R,R,F,L})$，$(y_2,y_3,y_4,y_5)=(1,0,1,1)$。

被动定位（passive localization）问题开始于如图 12.2（a）所示的一个给定地图和一个历史 I-状态 η_k，该问题计算非确定的 I-状态 $X_k(\eta_k)\subseteq X$。主动定位（active localization）问题就是计算某 k 和行动序列 $(u_1,\cdots,u_{k-1})$ 使得非确定的 I-状态尽可能小。在最好情况下，$X_k(\eta_k)$ 可能变成一个单元素集合，这意味着机器人知道它在地图中的位置和方向。但是由于存在后面例子会提到的对称性，所以它是不可能的。

解决被动定位问题 被动问题只要求机器人运动时正确计算非确定的 I-状态。下面给

出两个这方面的例子。

例 12.1（一个很容易的定位问题） 考虑图 12.2（a）给出的例子。假设机器人开始时处在位置 1、面向东。初始条件为 $\eta_0 = X$，这可以表示为：

$$\eta_0 = 1^{\rightarrow} \cup 2^{\rightarrow} \cup 3^{\rightarrow} \cup 4^{\rightarrow} \cup 5^{\rightarrow} \tag{12.14}$$

即 X 中所有 20 个状态的集合。假设应用行动序列 $(\mathrm{F,L,F,L})$。这样就产生了观测历史 $(y_2, y_3, y_4, y_5) = (1,1,1,1)$。

应用第一个行动 $u_1 = \mathrm{F}$ 以后，历史 I-状态为 $\eta_2 = (X,\mathrm{F},1)$。非确定的 I-状态为：

$$X_2(\eta_2) = 1^{\leftarrow} \cup 2^{\uparrow} \cup 3^{\rightarrow} \cup 4^{\downarrow} \cup 5^{\leftarrow} \tag{12.15}$$

这意味着任何位置都是可能的，但是成功的向前运动则排除了其他方向的可能性。例如 $1^{\downarrow}$ 是不可能的，因为 1 的正南方是一个障碍物。

应用第二个行动 $u_2 = \mathrm{L}$ 以后：

$$X_3(\eta_3) = 3^{\uparrow} \cup 5^{\leftarrow} \tag{12.16}$$

它只产生两个可能的状态。这可以从图 12.2（a）观察到只有两个状态，从这两个状态出发向前运动，紧接着的是向左运动。初始状态必须是 $1^{\leftarrow}$ 或者 $3^{\uparrow}$。

应用第三个行动 $u_3 = \mathrm{F}$ 以后，剩下的可能性只有 x_3 必须为 $3^{\uparrow}$。这产生：

$$X_4(\eta_4) = 4^{\uparrow} \tag{12.17}$$

这就是机器人的位置：它处在位置 4、面向北。应用最后一个行动 $u_4 = \mathrm{L}$ 后：

$$X_5(\eta_5) = 5^{\leftarrow} \tag{12.18}$$

这意味着在最后状态 x_5 下，机器人在位置 1、面向西。一旦机器人状态已知，那么就不会再积累新的不确定性，因为所有行动的效果都是可以预测的。尽管没有显示出来，但是有可能通过执行一些不产生运动的行动来裁剪一些可能的状态。 ■

例 12.2（涉及对称性的问题） 现在将图 12.2（a）中的地图进行扩展以形成如图 12.2（b）所示的环。在这种情况下，不可能确定机器人的精确位置。为简单起见，只考虑产生运动的行动（一定要相信自己，采取其他行动并不能解决问题）。

假设机器人初始时处在位置 1、面向东。如果执行行动序列 $(\mathrm{F,L,F,L,\cdots})$，那么机器人将围绕一个环运动。问题是也可能从位置 3、面向北的状态应用同样的行动序列。每个行动都成功地移动机器人，即对机器人来说，信息是相同的。可以应用该序列在环中运动的其他两种情况是：（1）从 5 向西；（2）从 7 向南。从 2 向东也可以产生相似的情形，如果应用序列 $(\mathrm{L,F,L,F,\cdots})$。你能够发现使得该序列在每个阶段都能移动机器人的其他 3 种开始状态吗？当按照顺时针方向运动、用向右代替向左时也存在与此相似的对称性。

该问题的状态空间包括 32 个状态，每个位置有 4 个方向。进行某些运动后，非确定的 I-状态可以减少到不超过 4 个可能状态的一个对称类型。如何证明呢？一种方法是使用下面所描述的算法。 ■

解决主动定位问题 从前面的两个例子应该可以清楚地看到如何计算非确定的 I-状态以及如何解决栅格上被动定位问题。现在考虑构造一个规划来解决主动定位问题。假设用计算机来帮助完成该任务。通常有两种方法。

- **预计算规划**：在该方法中，运行在计算机上的规划算法得到环境的地图，并且计算信息反馈规划，基于所有可能产生的 I-状态信息反馈规划能够立刻指示所采取的行

动（可以使用一个衍生 I-空间）。在执行期间，从所存储的、预先计算的规划中可以立刻确定所采取的行动。

- **懒惰规划**：在该情况下仍然给定地图，但是相应的行动只在每个执行阶段需要时才进行计算。机器人上的计算机必须能够基于目前的 I-状态计算所采取的行动。

该问题与第 5.6 节中基于采样的路线图方法相似。如果需要执行得更快，那么可以用预计算规划。如果预计算规划需要消耗太多的时间或空间，那么懒惰规划是更可取的。

使用这些方法时可以回顾第 12.1.1 节中的表述，那里将 $\mathcal{I}_{ndet}$ 当作一个新的状态空间 $\tilde{X}$，在该空间上可以用状态反馈。即使在没有大自然感测行动的情况下，观测也是不可预测的，因为状态通常是未知的。这意味着 $\tilde{\theta}$ 是未知的，在给定 $\tilde{x}_k$ 和 $\tilde{u}_k$ 情况下，将来的新状态 $\tilde{x}_{k+1}$ 是不可预测的。因此规划必须使用反馈，即它需要执行过程中的信息来解决问题。在例 12.1 和例 12.2 中为定位问题而阐述在新状态空间上的状态转移函数 $\tilde{f}$。初始状态 $\tilde{x}_I$ 是所有原始状态的集合。如果没有对称性，那么目标集合 $\tilde{x}_G$ 是 X 的所有单元素子集的集合；否则，它是所有可能到达的最小 I-状态的集合（这不必提前进行构造）。如果需要，可以定义代价项来产生一个最优规划问题。例如，如果发生运动，那么 $\tilde{l}(x,u)=2$，否则 $\tilde{l}(x,u)=1$。

考虑预计算规划的方法。第 12.1.2 节的方法可以应用于计算一个规划 $\pi:\tilde{X}\rightarrow U$，它解决从任何初始非确定的 I-状态开始的定位问题。该方法是空间密集型的，因为必须为 $\tilde{X}$ 中的每个状态存储一个行动。如果有 n 个格片，那么 $|\tilde{X}|=2^n-1$。如果初始 I-状态总是 X，那么有可能将 π 限制在 $\tilde{X}$ 的一个非常小的部分。从任意 $\tilde{x}\in\tilde{X}_G$，可以进行基于后向投影的搜索。如果初始 I-状态增加到 S，那么部分规划就能够可靠地定位机器人。没有指定 π 的那部分 $\tilde{X}$ 是从来不会到达的，因此可以将其忽略。

现在考虑懒惰方法。执行行动，然后观察产生了什么 I-状态，那么在机器人上运行的算法就能够进行一种搜索。这导致了 $\tilde{X}$ 上的有向图，该图是通过机器人的运动而逐步展示出来的。因为关于状态的信息逐渐得到改进提高，所以图是有方向的。例如一旦机器人知道它的状态（或者状态的对称类），那么它不能返回到代表更大不确定性的 I-状态。在许多情况下，机器人在执行过程中可能很幸运，它使用比为预计算的规划所需内存少得多的内存就可以进行定位。

为了确保搜索的系统性，机器人需要识别从不同方向到达的相同位置。即使机器人不一定知道它在地图中的位置，它通常也能够判断它先前是否到过某位置。一种方法是给曾经访问过的位置分配一个坐标 (i,j)。从 $(0,0)$ 初始位置开始。如果应用 F，那么假设已经到达位置 $(1,0)$，也就是机器人移向一个新的栅格。如果应用 R，那么如果机器人不被阻挡，那么到达 $(0,1)$。采用 (F,F,R) 或者 (R,F,F) 可以到达 $(2,1)$。一种解释是 $\mathbb{R}^2$ 中的一个局部坐标系固定在机器人的初始位置，这称为里程表坐标（odometric coordinates）。该坐标系和地图之间的方向在开始时未知，但是如果机器人能够精确确定它自己的位置，那么可以在它们两者之间进行变换。

现在多种搜索算法的定义可以从初始状态 $\tilde{x}_I$ 开始，尝试行动直到满足目标条件（例如没有更小的非确定的 I-状态是可达的）。但是这种搜索和第 2.2.1 节算法所进行的搜索之间有一个关键的区别。在第 2.2.1 节中，搜索可以从以前探索过的任何状态继续进行，而不需要付出更多的代价。在目前情况下，有两个问题。

- **变更路径**：大多数搜索算法使得可以在任意时刻从任意以前考虑过的状态延伸出新的状态。对于懒惰方法，机器人必须移动到一个状态并且应用一个行动来确定是否能够到达一个新的状态。机器人能够通过里程表坐标来返回到任意以前考虑过的状态。这就产生了以前搜索问题中所不存在的代价。不是在搜索树中从一个地方跳到另外一个地方，搜索是机器人所遍历的一条长的连续路径。我们将跳跃称为变更路径（reroute path）。这在第 12.3.2 节是非常重要的。
- **信息改进**：机器人可能不能返回到以前的非确定的 *I*-状态。例如，如果机器人按照（F,F,R）进行行动并且试图用（B,L,F）返回到相同的状态，那么它将确实知道它回到相同的状态，但是该状态仍然未知。但是也可能是这样的情况：执行（F,F,R）以后，将能够限定目前状态的可能性。用（B,L,F）返回时，非确定的 *I*-状态将是不同的。

这些问题意味着搜索算法应该考虑移动一个机器人所付出的代价，也预示着搜索图是有方向的。第二个问题实际上并不是问题，因为即使当返回到相同的位置时 *I*-状态可能不同，但它至少与以前一样好。这意味着如果 η_1 和 η_2 是来自于相同位置的原始的和后来的历史 *I*-状态，那么 $X(\eta_2) \subseteq X(\eta_1)$ 总是成立的。在这种情况下的定位问题中信息总是在改进提高。因此试图返回到以前 *I*-状态的同时，机器人将发现一个改进的 *I*-状态。

- **其他信息模型**　本节到目前为止所给出的模型只是许多有趣模型中的一部分。例如，假设机器人有一个能指示其方向的罗盘，那么就没必要将方向作为状态的一部分来跟踪。机器人可以用罗盘来指定与全局方向有关的行动。假设 $U = \{N,E,W,S\}$ 分别表示“北”、“东”、“西”、“南”4 个方向。那么例 12.1 和例 12.2 也就变得毫无意义了。例 12.1 可以用行动序列 (E,N) 来解决。例 12.2 不存在对称性问题，它也可以用行动序列 (E,N) 来解决，因为 (1,2,3) 是仅有的、与行动和罗盘读数相符的位置序列。

如果给机器人提供更少信息，那么就产生了另外一种有趣的模型。在目前的模型中，机器人可以轻而易举地推断出它目前相对于开始位置的位置。即使不一定知道开始位置在地图上的什么地方，它也总可以用相对坐标来表达。这是因为机器人依赖于不同形式的里程表。例如，如果方向是 E 并且机器人执行（L,L,L）序列，那么它会知道方向是 S，因为 3 次左转产生 1 次右转。假设不是在栅格中，而是机器人必须探测一个图。它应用行动经过一条边就能够直接从一个顶点移动到另外一个顶点。令这是一个嵌入 $\mathbb{R}^2$ 的平面图并用线段来绘制。在每个顶点上的可用行动的数量会发生改变。我们通常定义 $U = \mathbb{S}^1$，并且当特定的方向被阻挡时，机器人只能够旋转，而不能进行平移（这是对栅格情况的一种推广）。可以定义一个传感器，它指示从目前顶点采取什么行动才能够导致平移。在这种情况下，该模型对原始栅格模型进行了很好的推广。如果机器人知道到达同一个顶点的多条边之间的角度，那么它可以用角度里程表产生 $\mathbb{R}^2$ 中的局部坐标系来跟踪它的相对位置。

对机器人来说，情形可能会非常混乱。假设不采用 $U = \mathbb{S}^1$，每个顶点处的行动集合指示可以经过哪条边。机器人应用行动就可以经过一条边，但是它不知道关于相对于其他边的方向的任何信息。在这种情况下就不再使用角度里程表。例如它不能区分经过菱形、梯形或者四边形有什么不同。如果有可能用角度里程表，那么通过记录每个顶点处的边之间

的角度就可以避免某些对称性。但是新的模型不允许这样。有相同角度的所有顶点都是相同的。

12.2.2 连续定位的组合方法

现在考虑 X 是 $\mathbb{R}^2$ 中的连续区域情况下的定位。假设 X 以一个简单多边形为界（一个封闭的多边形链，内部没有孔）。给机器人提供 $\mathbb{R}^2$ 中 X 的一个地图。机器人的速度 $\dot{x}$ 直接由行动 u 来控制，这样的运动模型为 $\dot{x}=u$，对于该模型来说，U 是以原点为中心的单位球。这使得规划被指定为 X 中的一条连续路径。因此不用 u 来指定速度，而是直接指定路径，这样更简单。$\dot{x}=u$ 和更加一般的 $\dot{x}=f(x,u)$ 形式的模型分别参见第 8.4 节和第 13 章。

机器人使用两种不同的传感器。

（1）罗盘：一个好的罗盘可以解决第 12.2.1 节中出现的方向问题。

（2）可视性：图 11.15 所示的可视性传感器能够提供各个方向上的距离测量。

这两种传感器都没有大自然感测行动。

就像第 12.2.1 节所介绍的，定位涉及计算非确定的 I-状态。在目前情况下由于有罗盘和已知 $\mathbb{R}^2$ 中多边形的方向，所以没有必要将方向作为状态空间的一部分来表示。因此非确定的 I-状态只是 X 的子集。假设为图 11.15 所示的例子计算非确定的 I-状态，但是没有任何历史。这是式（11.6）所定义的 $H(y)\subseteq X$。给定传感器的目前读数，要求从所能得到的距离测量中计算状态，如图 11.15（b）所示。也就是说必须发现能够很好地覆盖图 11.15（b）所示的边的平移，这些边在图 11.15（a）所示的多边形边的上部。令 ∂X 表示 X 的边界。可视性传感器所得到的距离测量必须准确对应于 ∂X 的一个子集。对于该例子，它们只能从一个状态中得到，如图 11.15（a）所示。因此机器人不必通过移动来确定它自己的位置。

就像第 8.4.3 节中介绍的那样，令可视多边形 $V(x)$ 表示图 11.15（a）所示 x 可见的所有点的集合。为了有效地实现所需要的计算，必须通过处理多边形来确定可视多边形从 X 中的各点出现的不同方式。这涉及仔细确定 ∂X 的哪条边能够出现在 $\partial X(x)$ 上。可以将状态空间 X 剖分为有限数量的胞腔，每个区域上的不变性是指用边的相同集合来描述 $V(x)$ [138,418]。图 12.4 显示了一个例子。必须延伸出两种不同类型的射线来进行剖分。图 12.5 显示了一种情况：一对顶点相互可视并且有可能延伸一条向外的射线。其他情况如图 12.6 所示，其中射线在每个反射顶点向外延伸（顶点的内角大于 π，参见第 6.2.4 节）。那么在最坏情况下剖分能够产生 $O(n^2r)$ 个胞腔，其中 n 是形成 ∂X 的边的数量，r 是反射顶点的数量（需要注意的是 $r<n$）。一旦从传感器得到测量，那么就需要计算其中边或者距离测量很好匹配的一个或者多个胞腔来确定 $H(y)$（X 中的点集，从这些点可以得到目前距离测量）。文献[418]给出了基于可视骨架（visibility skeleton）的算法，它在 $O(m+\lg n+s)$ 时间内实现这些计算，且用到的空间为 $O(n^5)$，其中 n 是 ∂X 中顶点的数

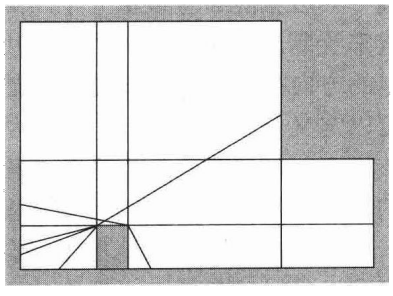


图 12.4 可视胞腔剖分的一个例子：在每个胞腔内，可视多边形由 ∂X 的相同的边组成

量，m 是 $V(x)$ 中顶点的数量，$s=|H(y)|$ 是非确定的 I-状态的数量。这种方法假设对环境进行预处理来实现执行过程中的快速查询。如果没有经过预处理，$H(y)$ 可以在 $O(mn)$ 时间内计算完毕。

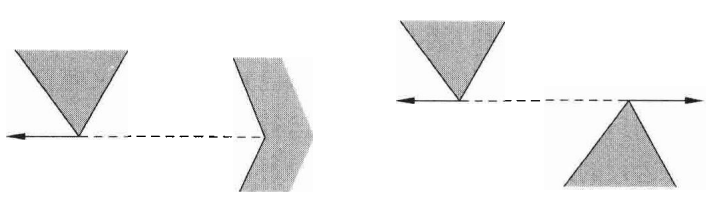


图 12.5　在可能的时候，射线从两个相互可视的顶点对向外延伸；右边是图 6.10 所示的一条双切线；但是这里的边向外延伸，而不是向内延伸，如可视图所要求的那样

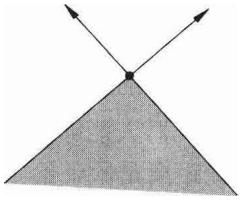


图 12.6　一个反射顶点：如果顶点的内角大于 π，那么两条向外的射线从角的邻边向外延伸

如果有多个状态匹配可视性传感器的距离数据，那么会怎样？因为文献[418]中的方法只计算 $H(y)\subseteq X$，所以必须规划机器人的某些运动来进一步减少不确定性。这也从另外一个角度阐述了 I-空间的作用。即使状态空间是连续的，这种情况下的一个 I-状态也可以从可能性的有限集合中消除状态的模糊性。

例 12.3 来自于文献[300]。

例 12.3（基于可视性的定位） 考虑图 12.7 所示的环境及初始状态。基于可视性传感器观测，初始状态可以是图 12.8 所示的四种可能性之一。因此 $H(y_1)$ 包括四种状态，其中 y_1 是传感器的初始观测。假设执行图 12.9 所示的运动序列。执行第一步以后，机器人的位置缩小到两种可能性，如图 12.10 所示。这是因为对于其余两种可能性来说走廊会更长。执行第二步以后，状态能够完全确定，因为已经探测到短的侧向走廊。 ■

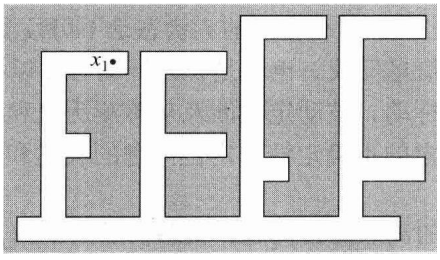


图 12.7　初始状态未知的例子[300]

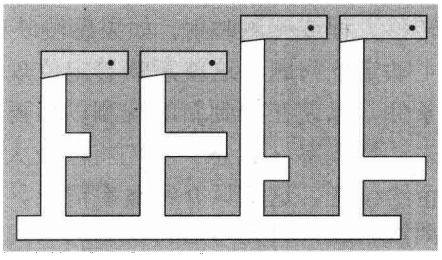


图 12.8　基于可视性传感器，图 12.7 中机器人的四种可能的初始位置

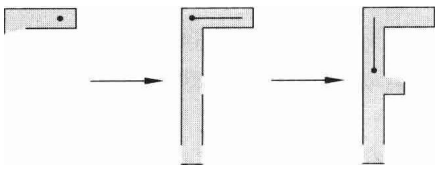


图 12.9　这些运动完全消除了状态的模糊

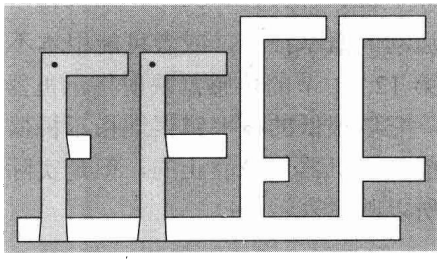


图 12.10　现在只有两种可能的状态

一般情况下，定位问题可以通过可视胞腔剖分来解决，如图 12.4 所示。初始情况下，

从初始可视多边形计算 $X_1(\eta_1)=H(y_1)$，用可视骨架就可以有效地完成这种计算[418]。假设 $X_1(\eta_1)$ 包含 k 个状态。在这种情况下重叠着地图的 k 个平移副本，使得 $X_1(\eta_1)$ 中所有的可能状态相一致。然后执行一个能够减小不确定性的运动。例如这可以通过穿过重叠中的一个胞腔的边界来完成，胞腔边界对应于一个或者多个地图副本，但不是全部 k 个副本。这使得可以从下一个 I-状态 $X_2(\eta_2)$ 中排除掉某些可能的状态。再次使用重叠就可以得到另外一个消除模糊的运动，它产生 $X_3(\eta_3)$。这种过程一直进行直到状态已知为止。在文献[300]中给出使机器人能够确定其自身位置的运动规划，采用的方法是运行不超过 k 次的最优距离，最优距离是验证给定状态所需要运行的距离。例 12.3 显示了这种特殊的定位问题的解决似乎并不太困难，但是如果允许任意简单多边形，那么用最优运动来进行的定位问题是 NP 复杂度的。在文献[300]中该结论的证明是通过下面的方法：每个抽象的决策树可以作为一个定位问题来实现，而且已经知道抽象的决策树是 NP 复杂度的。

通过改变感测模型可以构造连续空间中定位问题的一些有趣的变化形式。例如，假设机器人测量的距离有一个界限，机器人看不见界限以外的所有点。这对应于一些实际的感测系统，如可移动机器人的红外传感器、声纳以及测距扫描仪等。这可能扩大了 $H(y)$。假设机器人只在有限的方向上能够进行距离测量，如图 11.14（b）所示。另外一个有趣的变化是去掉罗盘。这就产生了第 12.2.1 节所观测到的方向混乱问题。甚至可以考虑没有感测或者很少感测的定位问题[754,755]，这样所产生的 I-空间与图 11.28 中倾斜托盘的例子所产生的 I-空间相似。

12.2.3 定位的概率方法

到目前为止，所考虑的定位问题只涉及非确定的不确定性，并且都假设大自然不影响状态转移方程或感测映射。如果感测映射中涉及大自然，那么将来的 I-状态是不可预测的。对于主动定位问题，这就意味着定位规划必须使用信息反馈。也就是说，行动必须以 I-状态为条件，以使在得到新的观测以后再采取相应的行动。被动定位涉及从感测历史和行动历史来计算概率 I-状态。我们将涉及大自然和非确定的不确定性的定位问题的表述和解决方法留给读者。这里只介绍概率情况。

离散问题 首先考虑将概率加入到第 12.2.1 节的离散栅格问题。状态仍然用 $x=(p,d)$ 表示。初始条件是 X 上的一个概率分布 $P(x_1)$。一个合理的选择是将 $P(x_1)$ 作为一个均匀概率分布，那么各个方向和位置都有等可能性。仍然给出机器人的四个行动，但是现在假设大自然干扰状态转移。例如，如果 $u_k=F$，那么机器人向前运动的概率很大，但是向右、向左运动的概率很小，或者可能根本不运动，即使没有受到阻挡。

第 12.2.1 节的传感器映射指示机器人是否运动。在目前情况下，大自然可以干扰这种测量。它以很低的概率错误地指示机器人在运动，而实际上机器人是静止的。相反地，它也可能指示机器人是静止的，但是实际上机器人却是运动的。因为传感器依赖于前两个状态，所以映射表示为：

$$y_k=h(x_k,x_{k-1},\psi_k) \tag{12.19}$$

如果给定概率模型 $P(\psi_k)$，那么这可以表示为 $P(y_k|x_k,x_{k-1})$。

为了解决被动定位问题，可应用第 11.2.3 节的计算衍生 I-状态的表达式。如果传感器

映射只用当前的状态，那么式（11.36）、式（11.38）和式（11.39）将不需要进行修改就可以直接应用。但是因为 h 依赖于 x_k 和 x_{k-1}，那么需要进行某些修改。回忆到这种传感器的观测从 y_2 开始。因此 $P(x_1|\eta_1)=P(x_1|y_1)=P(x_1)$，而不是用式（11.36）。

每个阶段以后，首先用式（11.38）来考虑行动 u_k 就可以从 $P(x_k|\eta_k)$ 计算 $P(x_{k+1}|\eta_{k+1})$。式（11.39）考虑传感器观测 y_{k+1}，但是没有给定 $P(y_{k+1}|x_{k+1},\eta_k,u_k)$，因为传感器映射也依赖于 x_{k-1}。使用边缘化，它变成：

$$P(y_k|\eta_{k-1},u_{k-1},x_k)=\sum_{x_{k-1}\in X}P(y_k|\eta_{k-1},u_{k-1},x_{k-1},x_k)P(x_{k-1}|\eta_{k-1},u_{k-1},x_k) \tag{12.20}$$

因为观测只依赖于 x_{k-1}、x_k 和大自然感测行动 ψ_k，所以和式中的第一个因子可以变成传感器模型：

$$P(y_k|\eta_{k-1},u_{k-1},x_{k-1},x_k)=P(y_k|x_{k-1},x_k) \tag{12.21}$$

式（12.20）中的第二个因子可以用贝叶斯公式来计算：

$$P(x_{k-1}|\eta_{k-1},u_{k-1},x_k)=\frac{P(x_k|\eta_{k-1},u_{k-1},x_{k-1})P(x_{k-1}|\eta_{k-1},u_{k-1})}{\sum_{x_{k-1}\in X}P(x_k|\eta_{k-1},u_{k-1},x_{k-1})P(x_{k-1}|\eta_{k-1},u_{k-1})} \tag{12.22}$$

其中 $P(x_k|\eta_{k-1},u_{k-1},x_{k-1})$ 被简化成 $P(x_k|u_{k-1},x_{k-1})$。这可以从状态转移概率中直接得到，将阶段索引向前移动就可以表示状态转移概率 $P(x_{k+1}|x_k,u_k)$。$P(x_{k-1}|\eta_{k-1},u_{k-1})$ 由式（11.38）给出。这样就完成了概率 I-状态的计算，同时也解决了被动定位问题。

解决主动定位问题十分困难，因为需要在 $\mathcal{I}_{prob}$ 上进行搜索。像离散定位问题一样也存在相同选择问题。在整个 I-空间 $\mathcal{I}_{prob}$ 上计算信息反馈规划在理论上是可能的，但是在大多数环境下是不现实的。这里也可以应用基于搜索的思路，该思路在第 12.2.1 节被用来递增地生长一个有向图。该方法的成功取决于为这种特定问题而采取的巧妙的启发式搜索。

连续问题　在连续空间中使用概率模型所进行的定位问题在近年来受到广泛的重视[260,450,625,828,888,962]。由于存在传感器数据噪声、建模误差以及对运算的长期鲁棒性有很高的需求等缘故，对可移动机器人的定位通常是很困难的。概率建模以及概率 I-状态的计算在许多试验系统中已经非常成功，无论可移动机器人是在室内还是在室外。图 12.11 显示了只用声纳成功进行的定位。这里的大量工作都涉及被动定位，因为机器人通常正在进行一些其他工作，比如正在向环境的一个特定区域前进等。因此注意力主要集中于计算概率 I-状态，而不是在 $\mathcal{I}_{prob}$ 上进行困难的搜索。

连续空间中的概率定位通常涉及概率密度 $p(x_{k+1}|x_k,u_k)$ 和 $p(y_k|x_k)$ 的定义（在状态传感器映射的情况下）。如果阶段表示等间隔的时间，那么这些密度在每个阶段通常保持固定。状态空间通常为 $X=SE(2)$，其中包含平移和旋转，但如果只考虑平移时它是 $X=\mathbb{R}^2$。密度 $p(x_{k+1}|x_k,u_k)$ 说明了在某固定时间区间上控制可移动机器人时所产生的不可预测性。文献[1004]论述了通过求解随机微分方程来为非完整机器人估计这种分布的方法。

密度 $p(y_k|x_k)$ 表明了当给定状态时不同测量间的相对可能性。通常这也就对从激光测距扫描仪、声纳阵列或者红外传感器所得到的距离测量进行了建模。假设机器人在一个二维环境中运动，并从不同方向进行深度测量。在机器人的体坐标系中，在 n 个角度上进行了深度测量。理想情况下，测量应该看起来像图 11.15（b）所示的情况。但是实际上，

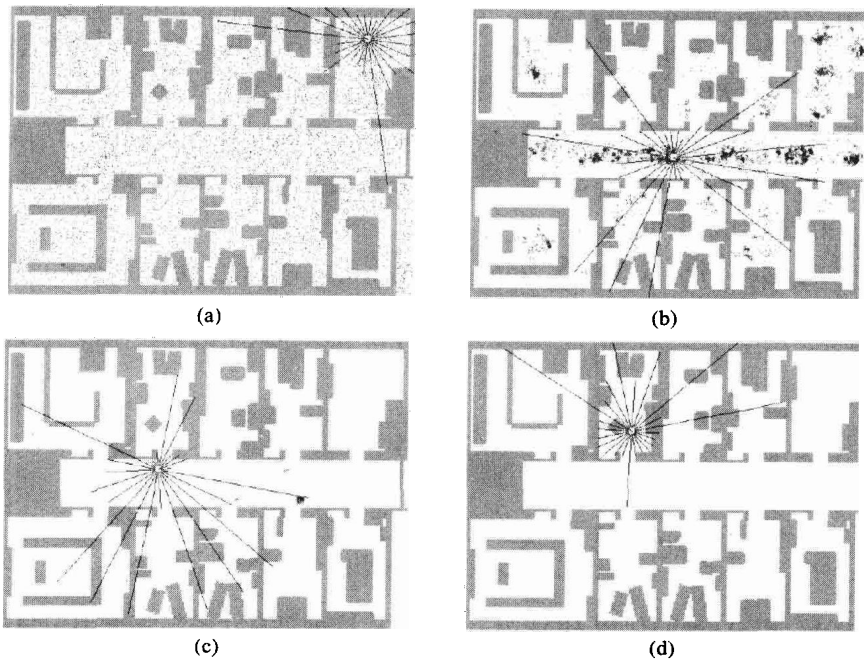


图 12.11 一个动画的 4 帧图像，它表示了一个室内机器人用声纳进行概率定位[352]

数据包含了大量的噪声。观测 $y \in Y$ 是含噪声的深度测量的一个 n 维向量。

定义 $p(y|x)$ 的一个平常的方法是假设每个距离测量的误差服从高斯密度。一旦给定 x，那么很容易将测量的均值作为真值来计算，并且方差应该从传感器的试验计算中确定。如果假设测量所得到的向量是独立同分布的随机变量的集合，那么高斯密度的乘积就得到 $p(y|x)$。

一旦对模型进行了表述，那么概率 I-状态的计算直接采用第 11.2.3 节和第 11.4.1 节中介绍的方法。初始条件是 X 上的一个概率密度函数 $p(x_1)$。那么可应用边缘化和贝叶斯更新规则来构造 $p(x_k|\eta_k)$ 形式的密度函数序列。

在某些应用中，用来表达 $p(x_{k+1}|x_k,u_k)$ 和 $p(y_k|x_k)$ 的模型可能是线性高斯的。在这种情况下，很容易应用第 11.6.1 节中的卡尔曼滤波。但是在大多数情况下，密度没有这种形式。那么可以用第 11.4.3 节中的基于矩的近似方法来近似密度。如果使用二阶矩，那么可以得到扩展卡尔曼滤波，原始问题经过线性高斯近似后仍然可以采用卡尔曼滤波。近年来，在试验性可移动机器人领域中受到广泛认可的一种方法是用基于采样的技术来直接计算和估计概率 I-状态。在定位问题中，第 11.6.2 节描述的粒子滤波方法有很好的试验性能。粒子滤波通常也被称为蒙特卡罗定位，请参见本章结尾的参考文献。

12.3 环境不确定性与制图

阅读第 12.2 节后，读者可能想知道如果没有给定地图，那么会发生什么。这些问题是机器人技术的基础。如果状态表示位形，那么在不需要知道精确的位形的情况下 I-空间也

可以完成该任务。但是如果状态也表示环境，那么甚至在不需要知道环境的完整表示的情况下 I-空间也可以完成任务。这是非常有好处的，因为对机器人的环境进行表示通常是非常昂贵的，也容易受到误差的影响，并且还可能很快过时。

12.3.1　栅格问题

为了清晰地理解问题，还是先考虑离散问题。这里的内容是第 12.2.1 节的内容的继续。在那里，状态表示一个位置 p 和一个方向 d。现在假设状态表示为 (p,d,e)，其中 e 表示包含机器人的特定环境。这就要求定义一个环境空间。其实很少明确表示环境空间，而通常表示为对环境类型的一个约束。例如环境的集合可以被定义为所有连通的二维栅格规划问题。单连通的栅格规划问题的集合可以进一步进行约束。

那么会立刻出现一个问题：一个环境的两个地图在什么时候是等价的？回忆图 12.2(a)和图 12.3(b)所示的地图。对于每个 90°旋转来说，图 12.3(b)中的地图是相同的，但是图 12.2(a)中的地图却不同。即使有不同，它也应该被认为是同一个环境，对吗？假设绘制一个遥远海岛的地图，但没有罗盘来指示北的方向。那么地图的方向可以任意选择。如果环境地图的绘制是在 $\mathbb{R}^2$ 上进行的，那么将会看到两个地图是等价的，如果 $SE(2)$ 中的一个变换能够将一个地图完美地覆盖在另外一个上面。

当定义了一个环境空间，那么还需要清楚地定义两个环境等价意味着什么，这很重要。例如，如果要求通过探测来绘制一个地图，那么还会要求提供精确的平移和旋转吗？可能会，也可能不会，但重要的是在问题的描述中加以明确。因此如果地图只是 $SE(2)$ 中的变换不同，那么根据具体应用，它们可以定义为等价的，也可以不定义为等价的。

为了考虑某些例子，可以定义一些有限的或者无限的环境集合。假设在二维栅格上进行规划。本节假设每个栅格点 p 都有整数坐标 $(i,j) \in \mathbb{Z} \times \mathbb{Z}$，如第 2.1 节所定义。令 E 表示一个环境集合。对于机器人来说，仍然可能面对 4 个方向，令 D 表示该方向集合。状态空间是：

$$X = \mathbb{Z} \times \mathbb{Z} \times D \times E \tag{12.23}$$

通常假设环境 $e \in E$ 由 $\mathbb{Z} \times \mathbb{Z}$ 的一个子集来确定，$\mathbb{Z} \times \mathbb{Z}$ 对应于所有可以放置机器人的白色格片的位置。其他格片都是黑的，这意味着它们是障碍物。如果允许 $\mathbb{Z} \times \mathbb{Z}$ 的任意子集，那么 $E = \mathrm{pow}(\mathbb{Z} \times \mathbb{Z})$。这包括一些无用的地图，如占据整个平面的棋盘等；这就启发产生了对 E 的某些约束。例如 E 可以限制为 $\mathrm{pow}(\mathbb{Z} \times \mathbb{Z})$ 的一个子集，它对应于包括原点 $(0,0)$ 处的一个白色格片的所有地图，对该子集来说，所有其他白色格片都是可达的，并且处在一个有界区域内。

首先考虑能够进行表述的问题的类型：

（1）制图：任务是访问每个可达格片并绘制一个地图。根据 E 是如何定义的，这可以识别 E 中的一个特定环境或是与探测相吻合的一个环境集合。这可以被称为同时定位与制图（simultaneous localization and mapping）或者 SLAM，因为构造一个完整地图通常意味着机器人的位置和方向最终是已知的[486,970]。因此式（12.23）所给出的完备状态 $x \in X$ 由地图的建立过程来确定。对于这里考虑的栅格问题，建立地图是很简单的，但是对于连续环境中的概率不确定性情况来说该问题就变得更加困难。这种情况参见第 12.3.5 节。

（2）确定环境：假设将机器人随机放置在一个建筑中。机器人被告知它处在建筑的一个固定数（例如 10）之中。它必须通过运动来确定这个数是多少。当可能的环境数增加时，问题就更像制图。实际上，制图可以认为是一种特殊情况：初始时对可能环境的集合没有施加或者很少施加约束。

（3）导航：在这种情况下，目的是到达目标位置，即使机器人没有地图。目标相对于机器人的位置可以通过传感器来确定。允许机器人在没有完全探测环境的情况下解决该问题。因此在完成任务以后，最后的非确定的 I-状态可以包含大量的可能环境。而解决问题只需要其中一部分。

（4）搜索：在这种情况下，目标状态只在到达它时才能识别出来（或者被短距传感器检测到）。在搜索中没有另外的传感器来协助。所以必须对环境进行系统的探测，如果发现目标，搜索可以提前结束，不必构造地图。搜索可以扩展到第 12.4 节介绍的追逃问题。

现在给出确定环境和导航的简单例子。

例 12.4（确定环境） 假设告知机器人它被放置在如图 12.12 所示的环境之一中。令机器人的初始位置是(0,0)，如白色圆圈所示。令初始方向向东、环境为 e_3。这些情况对于机器人是未知的。使用与第 12.2.1 节相同的行动和状态转移模型。目前的状态空间包括环境，但是环境不会发生变化。只有关于机器人处在哪个环境的信息会发生变化。感测模型只指示机器人应用最后一个行动后它的位置是否已经改变。

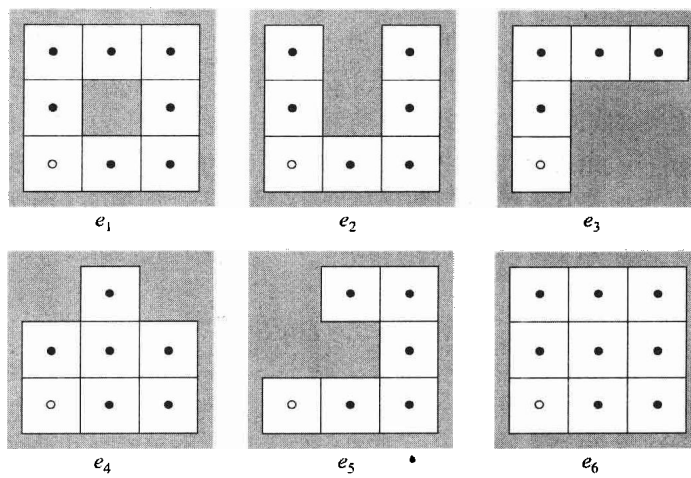


图 12.12 可能的二维栅格环境的一个集合；在每种情况下，“向上”的方向代表北，白色的圆圈代表原点 $p=(0,0)$

初始条件是 X，因为任意位置、方向和环境都是可能的。现在确定某些非确定的 I-状态。令 $(u_1,u_2,u_3)=(\mathrm{F},\mathrm{R},\mathrm{R})$。从该行动序列，传感器观测 (y_2,y_3,y_4) 报告机器人没有改变其位置。方向改变到向西，但是机器人并不知道（但是知道它现在正指向与初始方向相反的方向）。现在可以推断出什么？机器人已经发现它处在三面被障碍物所限制的一个格片上。这意味着从可能环境中排除 e_1 和 e_6。在剩余的 4 个环境中，机器人断定它必定处在最后几个格片其中之一上：（1）e_2 的左上部；（2）e_2 的右上部；（3）e_3 的下面；（4）e_3 的最右边；（5）e_4 的上面；（6）e_5 的左下部；（7）e_5 的左上部。也可以对其方向进行判断。甚至知道

行动 $u_4 = \mathrm{R}$ 将引起它移动，因为所有 4 个方向都没有阻挡。

应用 $(u_4, u_5) = (\mathrm{R},\mathrm{F})$。机器人将移动两次，到达 e_3 的左上部、面向北。在这种情况下，e_2、e_3、e_4 或者 e_5 都是可能的，但是现在知道在第 4 阶段它的位置不会在 e_5 的左上部。如果机器人在 e_3，那么它知道它一定在左上部，但是它仍然不知到它的方向（它可能向北或者向西）。机器人也可以在 e_2 的左下部或者右下部。

现在令 $(u_6, u_7) = (\mathrm{R},\mathrm{F})$，使机器人移动两次。这样，$e_4$ 和 e_5 将被排除掉，那么可能的环境集合是 $\{e_2, e_3\}$（e_2 的一个方向也被排除掉）。如果应用 $u_8 = \mathrm{R}$，那么传感器观测 y_9 报告机器人没有移动。这又排除了 e_2。最后机器人能够推断出它在 e_3 的右上部、面向南。它还能够判断它的初始状态是在 e_3 的左下部、面向东。因此通过构造非确定的 I-状态，所有的不确定性都已经被消除掉。

现在考虑将图 12.13 所示的环境加入到集合，再次讨论这个问题。环境 e_7 与环境 e_1 相同，除了原点发生移动以外，e_8 和 e_2 相同，只是旋转了 180°。在这两种情况下，机器人无法区分 e_1 和 e_7、e_2 和 e_8。可以声明这些环境是分段等价的。它们之间的区别仅在于制图的方式不同。

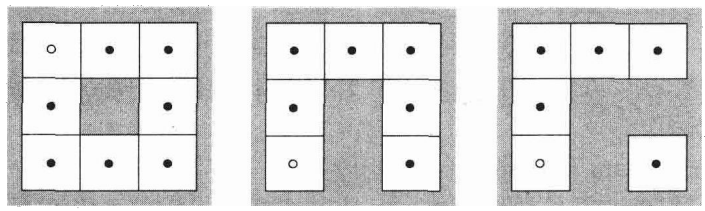


图 12.13　将这些环境加入到图 12.12 所描述的集合中；本质上它们都等价于一个已经给出的环境，并且通常不会影响规划问题

如果机器人执行与以前相同的行动序列，那么它将不能区分 e_3 和 e_9。对机器人来说不可能判断在不能到达的某个地方是否存在一个白色的格片。一般环境空间可能包括这种变化，这就妨碍了机器人精确确定其环境。但是这并没有给解决规划问题带来额外的困难。因此可以声称 e_3 和 e_9 是等价的。在没有精确知道环境的情况下就能够完成任务，这样一个事实是非常重要的。在某种意义上，对环境的观测是在足够解决问题的“分辨率”上进行的，那么比此更细的分辨率并不重要。因为机器人能够忽略不必要的细节，所以通常可以构造更便宜而且更可靠的系统。■

例 12.5（到达一个目标状态） 仍然假设给定如图 12.12 所示的环境集合。不过这次还假设已知位置 $p = (0,0)$ 和方向向东。环境是 e_4，但它对机器人来说是未知的。任务是到达位置 $(2,0)$，即机器人必须向东移动两个格片。在没有提供更多关于环境信息的情况下，规划 $(u_1, u_2) = (\mathrm{F},\mathrm{F})$ 能够实现目标。应用 $u_1 = \mathrm{F}$ 以后，知道环境不是 e_3，但是这里没有收集关于环境的额外信息，因为这与解决问题无关。如果目标是到达 $(2,2)$，那么应该得到关于环境的更多信息。例如，如果规划是 $(\mathrm{F},\mathrm{L},\mathrm{R},\mathrm{L})$，那么将会知道环境是 e_6。■

确定环境的算法　为了确定环境（它包括制图问题），到达并记住所有格片就足够了。如果机器人必须从一个可能性集合中确定环境，那么可以预先计算最优的最坏情况规划。这可以用第 10.2.3 节的值迭代方法或者非确定的 Dijkstra 算法在 $\tilde{X} = \mathcal{I}_{ndet}$ 上进行计算。当

机器人落入该环境，它应用最优规划来判断它的位置、方向以及环境。如果可能环境的集合太大（可能是无穷的），那么采用懒惰方法是最合适的。这包括制图问题，对它来说可能很少有，也可能没有关于环境的假设。制图问题的懒惰方法必须保证每个格片都被访问到。另外一个问题是最小化变更路线的次数，这在第 12.2.1 节提到过。既能解决问题，又能避免过多地变更路线的一个简单算法是第 2.2.2 节的深度优先搜索。

导航算法 导航的任务是到达指定目标，即使没有给定环境地图。假设目标是用相对于机器人初始位置和方向的坐标来表示的（这是里程表坐标）。如果目标只在机器人到达目标格片时才能识别出来，那么需要进行搜索，这将在稍后进行介绍。就像在例 12.5 中所见，不要求机器人学习整个环境来解决导航问题。可以应用第 2.2 节的搜索算法。例如 A^* 算法能够发现到达目标的最优路线，通过假设所有格片都是空的，这样可以定义尚需代价的启发性估计。尽管这种方法能够正常运行，但是没有考虑变更路线的代价。因此 A^* 计算的最优路径可能是毫无意义的，除非其他机器人后来利用该信息到达相同环境中的相同目标。对于首先出发的机器人来说，可能要进行大量的探测，因为 A^* 不是为执行过程中进行探测而设计的。即使第 2.2 节的搜索算法假设搜索图会在执行过程中逐步呈现出来，而不是提前给定的，它们也允许在搜索中目前的状态有跳动。在目前情形下，这要求将机器人送到环境的各个部分。第 12.3.2 节将介绍导航算法，它将 Dijkstra 算法扩展到当执行过程中产生代价时也能正常工作。它可以很好地应用于本节提到的基于栅格的导航问题，甚至在初始环境未知的情况下。

迷宫搜索算法 Blum 和 Kozen 给出使用 I-映射来显著减小 I-空间的一个例子[121]。地图的绘制要求空间是格片数的线性函数，但是有可能用更小的空间来对环境进行系统的搜索。对于二维栅格环境，不需要维持一个完整地图就能够解决搜索问题。它必须系统地访问每个格片，但是这不意味着它必须记住所有已经访问过的地方。重要的是只需要保证机器人在覆盖所有格片之前不陷入一个无限循环中。文献[121]指出：搜索任何迷宫所使用的空间都可以只是格片数的对数。这意味着许多不同的环境在机器中有相同的表示形式。从本质上讲，所研究的 I-映射将 $\mathcal{I}_{ndet}$ 收缩成一个更小的衍生 I-空间。

假设机器人运动模型与本节所给出的相同，但是初始时没有给出环境的地图。机器人初始时无论面向什么方向，都可以声明为北向，这没有什么危害。假设任何平面二维栅格都是可能的，因此对于 4 个方向来说，地图都是一样的。其中一个地图的北向可能通过将任意初始方向声明为北向来标注。但是这对于即将介绍的方法来说并不重要。假设机器人是带有二进制计数器的一个有限自动机。用计数器是因为它能存储任意大的值，而自动机不能。

为了防止机器人陷入无限循环，需要维持两个重要的信息：

（1）纬度（latitude），它是从机器人初始位置在向北方向上的格片数。

（2）当执行一个环形路径时，机器人需要知道它的定向，即环路是顺时针进行，还是逆时针进行。

这两者都可以从历史 I-状态计算而来，历史采用与式（12.12）相同的形式，除了目前情形下 X 是由式（12.23）给出、E 是所有有界环境的集合（有界意味着白色格片可以包含在一个大的矩形中）以外。从历史 I-状态，令 $\tilde{u}'_k$ 表示行动历史的子序列，行动历史对应于产生运动的行动。纬度 $l(\tilde{u}'_k)$ 可以用产生向北运动的行动次数减去产生向南运动的行动次数

来计算。环的方向可以用角度里程表来确定（角度里程表等价于该问题有一个罗盘[288]）。令值 $r(\tilde{u}'_k)$ 给出 $\tilde{u}'_k$ 中向右转的次数减去向左转的次数。注意到进行四次向右转就产生了一个顺时针环，且 $r(\tilde{u}'_k)=4$。进行四次向左转就产生一个逆时针环，且 $r(\tilde{u}'_k)=-4$。一般情况下，对于自身不相交的任意环路来说：要么 $r(\tilde{u}'_k)=4$，这意味着它沿顺时针运动；要么 $r(\tilde{u}'_k)=-4$，即它沿逆时针方向运动。

我们已经论述过，需要一个有限自动机和一个二进制计数器。当机器人运动时，计数器用来跟踪 $l(\tilde{u}'_k)$。不需要另外一个计数器来测量角度里程，因为当记录右转和左转次数时，机器人能够进行模 3 运算。如果形成一个环以后模 3 运算的结果为 $r(\tilde{u}'_k)=1$，那么机器人沿逆时针运动。如果模 3 运算的结果为 $r(\tilde{u}'_k)=2$，那么机器人沿顺时针运动。与维持纬度的情况相比较而言，这避免产生无限比特。目前的构造过程可以看作是一个 I-映射的其中一部分，它将历史 I-状态映射到一个更小的衍生 I-空间。

规划将用图 12.14 所示的例子来描述。对于任何一个环境，它内部有障碍物（这个例子有 6 个），并且还有一个外部边界。用纬度和方向信息可以在每个障碍物的边界上和环境的外部边界上确定一个独特点。独特点（unique point）被定义为障碍物最南边的顶点中最西边的顶点。如图 12.15 中小圆圈所示。通过使用纬度和方向信息，总能够发现独特点（见习题 4）。

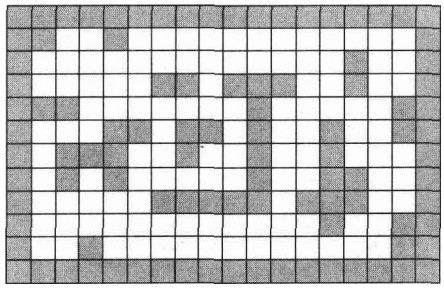


图 12.14　有六个障碍物的例子

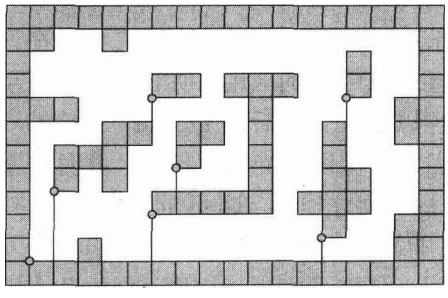
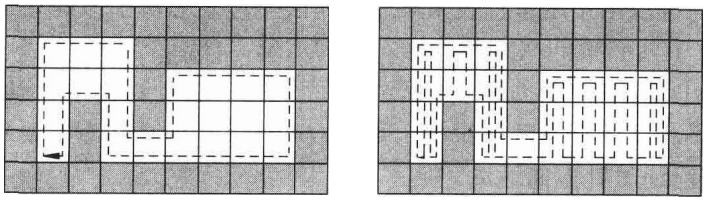


图 12.15　从每个障碍物的唯一一个点向下延伸出一个细的障碍物，这样将障碍物连接在一起

为了解决该问题，机器人向边界运动并顺着墙绕过该边界。机器人可以用它的感测信息来进行运动，运动的方式是使墙始终在其左边。假设机器人沿着边界总能够检测一个独特点，那么可以假设障碍物被图 12.15 所示的那样连接起来。有一个假想的细的障碍物，它从每个独特点向南延伸。用这种方式将障碍物连接在一起。也就是说，假设所有障碍物是从墙中伸出来的，而不是“漂浮”在环境中。通过拒绝穿过这些假想的障碍物，机器人在单个闭环路径中所有障碍物的边界附近运动。到目前为止的策略都不能保证每个胞腔都能够被访问到。因此如图 12.16 所示，改用“之”字形运动来保证每个格片都被访问到。可以将该问题的解与第 7.6 节中的生成树覆盖规划方法进行比较，那里假设给定一个完整的地图，并且目标是最优化运动距离。

如果环境中的某特殊目标只有在机器人到达后才能够被检测到，那么给出的策略总能够保证发现该目标，即使到最后结束时也没有一张地图。

所产生的方法可以被认为是 I-空间上的一个信息反馈规划。在这种意义上，Blum 和 Kozen 发现了能够解决任何问题的规划的“规划器”。不需要从问题的数据中计算其他规划，



(a) (b)

图 12.16 （a）顺墙运动所产生的一个逆时针环路；（b）访问内部所有格片的另外一个环路

因为规划不需要修改就能够处理所有可能的环境。与需要状态估计的情况相比而言，这就是在环境的集合上采用 I-空间的优势所在。

12.3.2 Stentz 算法（D*）

假设一个机器人探索一个未知的星球。机器人沿着崎岖不平的地形运动，同时用扫描测距器对它周围进行精确测量。当机器人运动时，它可能会发现某些地方比它原先认为的更容易通过。而在其他情况下，由于有悬崖或者峡谷，它也可能会意识到它想要前进的方向是不可能的。如果机器人的目标是到达某特定坐标，那么该问题就可以被认为是在未知环境中的一个导航问题。解决方案采用懒惰方法，参见第 12.2.1 节。

本节提出 Stentz 算法[914]，它已经在许多室外车辆导航中得到应用，如图 12.17 所示。该算法可以被认为是后向 Dijkstra 算法的动态形式。因此它保留了尚需代价值，并且搜索过程从目标开始进行，这不同于第 2.3.3 节 Dijkstra 算法从 x_I 开始的已付代价值。该方法可以应用于任何最优规划问题。根据状态转移图，假设边的转移的代价是未知的（等价地，每个代价 $l(x,u)$ 是未知的）。在导航问题中，正的代价表示从状态 x 到状态 $x' = f(x,u)$ 运动的困难程度。



图 12.17 无人自动巡逻车（XUV）装备了激光雷达和其他传感器，它用 Stentz 算法进行导航（General Dynamics Robotic System 提供）

为了通过具体问题来进行分析，假设一个星球的表面被分拆成高分辨率的栅格。状态空间只是格片的一个有界集合，因此 $X \subseteq \mathbb{Z} \times \mathbb{Z}$。给每个格片赋予一个正的实数值 $c(x)$，它表示经过该格片的困难程度。可以用标准邻近格来选择每个格点上的行动 $U(x)$（例如四邻近格或者八邻近格）。那么现在就定义了 X 上的一个状态转移图。从任意 $x' \in X$ 和 $u' \in U(x')$ 使得 $x = f(x',u')$，用 c 来赋值的代价项为 $l(x',u') = c(x)$。该模型是对第 12.3.1 节中栅格的推广，在那里格片要么为空，要么被占据，而在这里则允许任意正实数值。在即将进行的说明中，所用的代价可能比从 $c(x)$ 开始的代价更一般，并且状态转移图不需要从栅格产生。给所有 $l(x,u)$ 赋予某些任意初始值。例如在行星探测中，假设经过一个平坦的、无障碍的表面所用的代价都是一样的。

任务是引导到某目标状态 x_G。方法是通过初始时在包含 x_I 和 x_G 的 X 的一个子集上构造一个反馈规划 π 而进行的。规划 π 的计算是通过迭代应用图 12.18 中的步骤直到得到 x_I

上的最优尚需代价。像 Dijkstra 算法一样需要维持一个优先级队列 Q，但是由于在执行过程中感测到信息，Stentz 算法允许修改 Q 中元素的代价。令 $G_{best}(x)$ 表示 Q 中在所用时间期间 x 的最低尚需代价。假设根据 $G_{best}(x)$ 对 Q 进行排序。令 $G_{cur}(x)$ 表示目前的尚需代价值，如果某代价更新引起它增加，它可能会比 $G_{best}(x)$ 大。假设应用某 $u \in U(x)$ 到达一个状态 $x' = f(x,u)$。令 $G_{via}(x,x')$ 表示从 x 运动 x' 所用的尚需代价：

$$G_{via}(x,x') = G_{cur}(x') + l(x,u) \tag{12.24}$$

如果 $G_{via}(x,x') < G_{cur}(x)$，那么表示 $G_{cur}(x)$ 可以减小。如果 $G_{cur}(x') \leqslant G_{best}(x)$，那么进一步知道 $G_{cur}(x')$ 是最优的。如果这两个条件都满足，那么 $G_{cur}(x)$ 将被更新到 $G_{via}(x,x')$。

STENTZ 算法：

（1）从 Q 删除 x，该状态的 $G_{best}(x)$ 值最低。

（2）如果 $G_{best}(x) < G_{cur}(x)$，那么在 Q 中 x 的值已经增加了。如果 x 能够通过经过一个最优尚需代价已知的邻近状态来改进它的代价，那么它将这样做。因此对于每个 $u \in U(x)$，检验 $x' = f(x,u)$ 是否有 $G_{via}(x,x') < G_{cur}(x)$ 且 $G_{cur}(x') \leqslant G_{best}(x)$。如果是这样，那么更新 $G_{cur}(x) := G_{via}(x,x')$ 和 $\pi(x) := u$。

（3）本步骤和余下的步骤对 x' 重复进行使得存在 $u' \in U(x')$，对于该 u' 有 $x = f(x',u')$。如果 x' 未被访问过，那么赋予 $\pi(x') := u'$，并将 x' 及其代价 $G_{via}(x',x)$ 放入 Q。

（4）如果由于 $\pi(x') = u'$ 但是 $G_{via}(x',x) \neq G_{cur}(x')$ 的缘故，x' 的尚需代价出现不正确，那么需要一次更新，并且将 x' 及其代价 $G_{via}(x',x)$ 放入 Q。

（5）如果 $\pi(x') \neq u'$ 但是 $G_{via}(x',x) < G_{cur}(x')$，那么从 x' 经过 x 比用 $\pi(x')$ 好。如果 $G_{cur}(x) = G_{best}(x)$，那么将 $\pi(x') := u'$ 和 x' 插入 Q，因为 x 的最优尚需代价已经知道。否则应该将 x（而不是 x'）及其目前的值 $G_{cur}(x)$ 插入 Q。

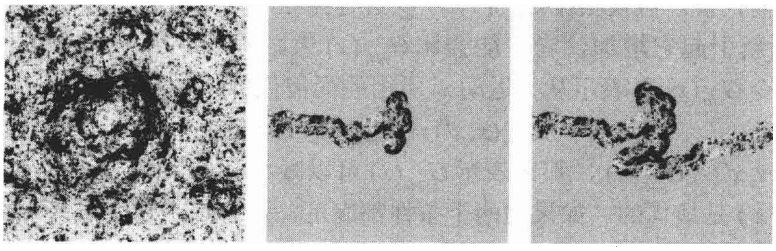
（6）需要最后一个条件来避免在 π 中产生循环。如果 x' 是不活动的（已经访问过，不再在 Q 中），那么需要将它及其代价 $G_{cur}(x')$ 重新插入 Q。必须这样做，如果 $\pi(x') \neq u'$，$G_{via}(x,x') < G_{cur}(x)$ 且 $G_{cur}(x) > G_{best}(x)$。

图 12.18　Stentz 算法通常被称为 D^*，它是 Dijkstra 算法的一种变形，在执行过程中当获悉代价项时，它动态地更新代价值；这里的步骤只是从 Q 中删除一个状态后更新代价的一次迭代

图 12.18 的迭代过程完成以后，机器人执行 π，这将产生访问过的状态的序列。令 x_k 表示在执行过程中目前的状态。若发现：如果应用 $\pi(x_k) = u_k$，在目前模型下所产生的代价与代价 $l(x_k,u_k)$ 不匹配，那么代价需要进行更新。更一般情况下，在对应于传感器作用范围的 x_k 附近区域内机器人必须能够对代价进行更新。对于下面的描述，假设只得到 x_k 的一个更新 $l(x_k,u_k)$（更一般的情况也可以用相似的方法进行处理）。首先 $l(x_k,u_k)$ 更新到新的测量值。如果 x_k 恰好是不活动的（已经访问过，但是不再在 Q 中），那么将它再次插入到 Q，其代价为 $G_{cur}(x_k)$。执行图 12.18 中的步骤直到对于所有 $x \in Q$，有 $G_{cur}(x_k) \leqslant G_{best}(x)$。接着，继续执行规划直到到达目标或者发现另外一个代价不匹配。在执行过程中的任何时刻，在给定关于代价的目前信息的情况下，机器人运动是最优的[914]。

图 12.19 阐述了算法的执行过程。图 12.19（a）显示了一个用随机分形方法生成的合成地形。更黑的灰度值表示更高的代价。在中心处，代价很高的地形被看作障碍物，避开障碍物的路线是向下方向。初始状态在环境左边缘的中央，目标状态是环境的右边缘。机器人初始时规划了一条直线路径，然后在运动过程中递增地更新每一步路径。在图 12.19（b）中，机器人已经遇到代价高的中心区域并开始对其周围进行搜索。最后到达目标，如

图 12.19（c）所示。所执行的路径实际上是执行一系列最优路径后的结果，每条最优路径都是基于应用单个行动时的已知信息。



(a) (b) (c)

图 12.19 执行 Stentz 算法的一个例子（由 Tony Stentz 提供）

用 *I*-空间进行阐述 为了加深对环境集合的 *I*-空间的理解，现在给出另外一种表述。像以前一样，将状态空间定义为一个位形空间 $C=\mathbb{Z}\times\mathbb{Z}$。令 $q\in C$ 表示一个位形。假设每个可能的环境对应于一种给位形转移图中所有边赋予代价的方法。该问题的所有可能环境的集合 E 似乎是赋予代价 $l(q,u)$ 的所有可能方法。现在可以定义状态空间为 $C\times E$，并且对于每个状态 $x=(q,e)\in X$，指定位形和代价的完备集合。初始情况下，猜测机器人处在某特定 $e\in E$。如果发现代价不匹配，这意味着现在呈现不同的环境模型，因为转移代价不同于所预料的代价。代价实际上应该被记为 $l(x,u)=l(q,e,u)$，即代价依赖于特定环境。

非确定的 *I*-状态对应于可能赋值代价的一个集合及其相应的位形。因为该方法要求赋值还没有观测到的代价，那么它进行猜测并假设非确定的 *I*-状态中的一个特定环境是正确的。当发现代价不匹配时，那么它意识到以前的猜测处在更新的非确定的 *I*-状态以外。因此修正猜测以融合新的代价信息。当过程进行时，非确定的 *I*-状态继续收缩，但是注意到：即使对 $e\in E$ 的猜测到最后还是不正确的，机器人依然可能解决该问题。某些格片从来没有被访问过，所以它们的真正代价是未知的。为了解决该问题，对它们的代价有一个默认假设，但是真正的 $e\in E$ 只在所有格片都被访问过后才能够知道。事实是最后的默认假设值处在最后的非确定的 *I*-状态内。

12.3.3 未知连续环境中的规划

我们现在从离散环境转移到连续环境，但是还继续使用非确定的不确定性。首先提出几个 bug 算法[507,670,508]，它们代表了一族运动规划，其思路与第 12.3.1 节的探测迷宫的思路有某种联系。除了 bug 算法以外，我们还将简要介绍竞争率的概念。

bug 算法将会使用接下来的模型。假设将一个点机器人放置在一个未知的二维环境中，该环境可能包括任意确定数量的有界障碍物。假设每个障碍物的边界和外部边界（如果存在）都是分段解析的（解析（analytic）是指每段都是平滑的，它的曲率符号只有有限次切换）。因此障碍物可能是多边形、平滑曲线或者曲线与直线的某种组合。虽然可能环境的集合 E 是势不可挡的，但是将避免对它进行明确的构造。机器人位形可用位置和方向进行刻画。

有两种主要的传感器[1]：

（1）目标传感器：它指示与目标之间的欧氏距离以及方向，方向是相对于正北方向而言。

（2）局部可视性传感器：它提供在距离机器人近距离范围内边界的精确形状。机器人必须接触或者几乎接触到边界以观测其一部分；否则传感器无法提供有用信息。

目标传感器本质上是在极坐标系中对机器人的位置进行编码（目标在原点）。因此可以给机器人访问过的任何一个位置赋予一个独特的 (x,y) 坐标。这使它能够逐步描绘出它已经过的障碍物的边界。局部可视性传感器提供的信息只够进行顺墙运动；传感器的作用距离非常小，所以机器人不能获悉关于环境结构的更多信息。

现在为机器人考虑某些策略。每个策略都可以被认为是一个非确定的 I-空间上的一个信息反馈规划。

Bug1 策略　文献[670]提出了一种被称为 Bug1 的策略，如图 12.20 所示。其执行过程如下：

（1）向目标移动直到遇到障碍物或者到达目标。如果到达目标，那么过程停止。

（2）向左转，沿着所接触到的障碍物的周边行进。一旦访问了障碍物的整个周长，那么回到离目标最近的点，然后返回步骤（1）。

确定已经经过障碍物的整个周长这个过程需要进行标记，这可以通过发现目标传感器读数重复的点来进行推断。

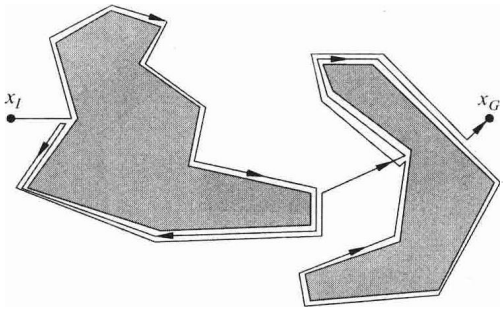


图 12.20　阐述 Bug1 策略

最坏情况从概念上很容易理解。机器人行进的整个距离不大于：

$$d+\frac{3}{2}\sum_{i=1}^{M}p_i \tag{12.25}$$

其中 d 是从初始位置到目标位置的欧氏距离，p_i 是第 i 个障碍物的周长，M 是障碍物的数量。这意味着经过每个障碍物边界的次数不会超过 3/2。图 12.21 显示了一个例子，其中每个障碍物经过了 3/2 次。该界限依赖于这样一个事实：机器人总能够回忆起沿着边界、距离它需要离开的点的最短路径。这似乎是合理的，因为机器人能够用传感器推断出沿着边界行进的距离。如果这是不可能的，那么将不得不用 2 来代替 3/2，因为在最坏情况下机器人几乎要行进整个边界两次。

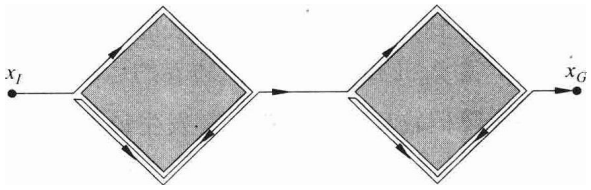


图 12.21　Bug1 情况的一个不好的例子；经过每个障碍物的周长 1.5 次

1　这只是可能的感测模型。也可以用多个传感器的组合，只要它们使能在运动策略中所要求的运动和决策。

Bug2 策略 除 Bug1 策略以外还有一种选择就是 Bug2 策略，如图 12.22 所示。机器人总是试图沿着连接初始位置和目标位置的直线进行运动。当机器人在该直线上的时候，目标方向将与初始时相同，或者相差 π 弧度（如果机器人在目标的另外一侧）。第一步与 Bug1 相同。第二步机器人沿着周长行进直到到达直线，并沿着朝向目标的方向行进。从那里它又回到步骤（1）。这有可能发生无限循环。因此需要进行一些修改。如果机器人检测到它将要离开已经访问过的边界上的点，那么它继续沿着边界行进直到到达它没有访问过的直线上的点为止。从该点，它离开直线并返回到步骤（1）。

对于 Bug2 策略，整个行进的距离不会超过：

$$d+\frac{3}{2}\sum_{i=1}^{M}n_i p_i \tag{12.26}$$

其中 n_i 是第 i 个障碍物与初始位置和目标位置之间线段相交的次数。图 12.23 阐述了由于相交而引起的问题。

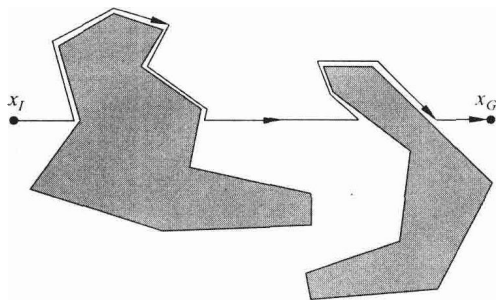


图 12.22 阐述 Bug2 策略

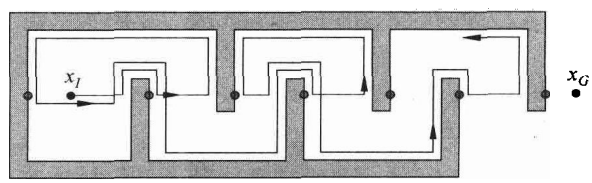


图 12.23 Bug2 情况的一个不好的例子；它只产生所示的路径；图中指出了机器人从哪些点可以离开边界

使用测距数据 VisBug[669]和 TangentBug[508,595]策略融合了测距传感器和可视性传感器的距离测量数据来提高效率。这里将对 TangentBug 策略进行描述，如图 12.24 所述。假设除了以前所描述的传感器以外，它还配有进行如图 12.25 所示的测量的传感器。策略如下：

（1）穿过空间的内部或者顺着墙向目标移动，直到意识到机器人陷入一个局部最小值或者达到目标。这与第 5.4.3 节的势场规划器的梯度下降运动相似。如果到达目标，那么过程停止；否则进行下一步。

（2）沿着边界进行运动。首先通过将以前的航向与目标方向进行比较来选择一个方向。当沿着边界运动时，跟踪两个距离：d_f 和 d_r。d_f 是距离目标的最小距离，它是沿着边界运动时观测的。d_r 是从目前位置到目标的最短路径的长度，假设对测距传感器来说只有障碍物是可视的。如果 $d_r < d_f$，那么机器人会停止沿边界的运动。在这种情况下，返回到步骤（1）。如果机器人围绕整个障碍物环形运动都没有发生该条件，那么算法会报告目标是无法到达的。

为测距传感器设置一个深度限制就可以产生 TangentBug 算法的一个参数族。当最大深度减小时，机器人会变得更加近视，性能也会下降。文献[508]指出，行进的距离不会大于：

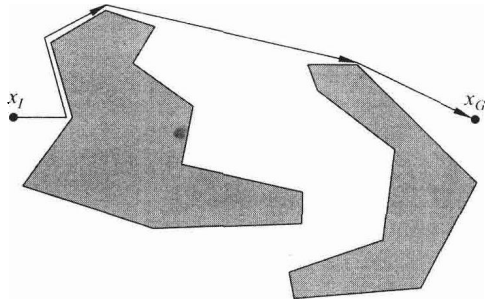


图 12.24　对半径不受限的 VisBug 策略的一种阐述说明

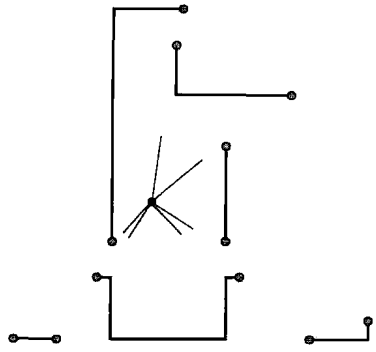


图 12.25　关于测距传感器的候选运动是在深度图中存在跳变的方向上；从机器人到小圆圈之间的距离用来选择所期望的运动

$$d+\sum_{i=1}^{M} p_i+\sum_{i=1}^{M} p_i m_i \tag{12.27}$$

其中 m_i 是对于第 i 个障碍物来说局部最小值的数量，d 是距离目标的初始距离。从 M 个障碍物中选取界限，假设它们与以 d 为半径、以目标为中心的圆相交。文献[595]为受限视场的星际飞行器而提出了 TangentBug 的一种变化形式，称为 WedgeBug 算法。

竞争率　在算法领域已经出现对利用不同信息的算法进行评估的一种普遍方法。思路是计算竞争率（competitive ratio），它将一个在线算法与接收更多信息的算法进行竞争[677,893]。该思路通常可以应用于规划。首先表述一个代价，例如机器人为了解决导航任务而行进的总距离。然后定义竞争率：

$$\max_{e\in E}\frac{\text{执行提前不知道 } e \text{ 情况下的规划的代价}}{\text{执行提前知道 } e \text{ 情况下的规划的代价}} \tag{12.28}$$

最大值在所有 $e\in E$ 上选取，就像 bug 算法情况中那样，$e\in E$ 通常是一个无限集合。将最优距离和机器人执行在线算法过程中所行进的总距离进行比较就可以产生导航问题的竞争率。因为 E 是无限的，所以一些算法不能产生有限竞争率。bug 算法就是这种情况的一个例子。假设一个目标，它距离我们非常近，但是需要探测一个大的障碍边界。在使得离目标的最优距离非常小的同时，可以使得障碍边界任意大。当用式（12.28）进行计算时，在所有环境上的结果是上无界的。在某些背景下，如果用一个函数来表达，竞争率还是非常有用的。例如，如果 E 是有 n 条边的多边形，那么竞争率 $O(\sqrt{n})$ 意味着式（12.28）在所有 n 上以 $c\sqrt{n}$ 为界，对于某 $c\in\mathbb{R}$。在 bug 算法中竞争率的分析见文献[377]。

lost-cow 问题提供了对竞争率分析和竞争率问题的一个非常好的阐述。如图 12.26（a）所示，近视的母牛顺着一个无限长的栅栏行进想要找到门。这就产生了一个一维规划问题。如果给定门的位置，那么母牛可以直接到达它。如果该母牛被告知门就在距离 1 以外，那么它在一个方向上行进一个单位，如果没有发现门，它转向其他方向。这种情况下的竞争率是 3（环境的集合对应于门的布置）。如果母牛被告知门至少在距离 1 以外，会怎样？在这种情况下，最好的策略是螺旋搜索，它将“之”字形前后进行搜索，在每个方向上加倍行进的距离，如图 12.26（b）所示。也就是说：向左一个单位，向右一个单位，向左两个单位，向右两个单位，向左四个单位，如此继续。这种策略的竞争率是 9，它是最优的。

这种方法类似于第 2.2.2 节中的迭代深化方法。

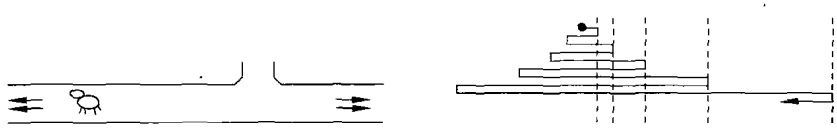


图 12.26 （a）一头迷路的母牛必须发现通向大门的路，但是它不知道门在哪个方向；（b）如果没有距离的限制，那么双螺旋策略能够很好地解决这个问题，它产生的竞争率为 9

12.3.4 没有几何模型情况下的最优导航

本节提出缺口导航树（gap navigation tree，GNT）[943,945]，它是一个数据结构，并且是与实现第 12.3.3 节所考虑的连续环境中的最优导航有关的规划算法。本节假设机器人配有一个缺口传感器，如图 11.16 所示。在每个时刻，机器人对每个缺口都有一个可用的行动。如果应用一个行动，那么机器人向相应的缺口方向行进。这可以连续进行使得机器人能够“追逐”一个特定的缺口。机器人没有其他感测信息：它没有罗盘，也不能测量距离。因此它不可能构造一个包含度量信息的环境地图。

假设将机器人放置在一个未知但单连通的平面环境 X 中。GNT 也可以扩展到多连通环境，但是在多连通环境下，有一些关于可分辨性的细小问题，并且只可能在路径的一个同伦类中保证最优性[944]。通过分析临界事件在缺口传感器中发生的方式就可以构造一个树的表示，它指示在环境中如何进行最优运动，即使没有精确地测量信息。因为缺口传感器不能测量距离，机器人在没有收到距离（或者度量）信息的情况下仍然能够沿着最短路径运动，这似乎是很不寻常的。这又一次显示了 I-空间的强大功能。

相对于机器人位置的环境被编码为一棵树，它指示当机器人运动时，缺口是如何变化的。它给机器人提供足够的信息使之沿着最短路径向环境的任何位置运动。树并不对应于环境的某静态地图，理解这一点是非常重要的。它表达了相对于机器人，环境是如何出现的，因此当机器人在环境中运动时，树也可能发生改变。

树的根表示缺口传感器。对于出现在传感器中的每个缺口，将有一条边连接到根。令这些边为根边（root edges）。每条根边对应于机器人可以应用的一个行动。通过选择一条根边，那么行动使机器人沿着朝向缺口的直线运动。因此有一个简单的控制模型使得机器人沿着边界 ∂X 精确地朝一个特定点运动，如图 12.27 所示。

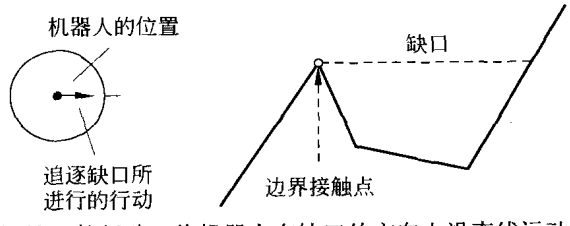


图 12.27 应用追逐缺口的行动，将机器人在缺口的方向上沿直线运动直到接触到边界；一旦接触到边界，环境的一个新的部分就是可视的了

令 $V(x)$ 表示可视区域（visibility region），它是 X 中对 x 可视的所有点的集合。令 $X \setminus V(x)$

为阴影区（shadow region），它是 x 不可视的所有点的集合。令阴影的每个连通分支为阴影分支。缺口传感器中的每个缺口对应于 X 中的一条线段，该线段在两个地方接触到 ∂X（见图 11.15（a））。每条线段形成了可视区域和一个阴影分支之间的一个边界。如果机器人愿意向阴影分支行进，那么最短的路线是直接向缺口运动。当向缺口运动时，机器人最终会到达 ∂X，在该点上必须选择一个新的行动。

临界缺口事件　当机器人运动时，在缺口传感器中会发生几个重要的事件：

（1）消失：一个缺口消失，因为机器人穿过了一条拐切射线，如图 12.28 所示。这意味着某些以前的阴影分支现在是可视的。

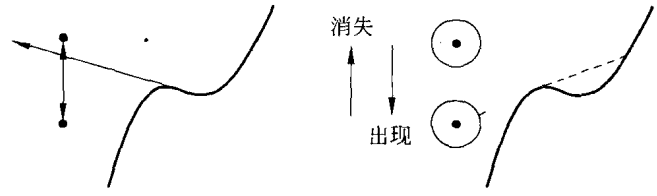


图 12.28　（a）机器人穿越一条从拐切线延伸过来的射线；（b）一个缺口在缺口传感器中出现或者消失，这依赖于方向

（2）出现：一个缺口出现，因为机器人沿相反方向穿过了一条拐切射线。这意味着出现了新的阴影分支，它表示环境中刚刚隐藏起来的部分。

（3）分裂：一个缺口分裂为两个缺口，因为机器人穿过了一个双切射线，如图 12.29 所示（也可见图 12.5）。这意味着一个阴影分支分裂成两个阴影分支。

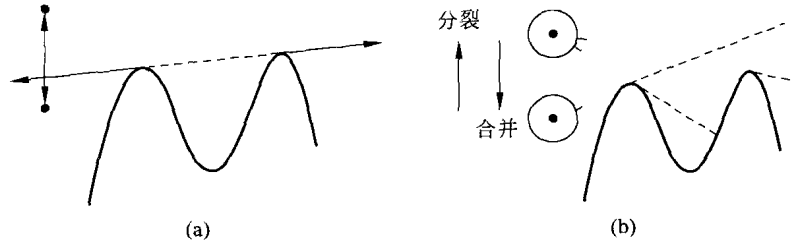


图 12.29　（a）机器人穿越一条从双切线延伸过来的射线；（b）缺口分裂或者合并依赖于方向

（4）合并：两个缺口合并成一个，因为机器人沿相反方向穿过了一个双切射线。在这种情况下，两个阴影分支合并成一个阴影分支。

在最广位置假设下，这是可能发生的事件的完整列表，该假设排除了引起退化的环境，如将三个缺口合并成一个，或者一个缺口准确地出现在其他两个缺口分裂的地方。

当这些缺口事件发生时，需要在树中有所反映。如果一个缺口消失，如图 12.30 所示，那么相应的边和顶点将被删除。如果发生合并，那么插入一个中间顶点，如图 12.31 所示。这说明：如果追逐一个缺口，那么它将分裂为两个新的缺口。如果分裂发生，如图 12.32 所示，那么可删除中间顶点。缺口的出现是一个重要情况，它在树中产生一个**本原顶点**（primitive vertex），如图 12.33 所示。本原顶点从来不会发生分裂，因为追逐它将导致它的消失。

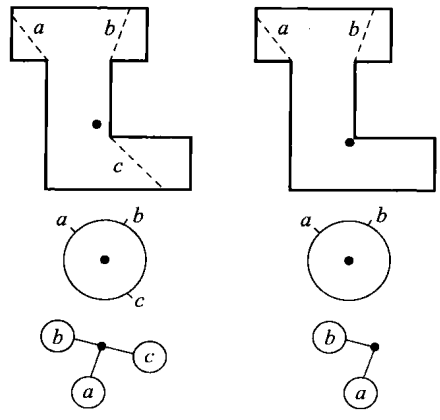


图 12.30 如果一个缺口消失，那么应从 GNT 中删除它

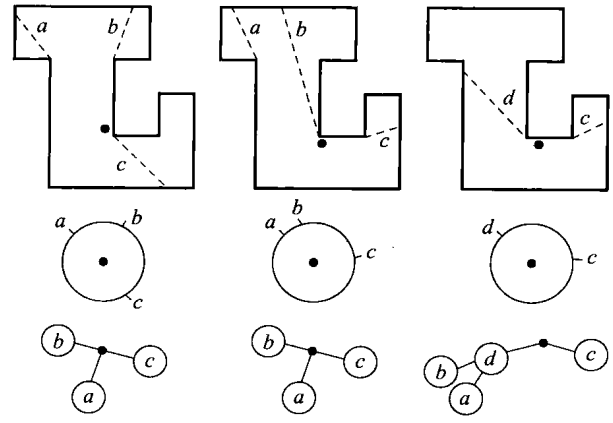


图 12.31 如果两个缺口合并，那么一个中间顶点将被插入树中

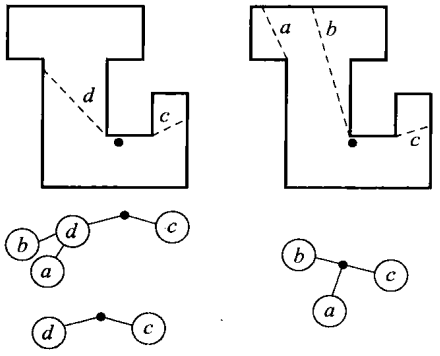


图 12.32 如果两个缺口分裂，那么删除中间顶点

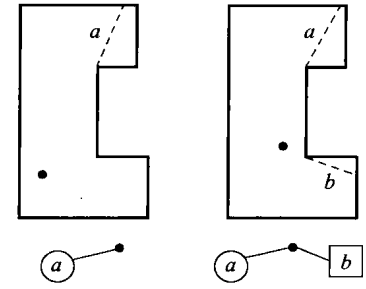


图 12.33 一个缺口的出现导致了一个本原顶点，它可用一个方块表示

现在考虑一个简单例子。

例 12.6（缺口导航树） 假设机器人不知道图 12.34 中的环境。它从 1 胞腔顺序地向 7 胞腔运动，然后返回 1 胞腔。产生树的序列：$T_1,\cdots,T_7,T_6',\cdots,T_1'$，如图 12.35 所示。根顶点显示为一个实心黑圆点。不知道是本原的那些顶点显示为一个圆形；本原顶点是方形。注意到如果任何叶子顶点是一个圆，那么意味着被缺口隐藏的 R 的阴影区域还没有被完全探测。一旦机器人到达 5 胞腔，那么它已经看见整个环境。这发生在当所有叶子顶点都是本原时。当机器人返回到第一个区域时树会更大，因为它知道右边的区域由两个更小的区域组成。如果所有叶子都是方形的，这意味着环境已经被完全探测过了。■

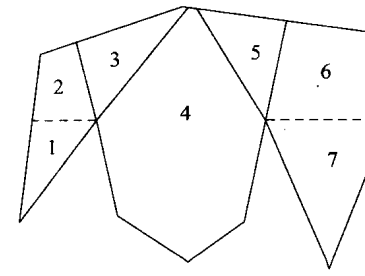


图 12.34 阐述缺口导航树的一个简单环境

在这个例子中，环境的所有感兴趣的部分都被探测过了。从此向前，所有叶子顶点都将是本原顶点，因为所有可能的分裂已经被发现。从某种意义上，已经用缺口传感器在分辨率的层次上完全获悉了环境信息。探测环境的一个简单策略

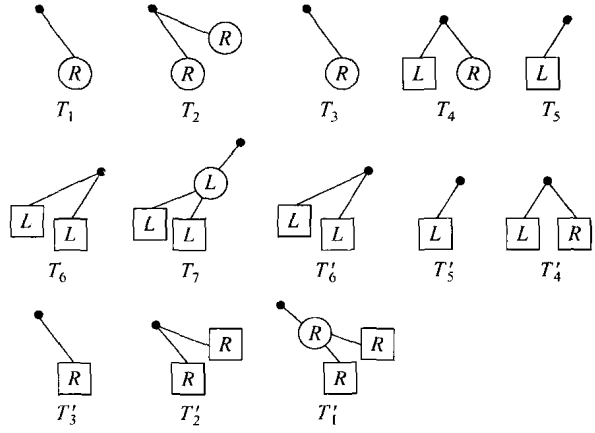


图 12.35　用缺口导航树来建立图 12.34 中环境的表示；顺序是从左至右；为方便起见，每个顶点内的“R”或者“L”分别指示阴影分支是在缺口的右边还是左边；算法不需要这种信息，但是它有助于人们对表示的理解

是追逐任何自身是非本原叶子顶点的缺口，或者它的孩子是非本原叶子顶点的缺口。重复应用追逐缺口传感器中相应缺口的行动就可以追逐树中的一个叶子顶点。这可能引起树递增地变化；但是如果选择行动是用来追逐哪个缺口隐藏了想要的顶点，那么没有任何问题，如图 12.36 所示。每个非本原叶子顶点将会分裂或者消失。追逐过所有非本原叶子顶点后，就进行了所有可能的分裂，只有本原叶子保留下来。在这种情况下，环境已经完全清楚了。

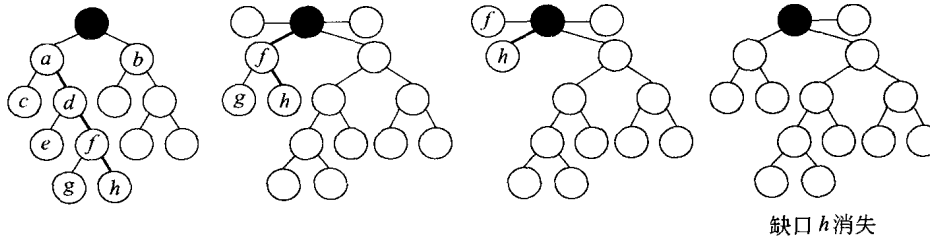


图 12.36　通过“追逐”GNT 中所期望的顶点直到它消失，可以实现到环境的特定部分的最优导航

用 GNT 进行最优导航　因为没有环境的精确地图，所以不可能用 $\mathbb{R}^2$ 中的坐标来表达目标状态。但是目标可以用那些必须追逐以使状态可视的顶点来表达。例如，假设当机器人探测环境时，给机器人显示一个目标。开始时目标是可视的，但是可能会出现一个隐藏目标的缺口。经过几次合并以后，树的深处的顶点可能对应于可以看见目标的位置。通过追逐第一次出现就隐藏目标的顶点，机器人可以最优地导向目标。一旦这种顶点及其相应的缺口消失，目标就可视了。这时机器人可以直接向目标运动（假设有另外一个传感器指示目标的方向）。文献[945]指出当机器人追逐GNT中的一个顶点时，它将精确地跟踪第6.2.4节中介绍的最短路径路线图。每对相继缺口事件对应于穿越一条双切线。

***I*-空间阐述**　根据环境集合上的 *I*-空间，GNT 认为多个环境的大的集合是等价的。这意味着构造一个 *I*-映射，该映射上的衍生 *I*-空间是可能 GNT 的集合。在这种 *I*-映射下，一些环境对应于相同的 GNT。由于此，机器人能够完成任务而并不需要更进一步的信息。例如如果两个环境只由于旋转或者比例而不同，那么 GNT 的表示是相同的。令人惊奇的是，

机器人甚至不需要关心环境的边界究竟是多边形的还是曲线的。而只关心缺口事件是如何发生的。例如图 12.37 中的环境都产生相同的 GNT，因此它们对于机器人来说是不可分辨的。同样地，第 12.3.1 节的迷宫探测算法不需要一张完整的地图来定位目标，GNT 也不需要一张完整的地图来进行最优导航。

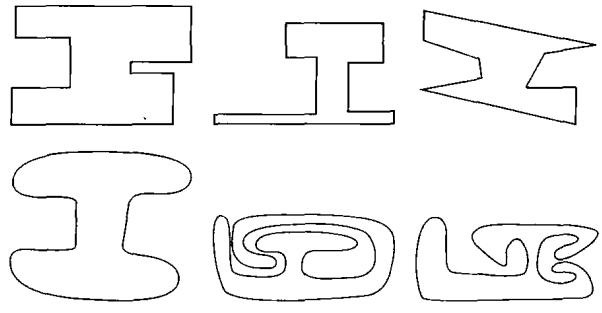


图 12.37 这些环境产生相同的 GNT，因此在衍生 *I*-空间的分辨率上它们是等价的；机器人不能测量距离，甚至也不关心墙究竟是直的还是弯曲的，这与导航任务无关；尽管如此，机器人执行用欧氏距离表示的最优运动

12.3.5 概率定位与制图

到目前为止第 12.3 节中的问题都避开了概率模型。这里假设状态转移和观测中存在概率模型。一些问题的表述可以用概率模型代替第 12.3.1 节中的非确定的模型。这将产生概率 *I*-状态，它们表示可能栅格集合上的分布以及每个栅格内的位形。如果问题处在它的最一般的情况下，那么 *I*-空间过于庞大，用目前的方式无法处理它。如果不要求最优性，那么某些特殊的情况下，有可能进行改进提高。

目前的问题是构造环境的地图，同时关于地图对机器人进行定位。请回忆图 1.7。本节介绍近年来在可移动机器人领域中广泛使用的一般框架。这里所进行的讨论可以认为是对第 12.2.3 节内容的推广，在那里只关心目前问题的定位。现在环境是未知的。目前的问题可以阐述为在如下定义的状态空间中的定位：

$$X = \mathcal{C} \times E \tag{12.29}$$

其中 $\mathcal{C}$ 是一个位形空间，E 是一个环境空间。状态 x_k 表示为 $x_k = (q_k, e)$，e 没有下标 k 是因为假设环境是静态的。历史 *I*-状态提供在状态确定过程中所使用的数据。就像第 12.2 节的定位一样，问题也有主动和被动之分。主动问题的一个递增版本有时被称为 next-best-view 问题[67,240,796]。困难在于机器人有相反的目标：(1) 试图在将会得到尽可能多的新数据的地方打开传感器；(2) 冗余的最小化可能使得很难将所有测量合并到一个全局地图。这里将描述被动问题，这些方法可以为解决主动问题提供信息。

假设机器人是一个在 $\mathbb{R}^2$ 中平移和旋转的点。按照第 4.2 节所讲述的，这就产生 $\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1$，它表示 $SE(2)$。令 $q \in \mathcal{C}$ 表示一个位形，它产生机器人的位置和方向。假设位形的转移用概率来建模，那么需要指定概率密度 $p(q_{k+1} | q_k, u_k)$。如果假设位形的转移与环境无关（假设没有碰撞发生），那么概率密度可以提升至状态空间来得到 $p(x_{k+1} | x_k, u_k)$。用 x_k 和 x_{k+1} 分别代替 q_k 和 q_{k+1}，其中对于任何 $e \in E$，有 $x_k = (q_k, e)$ 和 $x_{k+1} = (q_{k+1}, e)$。

假设观测来自于一个深度传感器，它在理想情况下产生像图 11.15（b）那样的测量数据，但是假设数据是有噪声的。第 12.2.3 节中讨论的概率模型可以用来定义 $p(y|x)$。现在假设机器人向环境的几个部分运动，就像图 11.15（a）所示的那样，机器人对每个地方都进行一次传感器扫描。如果进行扫描 y_k 的位形 q_k 未知，那么数据集合如何组合在一起来建立环境的一个正确的全局地图？在知道位形的情况下这非常容易，但是在不知道位形的情况下，问题就像智力拼图游戏一样。因此每个阶段的重要数据就形成了一个向量 (y_k, q_k)。如果传感器观测 y_k 没有标记于一个可以产生这些观测的位形 q_k，那么就会产生拼图问题。如果信息用来约束 q_k 的可能性，那么将这些数据拼在一起就容易得多。这种直觉就产生了如下　方法。

EM 算法　实际上问题的解决通常是采用最大化期望值算法（expectation-maximization，EM）[108]。在一般框架中有三种不同空间：

（1）通过某种测量和估计过程所确定的一个参数集合。在我们的问题中，这表示 E，因为主要的目标是确定环境。

（2）一个数据集合，它提供能够用来估计参数的信息。在定位和制图问题中，这对应于历史 I-空间 $\mathcal{I}_K$。每个历史 I-状态 $\eta_K \in \mathcal{I}_K$ 为 $\eta_K = (p(x), \tilde{u}_{K-1}, \tilde{y}_K)$，其中 $p(x)$ 是 X 上的一个先验概率密度。

（3）隐藏变量的集合，它们是未知的，但是需要对它们进行估计来完成确定参数的过程。在定位和建图问题中，这就是位形空间 C。

因为参数和隐藏变量都是未知的，所以对它们两者的选择似乎是随意的。将会看到可以产生能够很好地表达隐藏变量的概率密度的表达式，但是参数更加复杂。

EM 算法涉及一个求期望的步骤和一个求最大值的步骤。这两个步骤重复进行直到得到满足精度的解为止。方法保证在一般条件下收敛[271,977,978]。实际上甚至在无法理论上保证收敛的情况下它也能正常工作[940]。

令 E、$\mathcal{I}_K$ 和 C 表示 EM 算法的三个空间，因为它们直接与问题有关。假设机器人已经在环境中运动了 $K-1$ 个阶段，导致了最后一步 K。在每个阶段 $k \in \{1, \cdots, K\}$，用传感器得到一个观测 y_k。例如这可以代表用声纳或者测距扫描仪所得到的距离测量的一个集合。而且，从 $k=1$ 到 K 都分别应用行动 u_k。先验概率密度函数 $p(x)$ 初始时假设在 X 上。这产生式（11.14）所定义的历史 I-状态 η_k。

现在假设已经执行了 K 个阶段，任务是估计 e。从每个 q_k 得到部分环境的一个测量 y_k。EM 算法生成 e 的一个改进的估计序列。在两步 EM 的每次执行过程中，产生 $e \in E$ 的一个新的估计。令 $\hat{e}_i$ 表示第 i 次迭代后的估计。令 $\tilde{q}_K$ 表示 1～K 的位形历史。给定 $\hat{e}_i$，求期望的步骤计算 η_K 的期望似然函数。这可以表示为[1]：

$$Q(e, \hat{e}_{i-1}) = E\left[p(\eta_K, \tilde{q}_K \mid e) \middle| \eta_K, \hat{e}_{i-1} \right] = \int_C p(\eta_K, \tilde{q}_K \mid e) p(\tilde{q}_K \mid \eta_K, \hat{e}_{i-1}) d\tilde{q}_K \qquad (12.30)$$

其中期望采用位形历史。因为给定 η_K 并且在期望中消除了 $\tilde{q}_k$，所以式（12.30）只是 e 和 $\hat{e}_{i-1}$ 的函数。$p(\eta_K, \tilde{q}_K \mid e)$ 表示为：

1　实际上，对 $p(\eta_K, \tilde{q}_K \mid e)$ 采用对数形式，因为通常会出现包含指数的密度。采用对数形式在不影响优化结果的情况下使得表达更简单。这里没有用对数，因为还没有详细到这个程度。

$$p(\eta_K, \tilde{q}_K \mid e) = p(\tilde{q}_K \mid \eta_K, e) p(\eta_K \mid e) \tag{12.31}$$

其中 $p(\eta_K)$ 是 I-空间上的一个先验密度，只是需要给定环境 e。因式 $p(\tilde{q}_K \mid \eta_K, e)$ 与式(12.30)中积分的第二个因式的不同之处只在于 e 或者 $\hat{e}_{i-1}$。计算式（12.30）的主要困难在于计算 $p(\tilde{q}_K \mid \eta_K, \hat{e}_{i-1})$（或者使用 e 的版本 $p(\tilde{q}_K \mid \eta_K, e)$）。这在本质上是第 12.2.3 节给定地图的定位问题。到 k 阶段为止的信息可以用来为每个 q_k 产生概率 I-状态 $p(q_k \mid \eta_k, \hat{e}_{i-1})$，但是这却忽略了来自于其余阶段的信息。这些新的信息可以用来对旧的位形进行估计。例如基于目前的测量和对所应用的行动的记忆，我们有更好的关于几个阶段以前的位形的信息。在文献[941]中给出了一种计算 $p(q_k \mid \eta_k, \hat{e}_{i-1})$ 的方法，它包括两项：一项是 $p(q_k \mid \eta_k)$；另外一项是从 K 开始到 $k+1$ 的后向概率 I-状态。

一旦确定概率密度，那么式（12.30）只是 e 和 $\hat{e}_{i-1}$ 的函数。求最大值涉及选择一个 $\hat{e}_i$ 来最大化式（12.30）：

$$\hat{e}_i = \arg\max_{e \in E} Q(e, \hat{e}_{i-1}) \tag{12.32}$$

进行优化通常是很困难的，并且如果选择 $\hat{e}_i$ 使得 $Q(\hat{e}_i, \hat{e}_{i-1}) > Q(\hat{e}_{i-1}, \hat{e}_{i-1})$，那么存在收敛条件。重复迭代 EM 算法就产生了一种到达 E 中局部最小值的梯度下降。

该方法成功的一个重要因素在于 E 的表示。在 EM 的计算中通常使用一个地标集合，这在第 11.5.1 节提到过。它们是环境中能够被传感器识别的一些特殊位置，并且如果对它们进行正确分类，那么它们对定位将有显著的提高。在文献[941]中，当机器人运动时，地标由用户指示。分类和定位误差都可以用概率建模并融入 EM 方法。显著简化 E 的表示的另外一个思路是用更细分辨率的栅格来对环境进行近似。概率与栅格胞腔有关，这导致了一种被称为栅格占有率（occupancy grid）的数据结构[309,688,852]。在任何情况下，必须仔细定义 E 以确保当机器人开始运动时能够为 $p(e)$ 构造合理的先验分布来初始化 EM 算法。

12.4 基于可视性的追-逃问题

本节考虑基于可视性的追逃问题（visibility-based pursuit-evasion）[615,932]，它在第 1.2 节是作为一个捉迷藏游戏被介绍的。该论题对 I-空间概念的作用进行了非常出色的阐述。

12.4.1 问题的表述

本节考虑的问题表述如下。

表述 12.1（基于可视性的追逃问题）

1. 一个给定的连续环境区域 $R \subset \mathbb{R}^2$，它是一个以一条简单封闭曲线为界的开集合。边界 ∂R 通常是多边形，但是它可能是任意分段解析的封闭曲线。

2. 一个无界的时间区间 $T = [0, \infty)$。

3. 一个逃跑者，它是 R 中的一个运动点。逃跑者在时刻 $t \in T$ 的位置 $e(t)$ 由连续位置函

数来确定 $\tilde{e}:[0,1]\to R$。[1]

4. 一个追赶者，它是 R 中的一个运动点。逃跑者的位置函数 $\tilde{e}$ 对追赶者来说是未知的。

5. 一个可视性传感器，对于每个 $r\in R$，该传感器定义了一个集合 $V(r)\subseteq R$。

任务是为追赶者寻找一条路径 $\tilde{p}:[0,1]\to R$，使得保证能够检测到逃跑者，无论逃跑者的位置函数如何。这意味着 $\exists t\in T$ 使得 $e(t)\in V(p(t))$。追赶者的速度并不重要，因此如果追赶者速度慢，那么时间域可能需要加长。

通过证实没有逃跑者来解决问题将是很方便的。也就是说，追赶者寻找一条路径以确保没有其他位置能够隐藏逃跑者。这使得规划在执行过程中，追赶者将遇到任何逃跑者。实际上可能有任意数量的逃跑者，追赶者将会发现所有这些逃跑者。方法系统地排除了能够隐藏逃跑者的任何可能的位置。

状态包括追赶者和逃跑者的位置 $x=(p,e)$，这产生了状态空间 $X=R\times R\subset\mathbb{R}^4$。因为逃跑者的位置未知，所以目前的状态未知，于是就出现了 *I*-空间。观测空间 Y 是 R 的一个子集族。对于每个 $p\in R$，传感器产生了一个可视多边形 $V(p)\subseteq R$（这由 $y=h(p,e)$ 用第 11.1.1 节的符号来表示）。考虑 t 时刻的历史 *I*-状态。给定追赶者的初始位置 $p(0)$（如果没有给定，可以选择任意位置），逃跑者可能在 R 中的任何地方。历史 $\tilde{u}_t$ 可以表示为追赶者的历史 $\tilde{p}_t$[2]。因此历史 *I*-状态为：

$$\eta_t=((p(0),R),\tilde{p}_t,\tilde{y}_t) \tag{12.33}$$

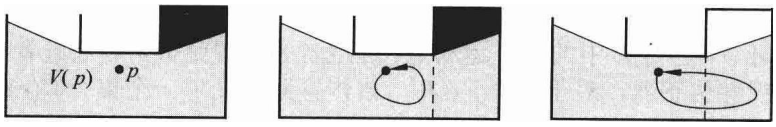
其中 $(p(0),R)\subset X$ 是初始条件，且 $p(0)$ 已知，并且逃跑者的位置 $e(0)$ 可能在 R 中的任何位置。

考虑非确定的 *I*-空间 $\mathcal{I}_{ndet}$。因为追赶者的位置总是已知的，那么 R 的有趣子集是可能隐藏逃跑者的地方。因此非确定的 *I*-状态可以表示为 $X_t(\eta_t)=\big(p(t),E(\eta_t)\big)$，其中 $E(\eta_t)$ 是在给定 η_t 的情况下逃跑者可能位置的集合。像通常的非确定的 *I*-状态一样，$E(\eta_t)$ 是与 η_t 中所有信息相符的最小集合。

考虑 $E(\eta_t)$ 是如何随时间变化的。第一个瞬间过后，观测到 $V(p(0))$，知道逃跑者处在 $R\setminus V(p(0))$，它是相对于 $p(0)$ 的阴影区域（在第 12.3.4 节定义）。当追赶者运动时，$E(\eta_t)$ 也发生变化。假设告诉你追赶者的位置现在在 $p(t)$，但是没有告诉你 η_t。那么 $E(\eta_t)$ 可能是什么？这取决于历史，但我们所感兴趣的只是每个阴影分支包含逃跑者的可能性是多少。对于某些分支来说，我们能够确信它们没有包含逃跑者。例如考虑图 12.38。假设开始时追赶者确信逃跑者可能在走廊的尽头。如果它沿着更小的闭环路径运动，那么非确定的 *I*-状态逐渐变化，但是当完成环路时非确定的 *I*-状态又返回到相同的值。但是如果追赶者经过更大的环路，那么基于所完成的环路能够确信走廊里没有包含逃跑者。这个例子中被穿越的点线可能会使你想到基于临界边界的胞腔剖分，如第 6.3.4 节中介绍的算法。稍后将沿着这个思路来研究解决该问题的完备算法。但是在提出完备算法之前，首先考虑一些有趣的例子。

1　如果遵循标准函数的符号，那么最好用 $\tilde{e}(t)$ 代替 $e(t)$ 来表示 t 时刻的位置，但这里不这样表示。

2　为了遵循第 11.4 节的符号，可以用 $\dot{p}=u$ 运动模型，其中 u 表示追赶者的速度。可以用大自然行动来建模逃跑者的速度以得到 $\dot{e}$。通过对 $\dot{p}$ 进行积分，可以得到任意时刻 t 的 $p(t)$。这意味着 $\tilde{p}_t$ 可以当作历史的一个简单表示，而不是速度。



(a) (b) (c)

图 12.38 （a）假设追赶者来到一个受污染的走廊尽头附近；（b）如果追赶者沿环形路径运动，那么非确定的 I-状态逐渐发生变化，但是能够返回到它的原始值；（c）但是如果越过一个临界边界，那么非确定的 I-状态会发生根本变化

例 12.7（问题在什么时候是可解决的？） 图 12.39 显示了四个相似的问题。逃跑者的位置没有显示出来，因为问题是通过确保没有还隐藏着的逃跑者来解决的。注意到追赶者的速度与非确定的 I-状态无关。因此可以用追赶者的路径来定义问题的解。前三个例子很容易解决。但是第四个例子无解，因为至少有三个不同的隐藏位置（你能发现吗）。令 $V(V(p))$ 表示至少从 $V(p)$ 的一个点可视的所有点的集合。妨碍问题的解决是由于对于每个 $i,j \in \{1,2,3\}$，$i \neq j$，存在三个位置 p_1、p_2、p_3 使得 $V(V(p_i)) \cap V(V(p_j)) = \phi$。当到达一个隐藏位置，逃跑者可以溜到其他两个位置。在最坏情况下，可能导致对逃跑者的无穷追赶，而逃跑者总能够躲开被发现的危险。我们希望算法能够系统地搜索 $\mathcal{I}_{ndet}$ 并确定是否存在问题的解。 ■

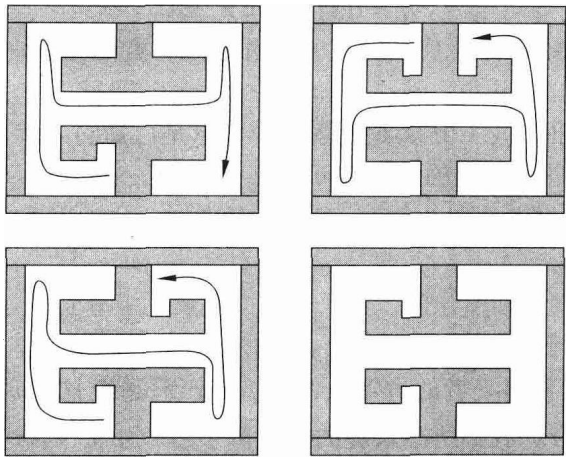


图 12.39 用一个追赶者就能够轻松解决的三个问题，以及一个引起无解的微小变化

因为某些问题只靠一个追赶者是不能解决的，那么人们就会想知道两个追赶者是否能够解决任何问题。下一个例子给出了一个环境序列，它指出对于任何正整数 k，都存在这样一个环境，它需要 k 个追赶者才能解决追逃问题。

例 12.8（一个困难问题的序列） 图 12.40 所示的序列中的每个环境都要求比以前的多一个追赶者[417]。构造是基于递归地确保有三个孤立隐藏位置，如图 12.39 最后一个问题中那样。每当这种情况发生时，需要增加一个追赶者。该序列递归地添加需要 k 个追赶者的三个环境来得到需要 $k+1$ 个追赶者的问题。总是需要一个额外的追赶者来守护在将三个环境的连接处。构造过程是基于 3-可分离性的概念，它是从文献[776]所研究的在一个图上进行的追逃问题中发展而来的。 ■

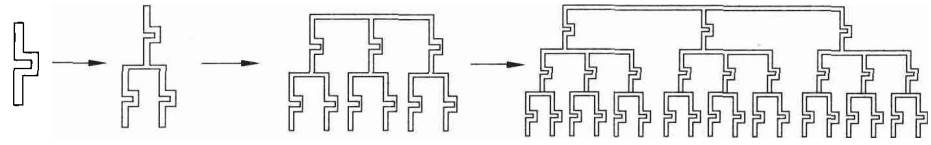


图 12.40　每个走廊族要求比前一个多一个追赶者，因为需要一个新的追赶者守卫在连接处

考虑多连通的环境，问题会更具挑战性（有多个孔的环境）。单个追赶者不能解决这些问题。确定解决这些问题所需要的追赶者的最小数量是一个 NP-复杂度的[417]。

12.4.2　一个完备算法

现在考虑为只有一个追赶者情况下的问题而设计一个完备算法。“完备的”（complete）是指如果解存在，那么必须找到该解；否则，它必须正确报告可能无解。图 12.38 的非确定的 I-状态只在越过临界边界以后才会以一种非常有趣的方式发生变化。仔细分析所有可能发生这些临界变化的情况就可以解决追逃问题。这与第 12.3.4 节所考虑的情况相同：穿越拐切射线和双切射线。图 12.38 就是一个穿越拐切射线的例子。图 12.41 指出第 12.3.4 节的缺口和可能包含逃跑者的部分环境之间的联系。

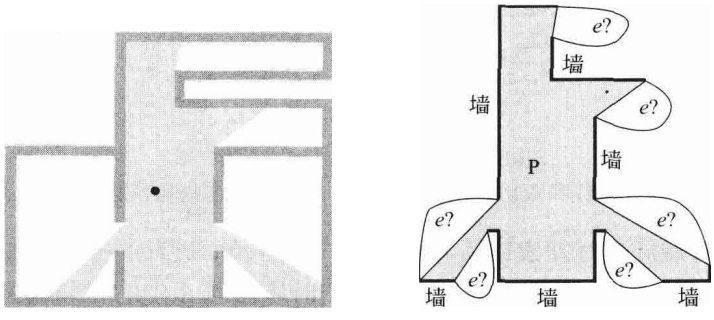


图 12.41　回顾图 11.15；每个缺口以外就是环境中可能包含，也可能不包含逃跑者的部分

阴影区域是对某 $p(t)$ 不可视的所有点的集合，这表示为 $R \setminus V(p(t))$。每个临界事件会改变阴影分支的数量。如果一条拐切射线被穿越，那么根据方向的不同，一个阴影分支要么出现，要么消失。如果一条双切射线被穿越，那么两个阴影分支合并成一个，或者一个分支分裂成两个。为了跟踪非确定的 I-状态，必须确定阴影区域的每个部分是干净的（cleared），还是受到污染的（contaminated），干净意味着确信它不包含逃跑者，污染是指它可能包含逃跑者。初始情况下，所有分支都被标记为受污染的，随着追赶者的运动，干净的分支能够合并。解决追逃问题等同于移动追赶者直到所有阴影分支都是干净的。这样就不会剩下隐藏逃跑者的位置。

如果追赶者穿越一条拐切射线并出现一个新的阴影分支，那么总将它标记为干净的，因为这是环境的可视部分。如果追赶者穿越一条双切射线并发生一次分裂，那么跨越两个分支进行标记：一个受污染的阴影分支分裂成两个受污染的分支，一个干净的分支分裂成两个干净的分支。如果沿相反方向穿过双切射线，则会导致分支的合并，那么情况会更加

复杂。如果一个分支是干净的，而另外一个是污染的，那么合并后标记为受污染的。只有两个分支都是干净的，那么合并后的分支才被标记为干净的。注意到在这四种临界情况中，只有合并有能力破坏追赶者的工作。也就算说，它可能导致再次污染。

考虑基于拐切射线和双切射线将 R 剖分为胞腔，如图 12.42 所示。这些胞腔有**信息守恒的性质**（information-conservative property），如果追赶者沿着处在一个二维胞腔内的任何环形路径行进，那么返回到开始时 I-状态保持相同。这意味着追赶者所选择的用来穿过一个胞腔的特别路径并不重要。追逃问题的解可以被描述为必须访问的相邻二维胞腔的一个序列。由于有信息守恒性质，所以可以任意选择穿过胞腔序列的特别路径。

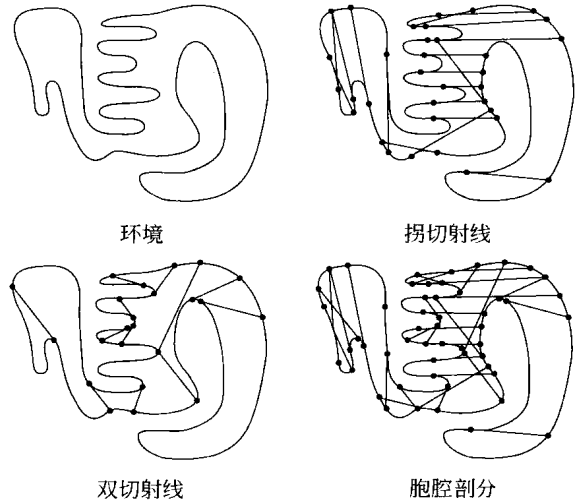


图 12.42 基于拐切射线和双切射线将环境剖分成胞腔，这些胞腔是仅有的关键可视事件

为寻找解而对胞腔进行搜索比第 6 章中为寻找路径而进行的搜索更加复杂，因为搜索过程必须在 I-空间中进行。追赶者可能在不同的时候访问 R 中的相同胞腔，但是对关于哪个分支是干净的或者受污染的，则有不同的认识。可以构造一个有向图 $\mathcal{G}_I$。对于 R 中的每个二维胞腔以及每个可能的阴影分支的标记，在 $\mathcal{G}_I$ 中定义一个顶点。例如，如果一个胞腔的阴影区域有三个分支。那么在 $\mathcal{G}_I$ 中就有 $2^3=8$ 个相应的顶点。在 $\mathcal{G}_I$ 中，在两个顶点之间存在一条边，如果：（1）它们相应的胞腔是相邻的；（2）各分支的标记与穿过两个胞腔的边界所引起的变化相一致。第二个条件意味着必须遵守出现、消失、分裂或合并等情况的标记规则。例如，如果穿越边界引起一个受污染的阴影分支发生分裂，那么新的分支必须标记为受污染的，并且所有其他分支必须保持原来的标记。注意到 $\mathcal{G}_I$ 是有方向的，因为 $\mathcal{I}_{ndet}$ 中的许多运动都不是可逆的。例如，如果一个受到污染的区域消失，那么它将不会被逆向路径再次污染。注意到该有向图的信息不会像第 12.2.1 节的懒惰离散定位的情况那样单调改进。在目前情形下，在阴影分支合并时的信息可能更糟了，因为污染是可以扩散的。

为了搜索 $\mathcal{G}_I$，从对应于所有阴影区域分支都标记为受污染的顶点开始，并不需要特定的起始胞腔。可以应用第 2.2 节的任何搜索算法来寻找一个目标顶点，目标顶点是 $\mathcal{G}_I$ 中这样的顶点：对于它来说，所有阴影分支都被标记为干净的。如果没有这样的顶点，那么算法将正确地宣布无解。如果发现一个目标顶点，那么 $\mathcal{G}_I$ 中的路径给出为解决问题而必须访

问的胞腔的序列。然后从胞腔的序列来构造 R 中的实际路径。某些胞腔可能不是凸的，但是它们的形状非常简单，所以不需要用复杂的运动规划算法来构造一条遍历胞腔序列的路径。

这里提出的算法从概念上非常简单，实际中也能很好地实现，但是它的最坏情况运行时间与拐切射线数量呈指数关系。考虑一个用 n 条边表示的多边形环境。可能有 $O(n)$ 条拐切线和 $O(n^2)$ 条双切线。胞腔的数量以 $O(n^3)$ 为界[415]。非常遗憾，$\mathcal{G}_I$ 有指数数量的顶点，因为有 $O(n)$ 个阴影分支，并且如果有 n 个分支，那么就有 2^n 个可能的标记。注意到在进行搜索之前不需要计算 $\mathcal{G}_I$。它是在规划过程中逐步显现出来的。最有效的完备算法虽然更加复杂，但能够在 $O(n^2)$ 时间内解决追逃问题，算法是这样产生的：首先证明一个追赶者采用可视多边形能够解决的任何问题都可以由只用两束光的追赶者来解决[773]。这将 $V(p(t))$ 从 R 中的二维区域简化为两条从 $p(t)$ 射出的可旋转射线，显著地减小了 I-空间的复杂度。

12.4.3　其他类型

本节提到的追逃问题有大量的变化形式。如果有多个追赶者，那么问题会变得更加困难。胞腔剖分可以基于改变阴影分支来进行，但是由于不同追赶者可视多边形之间存在相互影响，所以某些胞腔的边界是用代数表示的表面。因此实现一个完备算法是很困难的。另一方面，可以采用启发式方法来引导多个追赶者。单个追赶者可以用本节描述的完备算法。当该追赶者失败时，它可以运动到环境中的某位置进行等待，而此时第二个追赶者在每个阴影分支上应用完备的单追赶者算法。对于任意数量的机器人，这种思路可以递归使用。

对逃跑者的速度进行限制会使得问题更加复杂。即使这更有利于追赶者，但是很难设计一个正确利用这些信息的完备算法。这种情况目前不存在完备算法。

图 12.43 显示了几个检测模型，它们产生 $V(p(t))$ 的不同定义。每个都要求不同的追逃算法，因为 I-空间的结构对于不同的感测模型来说会有显著变化。例如使用图 12.43（c）中的模型，那么要求单个追赶者沿着 ∂X 运动。一旦它进入内部，那么阴影区域总变成一个单连通分支。这种模型有时被称为一个手电筒（flashlight）。如果有两个手电筒，那么其中一个可以运动进入内部，而另外一个保护以前的工作。如图 12.43 所示的深度受限情况是很实际的，但也非常具有挑战性。所需要的追赶者的数量一般情况下依赖于环境的度量性质，如最小“厚度”等。在文献[383]中，本节提到方法被扩展到视场受限的情况，将得到与第 6.3.4 节相似的临界曲线。

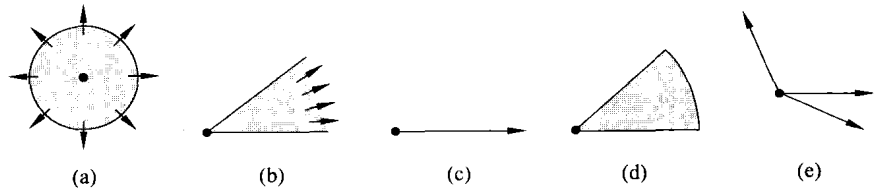


图 12.43　几种逃跑者检测模型：（a）没有距离限制的全向感测；（b）受限视场可视性；（c）能够旋转的单条可视射线；（d）受限距离和一个旋转视角锥，这对应于一个摄像机模型；（e）三条能够旋转的可视射线

12.5 存在感测不确定性下的操作规划

I-空间的最丰富来源之一就是操作规划。当机器人与世界中的障碍物或者物体之间相互影响时，状态估计的负担就变得更加繁重。讨论这个问题的经典方法是限制机器人与障碍物之间相互影响的方式。在第 7.3.2 节的操作规划框架下，这意味着机器人必须抓住并搬运物体到指定位置。机器人抓物体时，物体必须处在一个稳定位形，在抓住以后它必须返回到一个稳定位形。

当取消关于经典操作规划框架的假设时，预测机器人和其他物体的行为就变得更加困难。这就产生了预测中的不确定性问题，这是第 10 章的基础。接下来的问题是设计传感器，以使在存在不确定性的情况下依然能够完成规划。对于每个感测模型，都会出现 *I*-空间。

第 12.5.1 节将介绍原像规划的框架[313,662]，在这种框架下我们将针对特定操作规划问题而讨论第 10 章和第 11 章介绍的一些有趣的问题。*I*-状态、前向投影、后向投影以及终止行动都将在这种背景下进行描述，并且用该框架已经证明了在不确定性情况下的几种规划算法的复杂度结果。

第 12.5.2 节介绍的方法能够清晰阐述在 *I*-空间上讨论问题的优势所在。基本原理是允许非抓握形式的操作（例如推、挤、投掷等）以及设计简单的传感器，或者甚至完全不用感测。这在显著减小 *I*-空间的同时，依然存在可行规划。这与信息越多越好的直觉相矛盾。使用较少的信息会导致更大的状态不确定性，但是在某些问题中这是不重要的。重要的是 *I*-空间变得更简单了。

12.5.1 原像规划

原像规划（preimage planning）的框架（或者 LMT 框架，由 Lozano-Perez、Mason 和 Taylor 命名）是作为在存在不确定性情况下进行操作规划的一般方法而开发的[313,662]。尽管这个概念可以应用于一般位形空间，但这里是针对 $\mathcal{C}=\mathbb{R}^2$ 以及 $\mathcal{C}_{obs}$ 是多边形这种情况的。对于该框架都进行这种假设。这对应于在 $\mathcal{W}=\mathbb{R}^2$ 中进行平移的同时可能还要搬运工件的一个机械手的简单模型。peg-in-hole 问题就是一个例子，其中一个工件必须插入一个稍大一点的孔。当制造业中的机器人装配产品时经常进行这种操作。使用第 4.3.2 节的位形空间表示，机器人变成了一个在 $\mathbb{R}^2$ 中穿行于多个多边形障碍物之间的一个点。

在原像规划中所使用的模型有许多独特性质：

（1）机器人能够进行适应性的运动，即它能够沿着 $\mathcal{C}_{obs}$ 的边界进行滑动。这不同于第 II 部分中通常要求机器人必须避开障碍物。

（2）在预测中存在非确定的不确定性。一个行动确定一个运动的方向，但是大自然决定在执行过程中会出现多少误差。假设有一个有界误差模型。

（3）在感测中存在非确定的不确定性，所以并不能可靠地估计真正的状态。

（4）目标区域通常是 $\mathcal{C}_{obs}$ 的一条边，但是更一般的是 $\mathcal{C}_{free}$ 的闭包 $cl(\mathcal{C}_{free})$ 的子集。

（5）使用层次规划模型，其中给机器人发一系列运动指令，每个指令是通过应用基于

I-状态的 u_T 来终止的。

现在详细说明上述每种情况。

适应性运动　稍后读者将会看到执行适应性运动的可能性对减小机器人位置的不确定性是至关重要的。令 $\mathcal{C}_{con}$ 表示障碍物的边界 $\partial\mathcal{C}_{obs}$（即 $\mathcal{C}_{con}=\partial\mathcal{C}_{obs}$）。需要表述机器人的一个运动模型，同时满足 $q\in\mathcal{C}_{con}$。一般情况下，摩擦使问题复杂化。这里假设采用一个简单的库仑摩擦模型，关于在操作规划中对摩擦进行建模的详细内容见文献[684]。假设机器人在某 $q\in\mathcal{C}_{con}$ 处应用净力 F。该力可以用机器人控制中的广义阻尼器模型[966]。

所产生的运动采用摩擦力的锥（friction cone）来刻画，如图 12.44（a）所示。牛顿力学的基本原理是障碍物产生反作用力。如果 F 指向表面并垂直于该表面，那么障碍物的反作用力将会抵消 F，这样就没有运动产生。如果 F 不垂直于该表面，那么可能会发生滑动。F 还可能平行于该表面。在这种情况下，它一定会沿着边界滑动。一般情况下，F 可以被分解为平行分量和垂直分量。如果平行分量相对于垂直分量来说太小，那么机器人就不动。图 12.44（a）所示的摩擦力的锥精确地表明了发生运动所需要的条件。参数 α 描述摩擦力的数量（摩擦力越大导致 α 越大）。图 12.44（b）表明了 F 的不同方向上所发生的行为。颠倒摩擦锥就可以得到图 12.44（b）。如果 F 指向底部区域，那么机器人固定不动。如果 F 的方向离开障碍物的边界，那么会脱离接触（除非边界是黏性的）。对于其余两种情况，机器人沿着边界滑动。

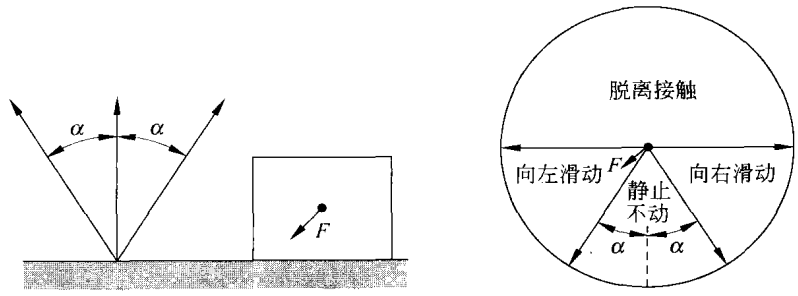


图 12.44　适应性运动模型；如果机器人在 $q\in\mathcal{C}_{con}$ 处应用力 F，那么只有当 $-F$ 指向摩擦力的锥的外面时，它才沿着边界运动

不确定性的来源　大自然干扰位形转移和传感器。令 $U=[0,2\pi)$，它表示在 $\mathbb{R}^2$ 中机器人被命令前进的方向。大自然将影响这个命令，并且实际的方向处在 $\mathbb{S}^1$ 的一个区间内。如图 12.45（a）所示，一个固定行动 $u\in U$ 的前向投影产生了将来可能位形的一个锥。可用例 13.15 中的微分方程来指定精确运动模型。如图 12.45（b）所示的感测模型已经在第 11.5.1 节给出。大自然感测行动形成了式（11.67）给出的一个圆和 $y=q+\psi$，其中 q 是真实位形，ψ 是大自然感测行动，y 是观测。结果如图 11.11 所示。

目标区域　因为允许与障碍物接触，因此定义的目标区域除了包括 $\mathcal{C}_{free}$ 中的点以外，还包括 $\mathcal{C}_{obs}$ 的边。通常选择 $\mathcal{C}_{obs}$ 的一条边作为目标区域。

运动命令　现在描述规划问题。用连续时间来表达模型，这与离散情况不同。这是一个可行的方法，但是导致了微分约束下的规划，微分约束是第Ⅳ部分将要介绍的内容，它相当复杂。在原像规划框架中，采用层次方法。将会定义一种受限的规划 μ，它被称为运

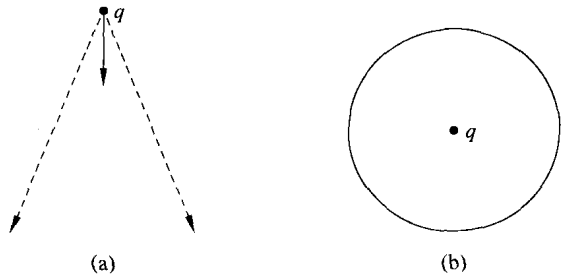


图 12.45 大自然干扰位形转移和传感器观测

动命令（motion command），并且目标是通过构造一个运动命令序列来完成的。其效果是将连续时间决策问题转换为涉及离散阶段的规划问题。每次应用运动命令时，机器人必须应用终止行动来结束它。然后再发另外一个运动命令。因此假设有一个高层模块发出运动命令，一个底层模块执行命令直到满足终止条件。

对于某行动 $u \in U$，令 $M_u = \{u, u_T\}$，其中 u_T 是终止行动。一个运动命令是一个反馈规划 $\mu : \mathcal{I}_{hist} \rightarrow M_u$，其中 $\mathcal{I}_{hist}$ 是基于初始条件、行动历史和感测历史的标准历史 *I*-空间。运动命令是在连续时间上执行的。$t = 0$ 时，$\mu(\eta_0) = u$。使用在执行过程中收集到的历史 *I*-状态 η，运动命令最终将产生 $\mu(\eta) = u_T$，从而结束运动命令。如果目标没有完成，那么高层模块可以发出另外的运动命令。

原像 现在考虑如何构造运动命令。使用层次方法，至于目标区域这个主要任务可以分解为多个中间子目标。运动命令 μ 和子目标 $G \subset cl(C_{free})$ 的原像 $P(\mu, G)$ 是确保能够完成运动命令 μ 的所有历史 *I*-状态的集合，尽管都受到大自然的干扰。每个运动命令必须识别已经完成的子目标，以使它能够应用它的终止行动。一旦完成一个子目标，那么所产生的历史 *I*-状态必须落在规划中下一个运动命令所要求的历史 *I*-状态的集合内。令 $\mathcal{M}$ 表示能够定义的所有可容许的运动命令集合。这实际上可以被认为是高层模块的一个行动空间。

用运动命令进行规划 可以构造一个高层开环规划[1]

$$\pi = (\mu_1, \mu_2, \cdots, \mu_k) \tag{12.34}$$

它是 k 个运动命令的序列。尽管机器人执行的路径是不可预测的，但假设运动命令序列是可预测的。每个运动命令 μ_i，$1 < i < k$，必须终止于一个 *I*-状态 $\eta \in P(\mu_{i+1}, G_{i+1})$。$\mu_1$ 的原像必须包括初始 *I*-状态 η_0。目标由最后一个运动命令 μ_k 来完成。

更一般情况下，所选择的特殊运动命令不一定是可预测的，它可能依赖于执行过程中的 *I*-状态。在这种情况下，可以开发高层反馈规划 $\pi : \mathrm{I}_{hist} \rightarrow \mathrm{M}$，其中运动命令 $\mu = \pi(\eta)$ 的选择是基于前一个运动命令结束后所产生的历史 *I*-状态 η。文献[283,313,591]介绍了这些变化。

高层规划问题可以用第 2 章～第 10 章中的离散规划算法来解决。在原像规划框架内最常用的方法是从目标开始进行后向搜索。尽管这看起来似乎很简单，但是可能的运动命令集合是无限的，并且很难用导致完备性的方法对 μ 进行采样。另外复杂性还在于终止是基于历史 *I*-状态的。因此规划十分具有挑战性。文献[313]通过图灵机停止问题[892]指出：一般情况下的原像是不可计算的。文献[735]显示原像规划的三维版本是 PSPACE 复杂度的，

1 注意到，这个开环规划由闭环运动命令组成。使用层次建模，这是完全可以接受的。

其中障碍物是多面体。文献[174]显示它甚至是 NEXPTIME 复杂度的[1]。

后向投影　通过分离可达性和可辨识性问题[313,314]，Erdmann 提出了一个实用方法来计算运动命令。可达性指的是保证是可达的点的集合。可辨识性是指获悉基于历史 I-状态已经到达子目标。对分离的另外一种解释方法是大自然对位形转移的影响从大自然对感测所造成的影响中分离出来。

为分析可达性，忽略感测不确定性。可以用第 10.1.2 节的前向投影和后向投影的概念。不同仅在于它们被应用于连续空间和运动命令（代替 u）。令 S 表示 $\mathrm{cl}(\mathcal{C}_{free})$ 的一个子集。可以定义弱后向投影 $\mathrm{WB}(S,\mu)$ 和强后向投影 $\mathrm{SB}(S,\mu)$，并且可以定义无方向的后向投影[285] $\mathrm{WB}(S)$ 和 $\mathrm{SB}(S)$，它们分别类似于式（10.25）和式（10.26）。

图 12.46 显示了一个简单问题，该问题的任务是用向下的运动命令以到达目标边。这是受到 peg-in-hole 问题的启发。对于图 12.46 中的问题，图 12.47 阐述了几个从目标区域开始的后向投影。行动是 $u=3\pi/2$，但是实际的运动处在大自然所引起的锥中。首先假设除了在目标区域内以外，在其他地方都不允许接触障碍物。图 12.47（a）给出强后向投影。从三角区域中的任意点开始，保证能够到达目标，不管大自然的影响。弱后向投影是图 12.47(b) 所示的无界区域。这表示了可能到达目标的位形。不进一步考虑弱后向投影，因为保证达到目标是更重要的。强后向投影可以完成该任务。所以假设在默认情况下后向投影都指的是强后向投影。用弱后向投影，有可能开发出一种错误检测和恢复（error detection and recovery，EDR）的框架，这是由 Donald 在文献[283]中提出的。

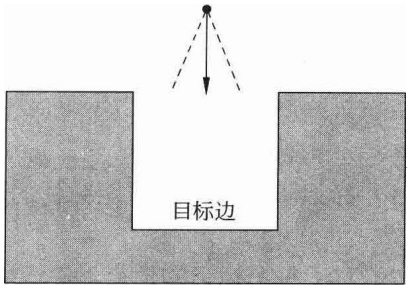


图 12.46　类似于 peg-in-hole 问题的一个简单例子

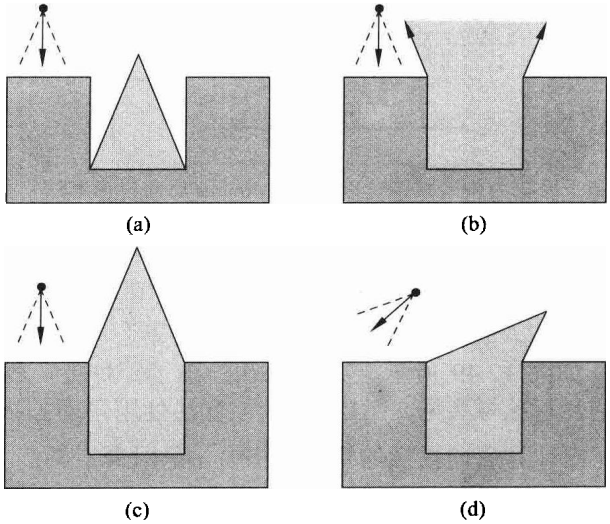


图 12.47　显示了 peg-in-hole 问题的几个后向投影

1　NEXPTIME 是可以在非确定的指数时间内得到解决的所有问题的复杂性类型。这超出了图 6.40 所示的复杂性类型。

现在假设能够沿着障碍物边界进行适应性运动。这有扩大后向投影的效果。为简单起见，假设没有摩擦（图 12.44（a）中的$\alpha=0$）。后向投影如图 12.47（c）所示。当机器人接触到侧面的墙时，它向下滑动直到到达目标。在这种情况发生时，不需要对精确位形进行跟踪。这就说明了在减少不确定性方面适应性运动的作用。这一点将在第 12.5.2 节进一步介绍。图 12.47（d）显示了不同运动命令的后向投影。

现在考虑在更一般背景下计算后向投影。后向投影可以从$cl(\mathrm{C}_{free})$的任意子集来定义，并且允许一个参数为α的摩擦锥。为了包含在后向投影中，必须避免有可能被粘住的点。注意到即使$\alpha=0$，也有可能被粘住。例如在图 12.46 中，大自然可能允许运动垂直于目标边界。在这种情况下，水平边被粘住，因为没有切向运动。一般情况下，必须确定在$\mathcal{C}_{obs}$的每条边和每个顶点上机器人是否有可能被粘住。机器人在一条边上是否可能被粘住，这取决于u、α以及由大自然引起的最大方向误差。如果机器人有可能在一个顶点的两条邻边上被粘住，那么它在该顶点上能够被粘住。

计算后向投影 已经开发了计算后向投影的一些算法。文献[313,314]给出了第一种算法。假设目标区域是包含在$\mathcal{C}_{con}$的边中的一个或多个线段。算法是针对固定的运动命令μ的，μ是基于如下方向$u \in U$：

（1）标记障碍物每个可能粘住的顶点。也标记目标区域的边界上的可能滑离目标的任意点。

（2）对于每个标记的顶点，基于执行u时大自然允许的最大可能偏差，延伸两条方向射线。这将颠倒如图 12.45（a）所示的锥。无摩擦情况下$(\alpha=0)$延伸的射线如图 12.48 所示。

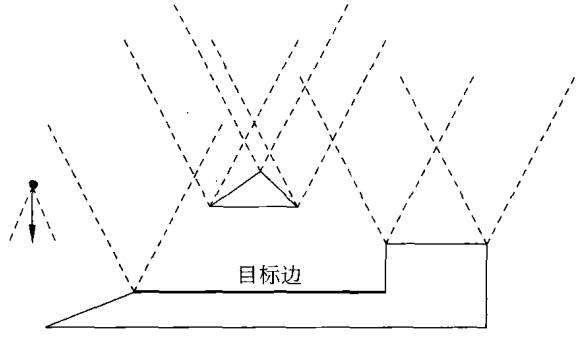


图 12.48 基于摩擦力构造圆锥体以后，Erdmann 后向投影算法将描绘出边界

（3）从每条目标边开始，描绘后向投影区域的边界。遇到的每条边定义了位形的一个半平面，机器人从这些位形一定能够运动到这些位形中。在图 12.48 中，这对应于在射线下面。当描绘出后向投影边界时，基于包含半平面中的点来确定每个交叉顶点的方向。

所产生的后向投影如图 12.49 所示。文献[285]给出了一种更一般的算法，它应用于包含$\mathcal{C}_{free}$中多边形区域的目标区域（见文献[591]）。该算法使用平面扫描原理（见第 6.2.2 节）来产生在$O(n\lg n)$时间内计算后向投影的算法，其中n是用来定义$\mathcal{C}_{obs}$的边的数量。后向投影本身的边不超过$O(n)$条。文献[142,285]给出了计算无方向后向投影的算法。在这种情况下后向投影边界可能十分复杂。文献[142]给出了一种在$O(n^2\lg n)$时间内计算大小为$O(n^2)$的无方向后向投影的递增算法。

一旦得到计算后向投影的算法，那么它需要进行适应性修改来计算原像。使用图12.45(b)所示的感测模型，通过收缩子目标区域 G 就可以得到原像。令 ε 表示图 12.45(b)中球的半径。令 $G' \subset G$ 表示子目标的一个子集，其中已经删除了一个厚度为 ε 的带。如果传感器返回 $y \in G'$，那么能够保证 $q \in G$。这就产生了一种通过收缩子目标来得到原像的方法。但是如果 ε 太大，这也许不能产生成功的规划，即使规划存在。

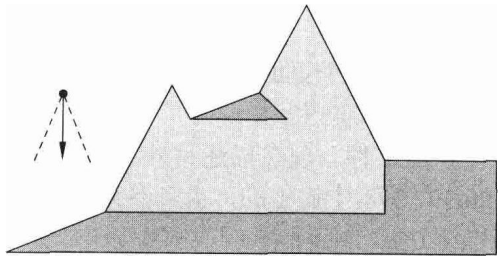


图 12.49　计算的后向投影；三角形的陡峭的边保证滑动的进行，因此，它包括在后向投影中；顶部的其他边有可能会发生停滞

可以通过进行后向搜索来寻找高层规划，后向搜索过程从目标区域（减小了 ε）开始计算后向投影。仍然存在 $\mathcal{M}$ 太大这种困难，这影响了搜索中的分支因子。一种可能是计算无方向的后向投影。另一种可能是对 $\mathcal{M}$ 进行离散化。例如在文献[591,593]中，$\mathcal{M}$ 减小到只有四种基本方向，并且通过将机器人被粘的边作为子目标来为复杂环境计算规划。但是离散化使得更难保证规划算法的完备性。

原像规划框架似乎只能用于非常特殊的模型，但是它可以被扩展和修改而应用于更一般的背景中。在文献[176]中它被扩展到半代数障碍模型，其中给出的规划方法的运行时间是 C 空间维数的两倍指数关系（基于第 6.4.2 节介绍的圆柱代数剖分）。在文献[149]中，对概率后向投影的介绍是通过给本节所考虑的大自然行动空间指派一个均匀概率密度函数来进行的。这可以进一步进行推广，将后向投影和原像定义为文献[600,608]中最优尚需代价函数的水平集。然后可以应用动态规划方法来计算规划。

12.5.2　非抓握操作

抓握操作是很受限制的。人们可以不用抓握方法来操作物体。可以对物体进行推、抛、投掷、挤压、旋转、拍打、吹等操作。这些都是非抓握操作。来自于厨房的一个典型例子是握住锅把的同时轻拍手腕将一张薄饼抛起来。

使机器人抓握物体是由于受到来自于状态估计和控制的困扰。对于非抓握操作来说，任务更难以完成，因为会出现失去对物体的直接控制这种情况。这就导致了预测中更大的不确定性以及更大的感测负担。但是在 I-空间中进行规划有可能避免这些问题。几个方面的工作显示：用很少的感测或者根本不用感测就可以实现操作目标。这也是一种形式的精简思想[177,323,684]，其中传感器的设计采用的是简化 I-空间的方式，而不是为精确估计而担忧。精简机器人系统的搜索与第 11.2.1 节提到的发现尽可能小的衍生 I-空间完全一致。感测系统应该很简单，但依然能够完成任务，并且最好不失去完备性。这个领域大多数工作开始关心的都是发现可行解，而不是最优解。这可以进一步简化 I-空间。

本节给出一个例子，它代表了精简主义的一个极端情况。开发一个无传感器操作的系统。乍看起来这很荒谬。从第 10.1.2 节的前向投影可以看出：如果大自然在位形转移中引起不确定性并且没有传感器时才会增加不确定性。与直觉相反，适应性运动能够减少不确定性。这与第 11.5.4 节的讨论相一致。只要知道已经成功地应用了某些运动命令，就能够

减少不确定性。在以前的关于无传感器操作的演示中，通过将托盘在几个方向上进行倾斜就可以将盘上的一个通用扳手放置在一个已知方向上[323]。无论扳手的初始位形如何，最后都能够实现同一个方向。这不需要传感器。这可以认为是对图 11.29 所示的在一个盘上滚动的球的一个更加复杂的扩展。这也是图 12.44 所示的适应性运动的一个例子，但是目前，F 由重力产生。

挤压工件 现在描述另外一个无传感器操作的例子，它是 Goldberg 和 Mason 在文献[396,397,398]中提出的，也可参见文献[684]。算法的一个 Java 实现出现在文献[133]中。假设凸多边形工件沿着工厂中的传送带一个一个地到达。它们用在装配操作中，并需要放置在给定方向。图 12.50 显示了一个平口虎钳的顶视图。图 12.50(a)显示的是挤压之前的工件，图 12.50(b)显示的是挤压之后的工件。假设机构有一个简单模型。平口虎钳钳口以恒定速度相向运动，假设它们运动非常慢以至于可以忽略动态。为了有助于工件滑入相应位置，把其中一个钳口当作是无摩擦的（真实设备就是如此，见文献[177]）。机器人能够在 $[0,2\pi)$ 中的任意方向上进行挤压操作（由于对称的缘故，实际上只需要 $[0,\pi)$）。令 $U=[0,2\pi)$ 表示所有挤压行动的集合。工件不能进一步挤压（在不压碎工件的情况下）时，挤压行动自行结束。

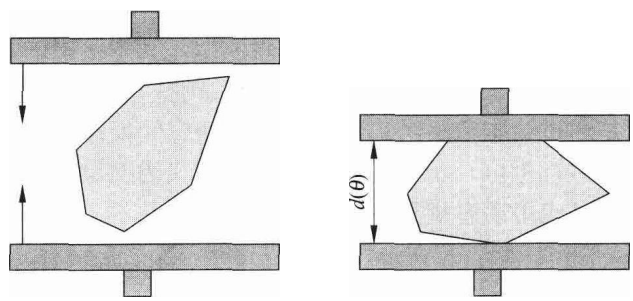


图 12.50 一个平口虎钳能够对一个工件进行定向，而不需要使用传感器

规划问题可以建模为与大自然之间的对策博弈。工件的初始方向 $x\in[0,2\pi)$ 由大自然进行选择，它是未知的。状态空间为 $\mathbb{S}^1$。对于给定的工件，任务是设计挤压操作序列：

$$\pi=(u_1,u_2,\cdots,u_n) \tag{12.35}$$

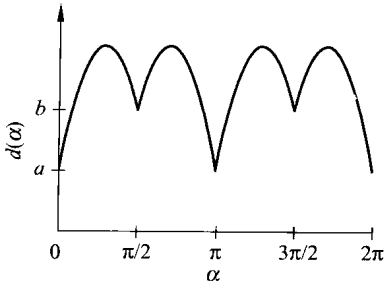
它导致工件朝向给定方向，不管它的初始状态如何。对最终状态没有特别的要求。i 个运动命令结束以后，历史 I-状态就是到目前为止所应用的挤压序列：

$$\eta=(u_1,u_2,\cdots,u_i) \tag{12.36}$$

将不使用非确定的 I-空间 $\mathcal{I}_{ndet}$。这可以叙述为得到只有一个非确定的 I-状态的单元素集合（只有一个可能的方向）。如果工件是对称的，那么任务就是确定单个对称类（它只包括有限个方向）。

考虑在未知方向上工件的行为。由于旋转对称，所以基于工件和机器人之间的相对角度，很容易描述挤压操作的效果。因此令 $\alpha=u-x$，并假设模 2π。初始情况下，α 可能是 $[0,2\pi)$ 中的任意值。经过一次挤压以后，α 总是被迫改变为一个确定值。这可以通过厚度函数 $d(\alpha)$ 来说明，它表示了在 α 方向上工件横截面最大厚度。图 12.51 显示了矩形的截面。距离函数的局部最小值表示了工件稳定下来以后的方向，如图 12.50（b）所示。在挤压过

程中工件改变其方向时，α 的值朝着逐渐减小 $d(\alpha)$ 的方向变化。因此 $[0,2\pi)$ 可以被分为多个吸引域，如图 12.52 所示。它们的行为就像第 8.5.1 节中的漏斗一样。



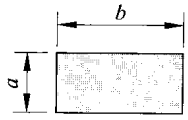


图 12.51　一个矩形的厚度函数

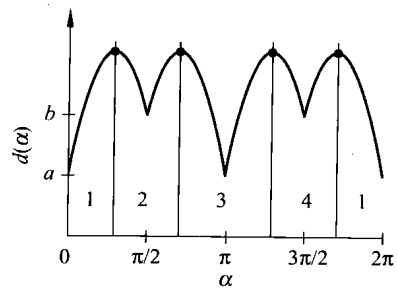


图 12.52　有四个吸引域，每一个代表一个方向区间

解决无传感器问题的关键是每次挤压不能使得不确定性向更坏的方向发展，并且通常还能减小不确定性。假设 u 是固定的。对于状态转移方程 $x'=f(x,u)$，在一个区间内的 x 值产生相同的 x'。由于旋转对称，这最好用 α 来表达。令 $s(\alpha)$ 表示一次挤压以后的相对方向。因为 α 是 x 和 u 的函数，这可以表示为一个挤压函数 $s:\mathbb{S}^1\to\mathbb{S}^1$，定义为：

$$s(\alpha)=f(x,u)-u \tag{12.37}$$

也可以定义关于 α 值的一个区间 A 的前向投影：

$$S(A)=\bigcup_{\alpha\in A}s(\alpha) \tag{12.38}$$

基于已经进行的历史挤压，任何区间 $A\subset[0,2\pi)$ 都可以阐述为一个非确定的 I-状态。但是它的定义是关于相对方向，而不是原始状态。第 12.1.2 节讨论的算法可以应用于 $\mathcal{I}_{ndet}$。文献[397]给出一种后向搜索算法，它从一个非确定的 I-状态开始。规划通过 $\mathcal{I}_{ndet}$ 上的后向搜索来进行。在每次迭代中，可能的相对方向的区间 A 就增加直到最后到达所有 $\mathbb{S}^1$。

从试图每步都迫使 A 尽可能大这个意义上，算法是贪婪的。注意到图 12.52 的吸引域在厚度函数极小值处最大。因此只有极小值可以作为 α 的选择。令 B 表示函数 s 的原像。第一步，算法发现一个 α 使得 $B(\alpha)$ 最大（根据 $\mathbb{S}^1$ 中的长度）。令 α_0 表示这种相对方向，令 $A_0=B(\alpha_0)$。对于每个迭代子序列，令 A_i 表示 $[0,2\pi)$ 中满足下列不等式的最大区间：

$$|S(A_{i-1})|<|A_i| \tag{12.39}$$

其中 $|\cdot|$ 表示区间长度。这意味着存在一个挤压序列，$S(A_{i-1})$ 中的任意相对方向都可以用一个挤压迫使其进入 A_i。重复进行这种迭代就生成 A_{-1}、$A_{-2}\cdots$，直到不再满足式（12.39）的条件。文献[397]指出：对于任意多边形工件，A_i 区间增大直到得到所有 $\mathbb{S}^1$。

假设已经计算序列 $A_{-k},\cdots,A_0$。这必须变换成用机器人体坐标系表示的一个规划。重新获得 k 步行动序列 $(u_1,\cdots,u_k)$ 的方法是：

$$u_i=s(\beta_{i-1})-a_i-\frac{1}{2}(|A_{i-k}|-|S(A_{i-k-1})|)+u_{i-1} \tag{12.40}$$

且 $u_{-k}=0$ [397]。式（12.40）中的每个 a_i 都是 A_i 的左端点。在排列中有某些选择自由度，式（12.40）的第三项选择中间行动来提高关于方向误差的鲁棒性。文献[197]证明了对于一个有 n 条边的工件来说，需要挤压操作的次数不会超过 $O(n)$，利用该证明可以知道完备算

法的运行时间为 $O(n^2)$。

例 12.9（挤压一个矩形） 图 12.53 显示了一个简单的规划实例，它要求两次挤压来对放置在任意方向的矩形工件进行定向。图中显示了规划的四种不同的执行，每列表示一种。第一次挤压后，工件的方向是 $\pi/2$ 的倍数。第二次挤压后，方向已知。即使看起来每次执行都不同，也不需要进行反馈，因为 I-状态没有包括传感器信息。 ■

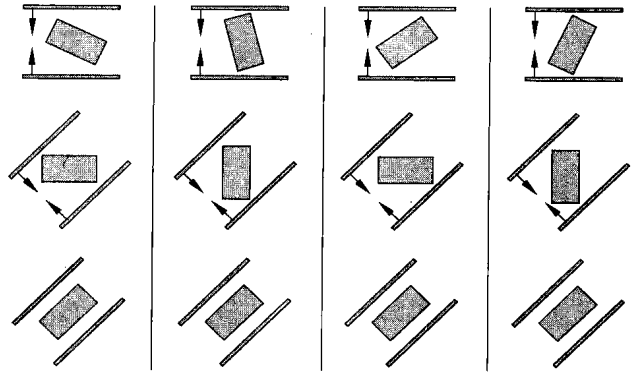


图 12.53 两步挤压规划[397]

进一步阅读

很容易将本章的内容扩展到关于有传感器不确定性的规划中。本章介绍了几个关键论题，但是还有大量的内容没有被介绍。为让读者进一步的阅读，这里给出一些建议。

因为第 12.1 节将 I-空间转换为一个平常状态空间，所以第 10 章中的许多方法和参考文献都是可用的。对于 POMDPs，大量的工作已经在运算和随机控制论中[567, 658, 717, 900]和最近的人工智能中[497, 650, 651, 740, 775, 794, 806, 808, 838, 1002, 1003]得到发展。许多算法压缩或者近似 $\mathcal{I}_{prob}$，产生的可能是非最优解，但是这能够处理状态比较多的问题。

第 12.2 节中的定位问题是机器人学中的最基本的问题，因此有数以百计的参考文献。文献[299, 344]考虑了在一个图中的定位问题。组合定位是基于文献[300, 418]的。由于对称性，所以会出现模糊[80]。文献[755]提出了使用较少感测能力的组合定位问题。关于概率定位的进一步阅读内容，参见文献[43, 260, 424, 450, 488, 496, 552, 624, 625, 757, 828, 888, 889, 962]。在文献[935, 936]中，定位不确定性用传感器不确定性的场来表示，该场是一个衍生 I-空间。

第 12.3 节的内容有许多来源。对于更多第 12.3.1 节中的关于迷宫搜索方法及其扩展到对图的探索，参见文献[121]。可分辨性和鹅卵石问题也出现在文献[89, 288, 289, 671, 843, 944]中。对于更多关于竞争率和在线导航的组合方法，参见文献[118, 262, 272, 334, 377, 510, 540, 677, 771, 814]。

更多关于 Stentz 算法及其相关工作，参见文献[546, 914]。一个探测地形的多分辨率方法出现在文献[764]中。对于关于 bug 算法的资料，参见文献[508, 571, 595, 669, 670, 812, 884]。文献[220, 221]将以传感器为基础的规划工作和基于广义 Voronoi 图联系起来，相关的还有文献[831]。文献[943, 944, 945]介绍了缺口导航树。对于关于最小地图的其他工作，

参见文献[487, 827, 875]。文献[371, 592, 885]考虑基于地标的导航。

有大量的文献介绍关于地图和定位问题的概率方法，其中有些被称为 SLAM[942]，也可参见文献[184, 223, 277, 720, 774, 982]。最早的工作之一出现在文献[898]中。动态规划在这种情形下的最早应用出现在文献[587]中。文献[161]中演示了 SLAM 技术。对于 EM 算法的介绍，参见文献[108]；文献[271, 977, 978]讨论了它的收敛性。更多关于可移动机器人的内容，参见文献[146, 298]。

第 12.4 节的内容主要是基于文献[417, 615]的。追逃问题首先在微分对策理论中得到研究[60, 425, 480]。文献[776]介绍了在一个图中的追逃问题，相关的理论分析出现在文献[107, 582, 717]中。文献[932]介绍了基于可视性的追逃问题，完备的算法首次出现在文献[614]中。对于在一个简单多边形中的单个追赶者来说，文献[772]给出了一种运行时间为 $O(n^2)$ 的算法。考虑弯曲环境、光束和其他情况下的变化形式出现在文献[210, 256, 306, 606, 621, 748, 890, 891, 931, 933, 981]中。文献[617]讨论了三维空间中的追逃问题。对于涉及很少的感测和没有先验地图的情况，参见文献[419, 506, 812, 843, 988]。跟踪有障碍物下的移动目标[83, 403, 404, 605, 726, 731]和没有障碍物下的移动目标[325, 473, 871]的问题都与追逃问题密切相关。对计算可视性信息的组合算法的综述，参见文献[759]。艺术品走廊和传感器放置问题也是相关的[143, 758, 876]。在可视复形[799]和形态图[785]中也出现了双切事件，它们与基于可视性的数据结构有关。

第 12.5 节是受到文献[285, 313, 323, 398, 662, 967]的启发。文献[684]对这些工作进行了综述。文献[150, 151, 607]考虑了概率原像规划。文献[351]考虑了可视化原像。文献[147, 148]对操作不确定性进行了仔细分析。更多原像规划的内容，参见文献[591, 593]。误差检测和恢复（EDR）用到许多原像规划的思路，但是它允许在执行过程中发生可确定的误差，所以能够解决更多的问题[283, 286, 287]。文献[142, 285, 489, 681, 683, 779]也考虑了适应性运动。文献[318]研究了 C-空间中摩擦力的影响。更多关于对工件的定向，参见文献[177, 324, 396, 397, 813, 969]。关于更多形式的非抓握操作，参见文献[12, 14, 112, 320, 673, 674, 922]。文献[682]介绍了“虚拟土壤原理”的概念，该思路后来出现在文献[842]和 iRobot 公司的 Roomba 自主真空吸尘器中。

习　题

1. 对于图 12.1(a)中的环境，给出行动序列（L,L,F,B,F,R,F,F）的非确定的 I-状态，如果初始状态是机器人在位置 3、面向北，并且初始 I-状态是 $\eta_0 = X$。

2. 描述如何应用图 10.6 中的算法来设计一个信息反馈规划，该规划有一个栅格的地图并可实现定位。

3. 假设在图 12.54 所示的环境中用例 12.1 中的运动和感测模型来操作一个机器人。设计一个尽可能简单而又能够定位机器人的信息反馈规划，不管初始位置如何。假设初始条件 $\eta_0 = X$。

4. 证明机器人在第 12.3.1 节的迷宫搜索算法中能够使用纬度和方向信息来探测每个障碍物边界的独特点。

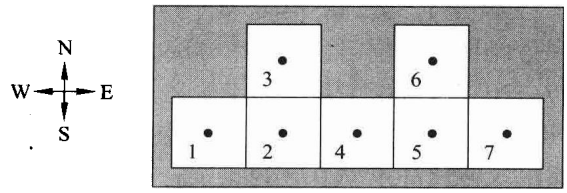


图 12.54 基于栅格定位的一个环境

5. 再次假设机器人被放置在如图 12.12 所示的六个环境之一中。它初始时处在右上角胞腔面向北，但是初始条件为 $\eta_0 = X$。当机器人执行行动序列（F,R,B,F,L,L,F）时，请确定传感器观测序列和非确定的 I-状态序列。

6. 证明第 12.3.1 节的迷宫搜索算法中的计数器可以被两个鹅卵石所取代，机器人仍然能够通过模拟计数器来解决问题。机器人可以将一个鹅卵石放在一个格片上，当机器人在该格片上的时候检测它们，然后将它们带到其他格片上。

7. 继续图 12.23 所示的轨迹，直到用 Bug2 策略到达目标。

8. 试说明应用于图 12.26 的 lost-cow 问题的双螺旋运动的竞争率是 9。

9. 推广 lost-cow 问题使得从目前母牛的位置产生 n 个栅栏（原始问题 $n=2$）：

（a）如果母牛被告知门就沿着一个未知的栅栏且不超过一个胞腔的距离，那么你认为最好的规划的竞争率是多少？

（b）假设母牛不知道离门的最大距离。提出一个规划来解决问题并建立它的竞争率。

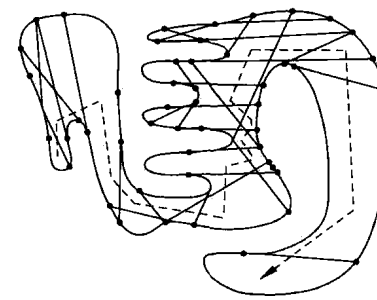


图 12.55 在一个初始未知环境中机器人所跟随的路径；机器人到达右下角

10. 假设一个点机器人落入如图 12.42 所示的环境中。显示当机器人沿着图 12.55 所示的路径运动时所得到的缺口导航树。

11. 构造一个例子，其中得到 Bug1 的最坏情况的界式（12.25）。

12. 某些环境如此复杂以至于在追逃问题中它们要求多次访问相同区域。为具有全向可视性的单个追赶者寻找图 12.56(a)中问题的一个解。

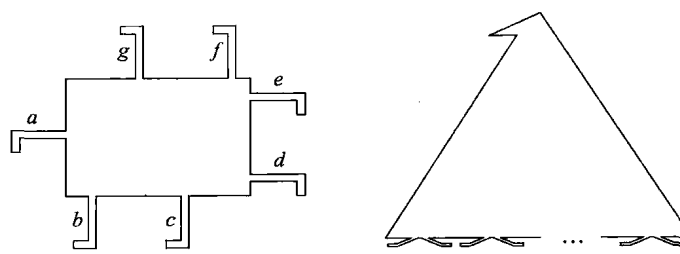


图 12.56 两个涉及再次受到污染的追逃问题

13. 为具有全向可视性的单个追赶者寻找图 12.56(b)中问题的一个追逃解，其中在多边形的底部可能出现任意双“脚”。

14. 证明对于一个多边形环境来说，如果有三个点 p_1、p_2 和 p_3，并且对于这些点来说 $(V(p_1))$、$(V(p_2))$ 和 $(V(p_3))$ 分别是两两不相交的，那么问题要求多于一个追赶者。

15. 证明第 12.5.2 节中的挤压算法的厚度函数的顶点不超过 $O(n^2)$ 个。给出实现这个界限的多边形序列。对于一个规则多边形，会发生什么？

16. 给出状态-大自然传感器映射的式（12.8）和式（12.9）的版本。

17. 给出基于历史的传感器映射的式（12.8）和式（12.9）的版本。

18. 详细描述第 12.3.1 节中的迷宫搜索中所用的 *I*-映射。指出该例子是如何显著地减少 *I*-空间的。得到的是一个充分的 *I*-映射吗？

19. 详细描述 Bug1 算法中所用的 *I*-映射。得到的是一个充分的 *I*-映射吗？

20. 假设几组点机器人在一个简单多边形中运动。每个机器人有一个全向可视性传感器，它想跟踪每个阴影区域的信息。对于每个机器人组和阴影区域，它想记录三种可能性之一：（1）区域中没有机器人组的成员；（2）可以有一个或者多个成员；（3）至少有一个成员。

（a）基于标记捕获相应信息的缺口来定义一个非确定的 *I*-空间。*I*-空间应该关于一个机器人来定义（每个机器人有它自己的 *I*-空间）。

（b）当机器人在环境中运动并观察其他机器人的时候，设计一个算法来跟踪非确定的 *I*-状态。

21. 回顾图 12.40 所示的连通走廊序列。尝试修改多边形使得需要相同数量的追赶者，但是多边形的边很少。尝试使用尽可能少的边。

实现

22. 解决第 12.2.3 节二维栅格的概率被动定位问题。实现你的解并在多个例子中证明它们。

23. 实现第 12.1.3 节中所描述的精确值迭代方法来计算最优尚需代价函数。测试关于几个小例子的实现。你能够产生多大的 K、Θ 和 Ψ？

24. 设计并实现一个图搜索算法，算法在 $\mathcal{I}_{ndet}$ 上进行搜索来进行二维栅格上的机器人定位。用几个例子来测试算法。试设计能够提高性能的搜索启发。

25. 实现 Bug1、Bug2 和 VisBug 算法（半径不受限）。设计一组好的例子来阐述它们的相关优势和不足。

26. 用软件实现计算当机器人在一个栅格中运动时的进行定位的概率 *I*-状态。

27. 实现第 12.3.4 节单连通环境下的方法，并在模拟多边形环境中证明它。

28. 在一个简单多边形中，实现一个追赶者的追逃算法。

29. 实现第 12.5.2 节中提出的工件挤压算法。

第Ⅳ部分　微分约束条件下的规划

第Ⅳ部分是第Ⅱ部分的延伸。在阅读第Ⅳ部分之前，一般情况下不需要阅读第Ⅲ部分。在第Ⅱ部分的模型和方法中，在没有障碍物的情况下，很容易确定任意两个位形之间的路径。例如基于采样的路线图方法假设两个邻近的位形可以用位形空间中的一条“直线”来连接。从限制在可允许位形的集合这个意义上，路径上的约束是全局的。

接下来的章节将介绍微分约束，它限制每个点上的允许速度。这些可以被认为是局部约束，这与由于障碍所引起的全局约束不同。诸如平滑要求等一些弱的微分约束曾在第 8 章出现过。第Ⅳ部分将更加详细地介绍微分约束。

微分约束无处不在。在机器人学中，几乎所有的问题都涉及来自于运动学和动力学的微分约束。一种处理微分约束的方法是在规划过程中忽略这些约束，然后在问题的改进中再处理它们。这种方法对应于第Ⅱ部分在机器人学中所应用的技术，而后采用控制技术来确保尽可能近地执行所计算的路径。但是这种方法只是一种可行的方法，比这更好的方法是在规划过程中考虑微分约束。这就产生了服从机械系统自然运动的规划。

第 13 章的思路与第 3 章相似。它将阐述如何构造和表示有微分约束的模型，第 3 章对几何模型做过同样的工作。该章通过对规划算法所用的大量模型进行分类，提出了第Ⅳ部分的背景和意义。微分模型通常表示为 $\dot{x} = f(x,u)$，它对应于状态转移方程 $x_{k+1} = f(x_k, u_k)$。因此第 13 章的焦点集中于定义转移函数。

第 14 章为涉及微分约束的问题引入基于采样的规划算法。这里没有关于组合算法的内容，因为它们仅存在于极端受限情况。微分约束似乎会破坏组合方法所需要的几乎所有好的特性。这里不研究完备算法，而是将焦点集中在分辨率完备的规划算法上。这可以通过三个空间的离散化来完成（状态空间、行动空间和时间），而第 5 章的分辨率完备性仅涉及 C 空间的离散化。这里主要的论题是扩展第 5.4 节的递增采样和搜索框架、扩展第 8 章的反馈运动规划以及为轨迹规划而研究解耦方法。

第 15 章介绍从控制论而来的非常有力的思路和工具。利用第 15 章的内容能够加强第 14 章的规划方法。这两章是互补的，第 14 章主要针对的是算法，而第 15 章主要是关于数学论述。第 15 章主要讨论系统的稳定性、最优性概念（Hamilton-Jacobi-Bellman 方程和庞特里亚金最小值原理）、轮式车辆的最短路径、非完整系统理论和非完整操控方法。非完整的概念来自于力学，它是指不能完全积分来去掉状态变量时间导数的微分约束。

第13章 微分模型

本章介绍与状态转移方程 $x_{k+1}=f(x_k,u_k)$ 对应的连续时间版本。在连续状态空间 X 上（假设是平滑流形），可以定义 $\dot{x}=f(x,u)$，它看起来与离散情况相似。它也被称为状态转移方程。在控制论中它被称为系统（控制系统的简称）。本章没有障碍区域。在第 14 章介绍规划算法时将会出现障碍物。在连续时间中，状态转移函数 $f(x,u)$ 产生一个速度，而不是下一个状态。因为转移不再是离散的，谈论“下一个”状态是没有意义的。满足微分方程的未来状态通过速度的积分来得到。因此可以仅指定速度。这依赖于第 8.3 节介绍的切空间和向量场的概念。

本章提出一些在第 14 章规划算法中将会用到的模型。对于 X 是一个或多个物体的 C 空间这种情况，第 13.1 节将研究其微分约束。这些约束普遍出现在轮式车辆中（例如汽车不能侧向移动）。为了表示动力学，也需要对加速度进行约束。因此第 13.2 节将介绍相空间，它能使得任何动力学问题都被表示成一个增广状态空间上的速度约束。这将高阶导数降到仅有一阶，只是增加了状态的维数。第 13.3 节将介绍牛顿-欧拉力学的基础，并表达自由刚体的动力学。第 13.4 节将介绍来自于分析力学的一些概念，包括拉格朗日和哈密顿函数。该节也将提出物体运动链的动力学模型，该模型应用在典型的机器人操作臂上。第 13.5 节将介绍多决策者的微分模型。

13.1 位形空间上的速度约束

本节假设 $X=C$，它是一个或多个刚体的一个 C 空间，如第 4.2 节所定义的。本节的微分模型都可表示为 C 中每个点处可允许速度集合上的约束。这导致一阶微分方程，因为仅有速度约束，这与加速度或者高阶导数情况不同。

为了仔细讨论速度，假设 C 不仅是第 4.1.2 节所定义的拓扑流形，而且还是第 8.3.2 节所定义平滑流形。记住 $C=\mathbb{R}^2$ 和 $C=\mathbb{R}^3$ 这些情况是有益的。速度的定义很简单，不需要借助于流形，并且维数足够低以至于概念可以被形象化。

13.1.1 隐含表示与参数表示

有两种一般方法来表示微分约束：参数的和隐含的。也有一些与参数方法和隐含方法相并行的方法。参数表示通常比较容易理解，在应用中也更容易使用。隐含表示则更加一般，但是通常难以利用。它们之间的直观区别在于隐含表示通常用来表达被禁止的速度，而参数表示直接表达允许的速度。本章都用参数表示，但理解这两种方法是很重要的。

13.1.1.1 隐含表示

平面情况 为进行阐述，假设$C=\mathbb{R}^2$。位形表示为$q=(x,y)\in\mathbb{R}^2$，速度表示为$(\dot{x},\dot{y})$。每个$(\dot{x},\dot{y})$都是切空间$T_q(\mathbb{R}^2)$的一个元素，它是每个(x,y)处的一个二维向量空间。考虑可以施加的约束的类型。在每个$q\in\mathbb{R}^2$，对速度集合的限制能够产生某集合$U(q)\subset T_q(\mathbb{R}^2)$。对于所有$q\in\mathbb{R}^2$，参数的和隐含的表示都可以用来表示$U(q)$。

这里有一些有趣而且简单的约束。$U(q)$是满足约束的$T_q(\mathbb{R}^2)$的子集。

（1）$\dot{x}>0$：在这种情况下，假设你在向正x轴方向流动的河流中划着一只小船。在y方向你可以得到任何速度。但是你不能流向相反的方向。这意味着所有积分曲线在x方向上随时间单调增加。

（2）$\dot{x}\geqslant 0$：该约束允许你在x方向停下来。可以得到垂直于水流的速度（例如(0,1)在正y方向引起单位速度的运动）。

（3）$\dot{x}>0$，$\dot{y}>0$：在这种约束下，积分曲线在x和y两个方向上必须单调增加。

（4）$\dot{x}=0$：前三个例子中，可允许速度的集合为两维。在$\dot{x}=0$约束下，可允许速度的集合仅为一维。对于任意$\dot{y}\in\mathbb{R}$，形如$(0,\dot{y})$的所有向量都是允许的。这意味着在x方向没有可允许的运动。从任意(x_0,y_0)开始，积分曲线的形式将会是$(x_0,y(t))$，$t\in[0,\infty)$，$(x_0,y(t))$将每条积分曲线限制在一条垂线上。

（5）$a\dot{x}+b\dot{y}=0$：该约束本质上与上一个是相同的。不同之处在于通过选择a和b，现在的运动可以沿着任何平行线族。例如，如果$a=b=1$，那么仅允许对角线运动。

（6）$a\dot{x}+b\dot{y}+c=0$：该约束与前面一个相似。积分曲线仍然被限制在直线之间，但是常数$c\neq 0$这个附加条件意味着速度矢量(0,0)是不允许的。沿着一条直线运动是允许的，但是无法停下来。

（7）$\dot{x}^2+\dot{y}^2\leqslant 1$：该约束在第8章中用到过。它对可行运动规划问题的解的存在性没有影响，因为任意方向的运动都是允许的。该约束只施加了一个最大速度约束。

（8）$\dot{x}^2+\dot{y}^2\geqslant 1$：该约束允许沿任意方向、以任意大于1的速度所进行的运动。不可能停止或者速度低于单位速度。

也有一些其他形式的约束，包括为每个$U(q)$在$\mathbb{R}^2$中所定义的非常复杂的区域。如果忽略$\dot{x}$和$\dot{y}$表示导数这样一个事实，那么第3.1节的几何模型的概念甚至可以用来定义每个q处的复杂约束。实际上，用$\dot{x}$和$\dot{y}$表示的约束只是第3.1.2节介绍的半代数模型的简单例子。只是用$\dot{x}$和$\dot{y}$代替x和y而已。

如果在每个q处都存在某开集合O使得$(0,0)\in O$和$O\subseteq U(q)$，那么对可行运动规划问题的解的存在性没有影响。所有方向上的速度都是允许的。对于任何平滑流形上的速度约束，这都是成立的[925]。

目前已经用相同的方式对每个$q\in\mathbb{R}^2$处的速度进行了约束，这意味着对于所有$q\in\mathbb{R}^2$，$U(q)$都是相同的。这种约束的形式是$g(\dot{x},\dot{y})\bowtie 0$，其中$\bowtie$表示=、<、>、$\leqslant$或$\geqslant$，$g_i$是从$\mathbb{R}^2$到$\mathbb{R}$的函数。典型地，关系“=”将$U(x)$的维数降低到1，而其他的关系通常并不使之发生变化。

现在考虑约束$\dot{x}=x$。这导致了每个$q\in\mathbb{R}^2$处可允许速度的不同的一维集合$U(q)$。对

于任何 $\dot{y}\in\mathbb{R}$，在每个 $q=(x,y)$ 上可允许速度的集合的形式必须是 $(x,\dot{y})$。这意味着当 x 增加时，x 方向的速度必须成比例地增加。从任意正 x 值开始，无法运动到 y 轴。但是从 y 轴开始，积分曲线将总能够保持在 y 轴上。这种约束通常可以表示为 $g(x,y,\dot{x},\dot{y})\bowtie 0$，它与 x 或 y 有关系。

一般位形空间 在一般 C 空间上也可以用相同的方式考虑速度约束。假设 C 是一个平滑流形（第 4 章的流形并不要求是平滑的，因为那里不需要导数）。所有约束都用第 8.3.2 节定义的坐标邻域来表示。对于表示微分模型，这实际上使得一个 n 维流形看起来非常像 $\mathbb{R}^n$。这可以隐含地理解为坐标的变化有时是需要的，但并不使得约束的表示更加复杂，那么有可能忽略流形的一些技术性，而认为约束好像是在 $\mathbb{R}^n$ 上一样。

现在考虑将速度约束放置在 C 上。假设如果允许任意半代数模型，那么速度约束会变得多么复杂。C 上的速度约束将会和任意 C_{obs} 一样复杂。甚至不必使用代数本原。一般情况下，约束可以表示为：

$$g(q,\dot{q})\bowtie 0 \tag{13.1}$$

其中 $\bowtie$ 仍然表示=、<、>、≤或≥。甚至在去除某些关系以后，依然能够保持应有的表达能力。例如形式为式（13.1）的任何约束可以表示成 $g(q,\dot{q})=0$ 和 $g(q,\dot{q})<0$ 的一种组合。但是这里允许所有这些关系以使表述更加简单。

以式（13.1）形式表达的约束称为是隐含的（implicit）。就像第 3 章和第 4 章所讲的，很难得到隐含方程解的参数表示。例如从第 4.4 节就可以清楚地看出这一点，在那里很难刻画满足封闭约束的位形集合。尽管如此，如果能够得到约束的参数表示，那么对研究规划算法是非常有利的。非常幸运，来自于机器人、车辆以及其他机械系统的大多数约束都可以用参数的形式表示。

13.1.1.2 参数化的约束

表达速度约束的参数化方法给出了对 $U(q)$ 的一种不同解释。不直接对应于速度，将每个 $u\in U(q)$ 视为一个抽象行动向量。那么通过将行动向量映射到 $T_q(C)$ 的函数就可以得到可允许速度的集合。这就产生了位形转移方程（或者系统）：

$$\dot{q}=f(q,u) \tag{13.2}$$

其中 f 是第 2.1 节介绍的状态转移函数的连续形式。注意到式（13.2）实际上表示的是 n 个标量方程，其中 n 是 C 的维数。将系统作为向量意义下的单个方程。通常对于所有 $q\in C$，$U(q)$ 是固定的。所以除非特别说明，通常都做这种假设。在这种情况下，固定的行动集合表示为 U。

有两种方法来阐述式（13.2）：

（1）**切空间的子空间**：如果 q 固定，那么 f 将 U 映射到 $T_q(C)$。这就对 q 点的可允许速度进行了参数化，因为对于每个 $u\in U(q)$ 都得到了速度向量 $f(q,u)$。

（2）**向量场**：如果 u 固定，那么 f 可以认为是将每个 $q\in C$ 映射到 $T_q(C)$ 的一个函数。这意味着 f 为每个固定的 $u\in U$ 在 C 上定义了一个向量场。

例 13.1（$\dot{q}=f(q,u)$ 的两种解释） 假设 $C=\mathbb{R}^2$，这在每个 $q=(x,u)\in\mathbb{R}^2$ 点上产生一个二维速度向量空间。令 $U=\mathbb{R}$，并且 $\dot{q}=f(q,u)$ 定义为 $\dot{x}=u$ 和 $\dot{y}=x$。

为了得到 $\dot{q}=f(q,u)$ 的第一种解释，保持 $q=(x,y)$ 固定；对于每个 $u\in U$，得到一个速度向量 $(\dot{x},\dot{y})=(u,x)$。$q=(x,y)$ 上所有可允许速度向量的集合是：

$$\{(\dot{x},\dot{y})\in\mathbb{R}^2\,|\,\dot{y}=x\} \tag{13.3}$$

假设 $q=(1,2)$，那么形式为 $(u,1)$ 的任何向量都是允许的，对于任意 $u\in\mathbb{R}$。

为得到第二种解释，保持 u 固定。例如 $u=1$。得到 $\mathbb{R}^2$ 上的向量场 $(\dot{x},\dot{y})=(1,x)$。 ■

当定义位形转移方程时，指定 U 是非常重要的。我们允许但尽量避免行动集合对 q 的依赖。对于任何 U 及每个 $q\in C$，通过定义：

$$U(q)=\left\{\dot{q}\in\mathbb{R}^n\,|\,\exists u\in U,\dot{q}=f(q,u)\right\} \tag{13.4}$$

任何微分约束 $\dot{q}=f(q,u)$ 都还可以表示为 $\dot{q}=u$。在该定义中，$U(q)$ 不必是 U 的一个子集。但是使用 $\dot{q}=f(q,u)$ 这种形式并对所有 q 都保持相同的 U，这样做通常更加方便。对 U 的通常解释是它是从 C 空间中的任意点都可以应用的固定行动的一个集合。

在平常的运动规划中，并不需要提及位形转移方程。这个问题在第 8.4 节讨论过。只要 U 包含一个含原点的开子集，那么任何方向的运动都是允许的。基本运动规划的位形转移方程只是 $\dot{q}=u$。因为对方向并无约束，所以也就没有明确提及它。对于本章将要论述的模型来说，将会施加限制可能方向的速度约束。那么就要求能够处理微分模型的规划算法，这是第 14 章的内容。

13.1.1.3 从隐含形式转换到参数形式

在表达微分约束的隐含方式和参数方式之间存在权衡。隐含表示更加一般，但是参数形式更有用处，因为它明确地给出了可能的行动。因为这个原因，通常希望能够从隐含形式产生参数表示。在理论上这是可能的。就像稍后所解释的，它是隐函数定理的一个结果。遗憾的是，这种转换的理论存在性对实际变换并无帮助。在许多情况下，无法得到参数表示。

为了建模机械系统，最简单的方式是用隐含形式表示约束，然后从隐含表示中产生参数表示 $\dot{q}=f(q,u)$。到目前为止在隐含表示中没有出现 u。因为 u 是行动，当产生参数表示时需要指定它。为了理解这个问题，首先假设隐含形式下的所有约束都是 $\dot{q}$ 的线性方程：

$$g_1(q)\dot{q}_1+g_2(q)\dot{q}_2+\cdots+g_n(q)\dot{q}_n=0 \tag{13.5}$$

它被称为 Pfaffian 约束。这些约束仅在 q 已知的情况下才是线性的。在目前的讨论中可以假设 q 固定为某已知值，这意味着式（13.5）中的每个系数 $g_i(q)$ 都是常数。

假设给定 k 个 Pfaffian 约束，$k\leqslant n$，并且它是线性无关的[1]。回忆求解线性方程的标准方法。如果 $k=n$，那么存在唯一解。如果 $k<n$，那么存在无穷多解，这形成了一个 $(n-k)$ 维的超平面。不可能有 $k>n$，因为线性无关方程不会超过 n 个。

如果 $k=n$，那么对于每个 $q\in C$，仅有速度向量满足约束。因此一个向量场可以由约束产生，并且从规划的角度而言问题并不令人感兴趣，因为没有速度选择。如果 $k<n$，那么 $\dot{q}$ 的 $n-k$ 个分量可以独立进行选择，然后计算其余 k 个分量以满足 Pfaffian 约束（这可以用线性代数来完成，如奇异值分解[401,961]）。可以独立进行选择的 $\dot{q}$ 的分量可以被认为是 $n-k$ 个标量行动，它们一起形成了一个 $(n-k)$ 维的行动向量 $u=(u_1,\cdots,u_{n-k})$。不失一般

1 如果系数构成一个 $k\times n$ 矩阵，那么它的秩是 k。

性，假设 $\dot{q}$ 的前 $n-k$ 分量由 u 指定。那么位形转移方程可以写为：

$$\begin{aligned}
\dot{q}_1 &= u_1 \\
\dot{q}_2 &= u_2 \\
&\vdots \\
\dot{q}_{n-k} &= u_{n-k} \\
\dot{q}_{n-k+1} &= f_{n-k+1}(q,u) \\
\dot{q}_{n-k+2} &= f_{n-k+2}(q,u) \\
&\vdots \\
\dot{q}_n &= f_n(q,u)
\end{aligned} \tag{13.6}$$

其中每个 f_i 都是 u 的一个线性函数，它们是这样从 Pfaffian 约束产生的：将从 1 到 $n-k$ 的 $\dot{q}_i$ 用 u_i 代替，然后求解 $\dot{q}$ 的其余分量。对于 q 的某些值来说，约束可能变成线性相关的。这只弱化了约束，这意味着即 u 的维数在失去无关性的任意 q 处可以增加。这种点通常是孤立的，将不予进一步考虑。

例 13.2（Pfaffian 约束） 假设 $C=\mathbb{R}^3$，并有一个式（13.5）形式的约束：

$$2\dot{q}_1 - \dot{q}_2 - \dot{q}_3 = 0 \tag{13.7}$$

对于该问题，有 $n=3$ 和 $k=1$。因为 $n-k=2$，所以有两个行动变量。位形转移方程为：

$$\begin{aligned}
\dot{q}_1 &= u_1 \\
\dot{q}_2 &= u_2 \\
\dot{q}_3 &= 2u_1 - u_2
\end{aligned} \tag{13.8}$$

其中将 u_1 和 u_2 分别代入式（13.7）中的 $\dot{q}_1$ 和 $\dot{q}_2$，然后求解 $\dot{q}_3$，就得到最后一个分量。

式（13.7）中给定的约束甚至不依赖于 q。相同的思路可以用到更一般的 Pfaffian 约束上，如：

$$(\cos q_3)\dot{q}_1 - (\sin q_3)\dot{q}_2 - \dot{q}_3 = 0 \tag{13.9}$$

按照相同的过程，位形转移方程变成：

$$\begin{aligned}
\dot{q}_1 &= u_1 \\
\dot{q}_2 &= u_2 \\
\dot{q}_3 &= (\cos q_3)u_1 - (\sin q_3)u_2
\end{aligned} \tag{13.10}$$

■

目前的思路很自然地扩展到在 $\dot{x}$ 中不是线性的这样的等式约束。在每一个 q，如果约束函数在 q 点的雅可比矩阵 $\partial(g_1,\cdots,g_k)/\partial(\dot{q}_1,\cdots,\dot{q}_n)$ 的秩为 k，那么能够保证存在一个 $(n-k)$ 维的行动集合 $U(q)$。这服从隐函数定理[511]。

假设除等式约束以外，还有 $g(q,\dot{q})\leqslant 0$ 形式的不等式约束。使用以前的概念，那么对于所有 i，$1\leqslant i\leqslant n-k$，仍然可以直接指定行动为 $\dot{q}_i=u_i$。如果没有不等式约束，那么就没有对 u 的约束，这意味着 $U=\mathbb{R}^n$。因为 u 是作为物理系统的输入，所以通常对它是有约束的。在一个物理系统中，所消耗的能量与 u 呈比例关系。代入 $\dot{q}_i=u_i$ 以后，不等式约束表明了对 u 的限制。这些限制用 q 以及 $\ddot{q}$ 的其余分量 $\dot{q}_{n-k+1},\cdots,\dot{q}_n$ 来表达。对于许多问题，不等式约束足够简单以至于可以直接在 U 上产生约束。例如，如果 u_1 表示汽车的标量加速度，那么它可以有诸如 $|u_1|\leqslant 1$ 这样简单的界。

有时还会发生的复杂情况是，行动变量可能已经在等式约束 $g(q,\dot{q},u)=0$ 中指定了。在这种情况下，仍然将 q 假设成固定的。如果有 k 个无关约束，那么根据隐函数定理，求解 $\dot{q}$ 来产生 $\dot{q}=f(q,u)$（尽管在理论上是可能的，但是在实际上可能非常困难）。如果 q 点的雅可比矩阵 $\partial(f_1,\cdots,f_n)/\partial(u_1,\cdots,u_k)$ 的秩为 k，那么可以应用行动来在 $T_q(C)$ 中 k 维超平面上产生任意速度。如果 $k=n$，那么存在足够的无关行动变量来克服约束的限制。通过选择 u 就可以实现 $T_q(C)$ 中的任意速度。这仅在没有 U 上的不等式约束的情况下才是事实。

13.1.2　轮式系统的运动学

在机器人技术中最常用的实例来自于车轮，对它们的要求是沿着它们所指的方向滚动。多数车轮被设计成不能侧滑的。这就对滚动车辆施加了一个速度约束。结果是它们的行动变量比自由度少。这种系统因此被称为欠驱动的（underactuated）。在许多情况下，有趣的是车辆进行能够克服约束的运动。例如汽车能够平行移位停靠在它们所能够到达的任意位置，如果所有四个车轮能够向任意方向旋转。这就产生了诸如非完整约束和短时局部可控性等非常正式的概念，这将在第 15.4 节介绍。

13.1.2.1　一辆简易汽车

最容易理解的实例是图 13.1 所示的简易汽车。我们都知道汽车不能侧向行驶，因为如果侧向行驶，则后轮不得不滑动，而不是滚动。这就是为什么很难平行移位停靠的原因。如果所有四个车轮都能够同时转向路边，那么停车就非常容易了。由于存在滚动约束，所以停靠简易车辆需要复杂的机动。

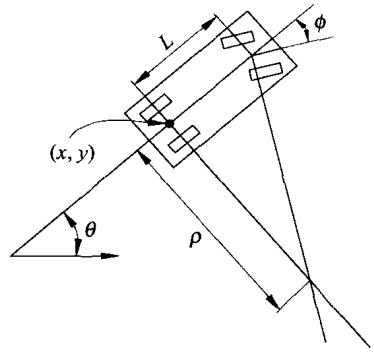


图 13.1　简易汽车有三个自由度，但是任何位形下的速度空间只是二维的

可以将汽车假设为在平面上运动的一个刚体。因此它的 C 空间是 $C=\mathbb{R}^2\times\mathbb{S}^1$。图 13.1 表明了几个与汽车有关的参数。位形用 $q=f(x,y,\theta)$ 表示。汽车的体坐标系原点在后轮轴的中间，x 轴指向汽车的主轴。令 s 表示汽车的（有符号的）速度[1]。令 ϕ 表示转向角（对于图 13.1 所示的车轮方向来说它是负的）。前后轮轴中间的距离用 L 表示。如果转向角固定为 ϕ，那么汽车做圆形运动，其中圆的半径为 ρ。可以从两个轴的交点来确定 ρ，如图 13.1 所示（轴之间的角度为 $|\phi|$）。

使用目前符号，任务是将汽车的运动表示成方程的集合：

$$\begin{aligned}\dot{x}&=f_1(x,y,\theta,s,\phi)\\ \dot{y}&=f_2(x,y,\theta,s,\phi)\\ \dot{\theta}&=f_3(x,y,\theta,s,\phi)\end{aligned}\tag{13.11}$$

在小的时间区间 Δt 中，汽车必须近似沿着后轮的指向运动。在 Δt 趋近于零的极限状态下，这表示 $\mathrm{d}y/\mathrm{d}x=\tan\theta$。因为 $\mathrm{d}y/\mathrm{d}x=\dot{y}/\dot{x}$ 以及 $\tan\theta=\sin\theta/\cos\theta$，该条件可以写为一个

1　有符号的速度似乎有点不符合习惯。如果汽车向后行驶，那么 s 是负的。s 的一个更准确的名字是在体坐标系的 x 轴方向上的速度，但是这太不方便。

Pfaffian 约束：

$$-\dot{x}\sin\theta+\dot{y}\cos\theta=0 \tag{13.12}$$

如果 $\dot{x}=\cos\theta$ 和 $\dot{y}=\sin\theta$，那么就满足约束条件，并且这个解的整倍数也是一个解，刻度因子直接对应于汽车的速度 s。因此位形转移方程的前两个标量分量是 $\dot{x}=s\cos\theta$ 和 $\dot{y}=s\sin\theta$。

接下来的任务是产生 $\dot{\theta}$ 的方程。令 w 表示汽车行驶的距离（速度积分）。如图 13.1 所示，如果转向角固定，ρ 表示行驶时后轮轴中心绘出的圆的半径。注意到 $\mathrm{d}w=\rho\mathrm{d}\theta$。由三角关系知道 $\rho=L/\tan\phi$，这意味着：

$$\mathrm{d}\theta=\frac{\tan\phi}{L}\mathrm{d}w \tag{13.13}$$

两边除以 $\mathrm{d}t$、使用 $\dot{w}=s$ 产生：

$$\dot{\theta}=\frac{s}{L}\tan\phi \tag{13.14}$$

现在已经对汽车的运动进行了建模，但是没有指定行动变量。假设速度 s 和转向角 ϕ 直接由行动变量 u_s 和 u_ϕ 指定。按照惯例，变量 u 将旧的变量名作为下标。这就容易在位形转移方程中对行动进行识别。得到二维行动向量 $u=(u_s,u_\phi)$。简易汽车的位形转移方程为：

$$\begin{aligned}\dot{x}&=u_s\cos\theta\\ \dot{y}&=u_s\sin\theta\\ \dot{\theta}&=\frac{u_s}{L}\tan u_\phi\end{aligned} \tag{13.15}$$

式（13.15）没有指定 $u=(u_s,u_\phi)$ 形式的行动的集合 U，那么转移方程尚不完整。首先假设任意 $u_s\in\mathbb{R}$ 都是可能的。那么转向角可能是怎样的？对于转向角 u_ϕ 来说，区域 $[-\pi/2,\pi/2]$ 足够大，因为任何其他值都等价于 $-\pi/2$ 和 $\pi/2$ 之间一个值。转向角 $-\pi/2$ 和 $\pi/2$ 是有问题的。为了产生 $\dot{x}$ 和 $\dot{y}$ 的表达式，假设汽车沿着后轮的指向运动。假设你在一辆三轮车上将前轮转到垂直于后轮的方向（$u_\phi=\pi/2$）。如果你还能够踩下踏板，那么三轮车将原地旋转。也就是说 $\dot{x}=\dot{y}=0$，因为后轮轴中心没有平移。

这种奇怪的行为对标准车辆来说是不允许的。后轮驱动的汽车在通过人行道时可能对前轮刹车。如果一辆前轮驱动的汽车也试图这样做，那么它的行为应该像三轮车，但是这通常是不可能的，因为当转向 $\phi=\pi/2$ 时，前轮将会与前轮轴发生冲突。因此简易汽车的最大转向角 $\phi_{\max}<\pi/2$，我们要求 $|\phi|\leqslant\phi_{\max}$。从图 13.1 中可以观察到最大转向角就意味着最小转弯半径 $\rho_{\min}$。对于三轮车来说，$\rho_{\min}=0$。当试图将车辆进行一个不合逻辑的调头时，你可能已经遇到最小转弯半径问题。有时将汽车调头而不驶离公路是很难的。

现在回到速度 u_s。在水平的人行道上，真实的车辆都有最大速度，随着速度的变化，车辆的行为也会发生显著的变化。例如，如果你想沿着最小转弯半径进行驾驶，那么速度不应该达到 140 千米/小时。速度越快，最大转向角越小。这就进入了动力学领域，在第 13.2 节中介绍相空间以后我们才会考虑动力学问题。所以含动力学的汽车模型将在第 13.2.4 节和第 13.3.3 节介绍。

这里隐含着一种情况：如果简易汽车慢速运动，那么可以忽略动力学。对速度可以施

加诸如$|u_s| \leqslant 1$这样的界限不会影响它所能够到达的位形。甚至速度约束为$u_s \in \{-1,0,1\}$也不会损坏可达性。但是需要注意诸如$0 \leqslant u_s \leqslant 1$这种界限。在这种情况下，汽车不能倒车。这就影响了可达位形的集合。假设一辆汽车面向墙而又不能倒车。当它行驶时，它被迫撞墙。

基于这种关于速度和转向角的考虑，有如下几种车辆模型。

三轮车：$U=[-1,1]\times[-\pi/2,\pi/2]$。假设如果$u_\phi=\pi/2$或者$u_\phi=-\pi/2$，前轮驱动的“汽车”能够在原地旋转。这对于简易汽车来说是不现实的。所产生的模型与式（13.18）中出现的简易独轮车相似。

- **简易汽车[599]**：$U=[-1,1]\times(-\phi_{\max},\phi_{\max})$。通过要求$|u_\phi| \leqslant \phi_{\max} < \pi/2$，可以得到最小转弯半径$\rho_{\min}=L/\tan\phi_{\max}$的汽车。
- **Reed-Shepp 汽车[817,924]**：进一步对简易汽车的速度进行限制使得$u_s \in \{-1,0,1\}$。[1] 这种模型使得u_s对应于三种离散“传动”状态——向后、停止或者向前。该模型的一个有趣的问题是，在没有障碍物的情况下，两个位形之间可能的最短路径（后轮轴中心在$\mathbb{R}^2$中驶过的）是什么？这将在第 15.3 节给予解答。
- **Dubins 汽车[296]**：从 Reed-Shepp 汽车中删除倒车速度$u_s=-1$后得到$u_s \in \{0,1\}$。该车在$\mathbb{R}^2$中最短路径与 Reed-Shepp 汽车有很大不同参见第 15.3 节。图 1.12 所示的汽车比 Dubins 汽车受到更多限制，因为它只能被迫向左转弯。

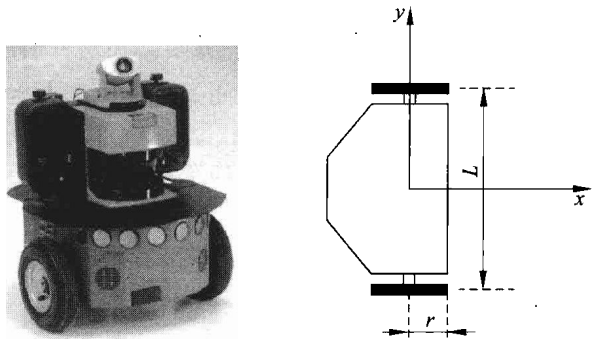
简易汽车的基本可控性问题已经得到研究。这些内容将在第 15.4 节介绍，但是有益的是，研究直观概念有助于对第 14 章规划算法的理解。简易汽车被认为是非完整的（nonholonomic），因为存在不能完全进行积分的微分约束。这意味着不将汽车位形限制在C的一个更低维的子空间上。只要有少量的空隙，Reed-Shepp 汽车就能够机动进入任意小的停车位。这种特性被称为短时局部可控性（small-time local controllability），我们将在第 15.1.3 节对其进行介绍。Dubins 汽车是非完整的，但是它不具有这种特性。假设如果没有倒车齿轮，平行移位停车是多么的困难。但是在一个没有障碍物的无限大的停车场中，Dubins 汽车能够到达任意位形。

13.1.2.2　差动机构

多数室内机器人不能像汽车那样运动。例如图 13.2(a)所示的可移动机器人。这是最常用的驱动室内机器人的方法。有两个主轮，每个主轮连接到它自己的发动机上。第三个车轮（在图 13.2(a)中不可见）放置在后方，它被动滚动且能够防止机器人的跌倒。

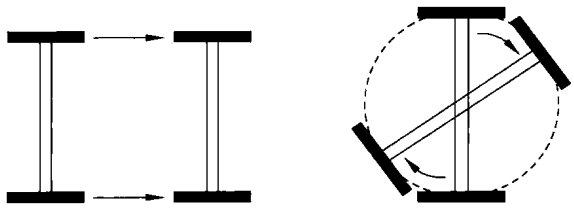
为了构造差动机构所产生的约束的简单模型，只需要两个车轮间的距离L和车轮半径r（见图 13.2(b)），行动向量$u=(u_r,u_l)$直接指定两个角速度。考虑在应用不同行动时机器人是如何运动的（见图 13.3）。如果$u_r=u_l>0$，那么机器人沿着车辆的指向向前运动。速度与r成正比。一般情况下，如果$u_l=u_r$，那么在t时间内行进的距离为rtu_l（因为tu_l是车辆总的角位移）。如果$u_l=-u_r\neq 0$，那么机器人顺时针旋转，因为车轮向相反方向转动。这启发了将体坐标系的原点放在车轮轴的中心。在这种安排下，如果车轮的旋转速率相同、方向相反，则不会发生平移。

1　在许多研究中，不包括速度$u_s=0$。它在这里出现以使我们可以定义适当的终止条件。



(a) (b)

图 13.2 （a）Pioneer 3-DX8（由 ActivMedia Robotics 提供：MobileRobots.com）以及其他一些可移动机器人采用了一个差动机构；除了两个驱动轮以外，还有一个转向轮放在后面的中间部位以防机器人翻倒；（b）一个普通差动机构机器人的参数



(a) (b)

图 13.3 （a）当两个车轮以相同的角速度运动时发生的是纯平移运动；（b）当两个车轮以相反速度运动时发生的是纯旋转运动

基于这些认识，位形转移方程是：

$$\begin{aligned}\dot{x} &= \frac{r}{2}(u_l + u_r)\cos\theta \\ \dot{y} &= \frac{r}{2}(u_l + u_r)\sin\theta \\ \dot{\theta} &= \frac{r}{L}(u_r - u_l)\,.\end{aligned} \tag{13.16}$$

平移部分包括 $\cos\theta$ 和 $\sin\theta$，就是像简易汽车一样，因为差动机构沿着驱动轮的指向进行运动。平移速度依赖于车轮平均角速度。为了更清晰地看见这种结果，考虑将一个车轮固定，而使另外一个旋转。初始时这使得机器人平移的速度是两个车轮都能旋转情况下速度的 1/2。旋转速度 $\dot{\theta}$ 与车轮角速度的变化成比例。机器人旋转速率随车轮半径的增加而线性增长，但是随车轮间距离的增加而线性降低。

有时更愿意对行动空间进行变换。令 $u_w = (u_r + u_l)/2$ 和 $u_\psi = u_r - u_l$。在这种情况下，u_w 是意味着“平移”的行动变量，u_ψ 是意味着“旋转”的行动变量。使用这些行动，位形转移方程变成：

$$\begin{aligned}\dot{x} &= r u_w \cos\theta \\ \dot{y} &= r u_w \sin\theta \\ \dot{\theta} &= \frac{r}{L} r u_\psi\end{aligned} \tag{13.17}$$

在这种形式下，位形转移方程类似于简易汽车的式（13.15）（设定 $u_\psi = \tan u_\phi$ 和 $u_s = ru_w$）。差动机构很容易模拟简易汽车的运动。对于差动机构，旋转速率的设置可以不依赖于平移速度。但是简易汽车的 $\dot{\theta}$ 表达式中包含速度 u_s。因此旋转速率依赖于平移速度。

回忆 Reed-Shepp 车和 Dubins 车的最短路径问题。同样的问题对于差动机构来说就不太有趣，因为差动机构引起车轴中心跟踪 $\mathbb{R}^2$ 中任意连续路径。如图 13.4 所示，它能够在任意两个位形之间运行，通过：（1）首先进行旋转使车轮指向目标位置，不产生平移；（2）平移到目标位置；（3）进行旋转指向所需要的方向，同样不产生平移。车轴中心行进的总距离总是两个期望位置之间在 $\mathbb{R}^2$ 中的欧氏距离。

由于坐标原点的放置，这种效果似乎有些奇怪。旋转似乎没有代价。如果速度保持固定[65]，那么这可以通过最优化车轮总转数或者所需要的时间来解决。假设 $u_r, u_l \in \{-1,0,1\}$。使人感兴趣的是确定在两个位形之间移动的最小时间，这将在第 15.3 节介绍。在那里将考虑机器人旋转所花费的代价，即使它不引起平移。

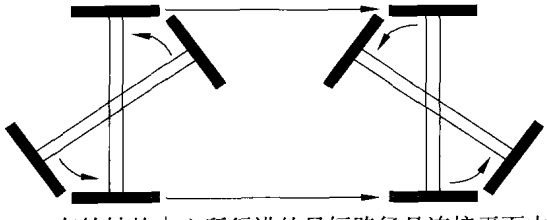


图 13.4 车轮轴的中心所行进的最短路径是连接平面中初始位置和目标位置的直线；旋转则并不花费代价

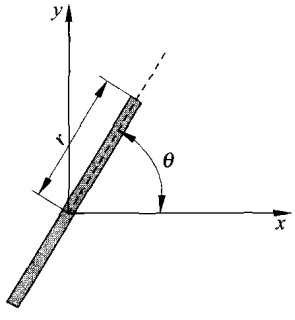


图 13.5 从顶部看，独轮车模型有一个改变车轮方向 θ 的行动 u_ω

13.1.2.3 一个简易的独轮车

考虑一个独轮车的简易模型，如图 13.5 所示。在不考虑平衡问题的情况下，有两个行动变量。骑独轮车的人可以设置踩踏速度和车轮相对于 z 轴的方向。令 σ 表示踩踏角速度，令 r 表示车轮的半径。独轮车的速度为 $s = r\sigma$。在这种模型中，速度直接由行动变量 u_s 来设置（或者踩踏速率可以是一个行动变量 u_σ，那么产生的速度为 $s = ru_\sigma$）。令 ω 为独轮车在 xy 平面内的角速度（因此 $\omega = \dot{\theta}$）。令 ω 直接由行动变量 u_ω 来设置。位形转移方程为：

$$\begin{aligned} \dot{x} &= u_s \cos\theta \\ \dot{y} &= u_s \sin\theta \\ \dot{\theta} &= u_\omega \end{aligned} \tag{13.18}$$

这仅仅是 $L = 1$、$u_s = ru_\sigma$ 时的差动机构方程式（13.17）。因此差动机构可以模拟独轮车。这似乎有些奇怪，但却是可能的，因为这些模型没有考虑动力学问题。注意到独轮车也可以模拟简易汽车模型。因此三轮车模型和独轮车模型是相似的。

13.1.2.4 汽车牵引拖车

给简易汽车连接一个或多个拖车就产生了简易汽车的扩展。你可能已经看见在机场的停机坪上的行李拖车。对这些模型的约束进行建模有许多微妙的问题。方程的形式对拖车

连接点的精确位置以及体坐标系的选择都非常敏感。可以用第 3.3 节的方式来表达运动学，但是在分析约束时，这可能使得问题更加复杂。这更像是一种寻找约束的简单表达的艺术。这里给出的模型是从文献[730]修改而来的[1]。

考虑如图 13.6 所示的一辆简易汽车牵引 k 个拖车。每个拖车都连接到前一辆车的后轮轴的中心。一个非常重要的新参数是栓系长度（hitch length）d_i，它是从第 i 个拖车的后轮中心到下一个拖车被拴系点的距离。使用第 3.3.1 节的概念，汽车本身向 C 贡献了 $\mathbb{R}^2 \times \mathbb{S}^1$，每个拖车向 C 贡献一个 $\mathbb{S}^1$ 分量。因此 C 的维数是 $k+3$。令 θ_i 表示第 i 个拖车的相对于世界坐标系的方向。

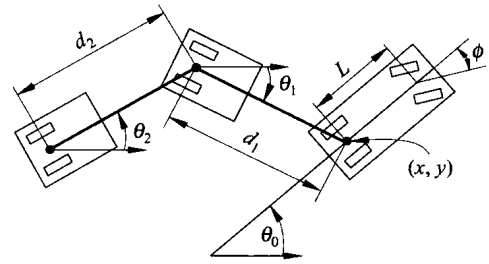


图 13.6 汽车牵引拖车的参数

位形转移方程为：

$$
\begin{aligned}
\dot{x} &= s\cos\theta_0 \\
\dot{y} &= s\sin\theta_0 \\
\dot{\theta}_0 &= \frac{s}{L}\tan\phi \\
\dot{\theta}_1 &= \frac{s}{d_1}\sin(\theta_1 - \theta_0) \\
&\vdots \\
\dot{\theta}_i &= \frac{s}{d_i}\left(\prod_{j=1}^{i-1}\cos(\theta_{j-1} - \theta_j)\right)\sin(\theta_{i-1} - \theta_i) \\
&\vdots \\
\dot{\theta}_k &= \frac{s}{d_k}\left(\prod_{j=1}^{k-1}\cos(\theta_{j-1} - \theta_j)\right)\sin(\theta_{k-1} - \theta_k)
\end{aligned}
\tag{13.19}
$$

该模型的一个变化是允许操纵拖车的车轮。对于单个拖车来说，这导致了类似于救火车的模型[165]。

13.1.3 速度约束的其他实例

目前所见到的微分模型是从沿着平坦表面滚动的车轮得到的。考虑物体相互接触的其他方式以后，可能有许多推广形式。在机器人领域会出现许多有趣的微分模型。本节简要介绍在 C 空间上速度约束的一些其他例子。现在仍然不考虑其中的动力学问题。这种模型有时被称为准静态的（quasi-static），因为即使发生运动，模型的某些方面也将物体当作静态的来对待。当运动的速度足够低时，这种模型通常是合理的。

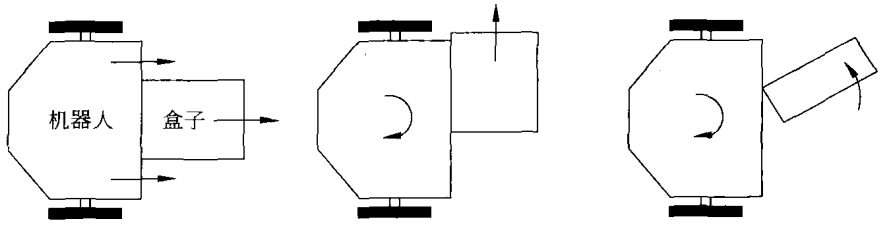
13.1.3.1 推动一个盒子

假设使用差动机构机器人推地面上的一个盒子，如图 13.7(a)所示。假设盒子是一个凸

1 原始模型要求一个连续的转向角。

多边形，它的一条边与机器人的前部相接触。在盒子和地面之间、盒子和机器人之间都存在摩擦。假设机器人运动速度足够低使得动态很小。假设盒子是静止的，除非机器人运动时推动它。禁止进行诸如“踢”盒子这种操作。术语“稳定地推动”[12,674,684]指的是机器人移动盒子，就好像盒子刚性地连接在机器人上一样。

当机器人推盒子时，盒子可能滑动或者滚动，如图 13.7(b)和图 13.7(c)所示。这些情况被认为是不合规定的，因为它们没有建立稳定的推动。那么机器人应该进行什么样的运动？从差动机构机器人的位形转移方程开始，然后确定需要对 U 进行什么约束才能保持稳定的推动。假设使用式（13.17）。只可能向前运动，因为如果机器人向相反方向运动，那么将与盒子脱离接触。因此 s 必须为正（并且为了匹配准静态模型，s 应该足够小以使动态效应可以被忽略）。应该如何对旋转速率 ψ 进行约束？对 ψ 的约束依赖于摩擦模型（例如库仑模型）、盒子的形状、推盒子的哪条边等。文献[674, 684]给出了关于这些约束的详细论述。这产生了一个区间 $[a,b]\subseteq[-\pi/2,\pi/2]$，其中 $a<0$ 和 $b>0$，并且要求 $\psi\in[a,b]$。约束的组合就产生了类似于 Dubins 车的运动模型。主要不同之处在于最大左转向角与最大右转向角可能不同。



(a) 稳定地推盒子　(b) 盒子相对于机器人发生滑动　(c) 盒子相对于机器人发生旋转

图 13.7　Lynch 和 Mason 指出，推一个盒子很像驱动简易汽车：（a）机器人小心翼翼地运动时，盒子的行动就像贴在机器人上一样；（b）但是如果机器人转弯太急，那么盒子将会侧滑，这就产生了对转向角的限制；（c）机器人急速转弯也可能使盒子发生旋转[674]

为了将这种模型应用于规划中，C 空间应该是 $\mathbb{R}^2\times\mathbb{S}^1\times\mathbb{R}^2\times\mathbb{S}^1$，因为有两个刚体。可以用第 7.3.2 节的操作规划框架来得到一个混合系统，也可得到表示机器人接触或者不接触盒子的各种操作图。例如机器人能够沿着几个可能的边来推动盒子。如果机器人运动受阻，那么它可以改变推的边以在新的方向上移动盒子。

13.1.3.2　飞机

将第 13.1.2 节的 Dubins 车的模型扩展到三维世界可以产生一个对空中交通分析来说可能是合理的飞机简易飞行模型。首先假设飞机保持固定高度，并且只能进行偏航旋转。在这种情况下，施加约束 $s=1$（或者某适宜的正向速度）就可以直接使用式（13.15）。这等价于 Dubins 车，除了禁止 $s=0$ 以外，因为 $s=0$ 意味着飞机瞬间停在空中。这种模型假设飞机相对于 C 空间来说非常小。更精确的模型应该考虑俯仰和滚动旋转、扰动以及动态的影响。在研究飞机飞行稳定性时，这些都非常重要。但是这里忽略这些问题。

除了进行类似于 Dubins 车的运动以外，现在考虑飞机还能够改变其高度。在这种情况下，令 $C=\mathbb{R}^3\times\mathbb{S}^1$，其中额外的一个 $\mathbb{R}$ 表示相对于一个光滑表面的飞行高度。位形表示为 $q=(x,y,z,\theta)$。令 u_z 表示直接引起高度变化的一个行动：$\dot{z}=u_z$。行动 u_ϕ 与 Dubins 模型中

的相同。位形转移方程为：

$$\begin{aligned} \dot{x} &= \cos\theta \qquad & \dot{z} &= u_z \\ \dot{y} &= \sin\theta \qquad & \dot{\theta} &= u_\omega \end{aligned} \tag{13.20}$$

对于满足 $u_z \neq 0$ 和 $u_w \neq 0$ 的一个固定值 $u = (u_z, u_w)$，它会产生螺旋状的路径。螺旋的中心轴平行于 z 轴，路径在 xy 平面上的投影是一个圆或者圆弧。应该基于最大可能高度变化和飞机偏航速率变化来设置 u_z 和 u_w 最大绝对值。

13.1.3.3 滚动小球

考虑在平面上滚动一个小球，而不是车轮。将球放置在桌面上，将你的掌心向下放在小球上并滚动它。这样应该有两个自由度：向前滚动和侧向滚动。小球应不能原地旋转。方向可以认为是两个行动变量。小球的总自由度为 5，因为它能够实现 $SO(3)$ 中的任意方向和平面中的任意位置 (x, y)，因此 $C = \mathbb{R}^2 \times SO(3)$。只给定两个行动变量，那么可能将小球滚动到任意位形吗？文献[635,494]认为这是可能的，甚至对于一个球体在另外一个球体上滚动这种更加一般的问题也是如此（可以认为平面是半径为无穷大的球面）。在机器人实际操作中，当接触球形物体时就会出现这种情况（例如机械手的手指可能有球形指头），参见文献[105,679,728,732]。

所产生的转移方程如文献[719]所示（也可参见文献[728]）

$$\begin{aligned} \dot{\theta} &= -u_2 \\ \dot{\phi} &= \frac{u_1}{\cos\theta} \\ \dot{x} &= -u_1\rho\sin\psi - u_2\rho\cos\psi \\ \dot{y} &= -u_1\rho\cos\psi + u_2\rho\sin\psi \\ \dot{\psi} &= -u_1\tan\theta \end{aligned} \tag{13.21}$$

在这些方程中，x 和 y 是接触点在平面中的位置，θ 和 ϕ 是在小球坐标系中接触点的位置，并用球面坐标进行表达。小球的半径为 ρ。最后 ψ 表示小球关于接触点的方向。

13.1.3.4 将位形限制在一个曲面上

约束可能会使位形被限制在一个更低维的曲面上。令 $C = \mathbb{R}^2$，并且考虑系统：

$$\dot{x} = yu \qquad \dot{y} = -xu \tag{13.22}$$

其中 $(x, y) \in \mathbb{R}^2$ 和 $u \in U = \mathbb{R}$。那么常数行动 $u \neq 0$ 的积分曲线是什么？从任意点 $(x, y) \in \mathbb{R}^2$，轨迹是以原点为圆心、以 $\sqrt{x^2 + y^2}$ 为半径的圆。沿着圆运动的速度由 $|u|$ 确定，方向由 u 的符号确定。因此式（13.22）表明位形被限制在一个圆上。除此之外，没有其他约束。

假设初始位形给定为 (x_0, y_0)。因为位形被限制在一个圆上，所以 C 空间也可以定义为 $C = \mathbb{S}^1$。$\mathbb{S}^1$ 上的每个点都可以映射到半径为 $r = \sqrt{x_0^2 + y_0^2}$、圆心在(0, 0)的圆上。在这种情况下，对速度没有微分约束，只要将运动限制在该圆上。圆上点的一维切空间中允许任何速度。该模型等价于式（13.22）。

现在考虑限制在一个圆上的可能轨迹：

$$h(x, y) = x^2 + y^2 - r^2 = 0 \tag{13.23}$$

这意味着对于所有时间 t：

$$h(x(t),y(t)) = x(t)^2 + y(t)^2 - r^2 = 0 \tag{13.24}$$

为了产生速度约束，上式对时间求导产生：

$$\frac{\mathrm{d}h(x,y)}{\mathrm{d}t} = 2x\dot{x} + 2y\dot{y} = 0 \tag{13.25}$$

这是式(13.5)给出的 Pfaffian 约束的一个例子。这种微分约束的参数形式恰好是式(13.22)。为 $(y,-x)$ 整倍数的任意速度向量都满足式（13.25）。当表示成一个微分约束时，半径 r 就无关紧要了。这是因为它可以从初始位形进行确定。

这里刚才发生的是完全可积微分模型的一种特殊情况。一般情况下，如果模型 $\dot{q} = f(q,u)$ 可以表示成 $h(q) = 0$ 这种形式的约束的导数，那么位形转移方程被认为是完全可积的。从 $h_i(q) = 0$ 形式的约束很容易得到一个隐含的微分模型。对每个约束进行微分就可以得到：

$$\frac{\mathrm{d}h_i(q)}{\mathrm{d}t} = 0 \tag{13.26}$$

例如在第 4.4 节中，在闭运动链中会出现这种约束，隐含的微分模型只表示这些约束的条件是速度必须处在切空间。但是很难得到参数形式的微分模型。通过第 7.4.1 节的线性代数可以计算任意 q 上可能的速度向量。

很难确定微分模型是不是完全可积的，完全可积意味着位形受限于一个更低维的曲面上。对式（13.41）所描述的某些系统来说，这可以用第 15.4.2 节的 Frobenius 定理来解决。如果这种系统不是完全可积的，那么它们称为非完整的；反之，它们是完整的。一般情况下，即使模型在理论上是可积的，然而实际进行积分却是另外一个问题。在多数情况下，对模型进行积分是困难的或者是不可能的。

因此有时需要直接用微分形式的约束，即使它们是可积的，并且，微分约束下的规划方法可以应用于有 $h(q) = 0$ 形式约束的问题中。例如闭运动链的运动规划可以用处理微分约束的规划算法来进行。

13.2 动力系统的相空间表示

第 13.1 节定义的微分约束通常被称为运动学的，因为它们是用 C 空间上的速度来表达的。这种表述对许多问题是非常有价值的，例如对轮式机器人运动的可能方向进行建模。但是它没有考虑动力学。例如，假设简易汽车行驶速度很快。如果考虑动力学，那么汽车就不能瞬间启动和停止。例如，如果它全速向墙驶去，那么任何合理的模型都不应该允许它仅在 1mm 之外进行刹车来避免碰撞。由于存在动量，所需要的停车距离取决于速度。你可能已经从驾驶员训练课程中学到过这些知识。

考虑动量和其他方面的动力学，那么需要高阶微分方程。通常对加速度 $\ddot{q}$ 有约束，$\ddot{q}$ 被定义为 $\mathrm{d}\dot{q}/\mathrm{d}t$。例如汽车可能只能以不导致车轮打滑（或者翻车）的最大速率进行减速。通常行动用高阶导数来表达。例如把汽车油门踩到底就能够直接设置加速度。将踩踏油门的量作为一个行动变量是合理的。在这种情况下，两次积分得到位形。第一次产生速度，

第二次产生位形。

因此除了速度 $\dot{q}$ 和位形 q 以外，动力学模型还涉及加速度 $\ddot{q}$。同样地，也存在隐含模型和参数模型两种情况。对于隐含模型，约束表示为：

$$g_i(\ddot{q},\dot{q},q)=0 \tag{13.27}$$

对于参数模型，它们表示为：

$$\ddot{q}=f(\dot{q},q,u) \tag{13.28}$$

13.2.1 通过增加维数来减小自由度

高阶导数约束非常复杂。本节介绍一种方法将含有高阶导数的约束转化为一阶导数的约束的集合。这涉及相空间（phase space）的概念，它比原始的 C 空间的维数更多。因此存在权衡问题，但是公认的是增加空间的维数比处理高阶导数更加容易。一般情况下，状态空间是指 C 空间或者 C 空间所产生的相空间。

13.2.1.1 标量情况

为了使得所讨论的问题更加具体，这里考虑如下微分方程：

$$\ddot{y}-3\dot{y}+y=0 \tag{13.29}$$

其中变量 $y\in\mathbb{R}$ 是标量。这是一个二阶微分方程。相空间的定义如下。令 $x=(x_1,x_2)$ 表示二维相向量，它们的定义为 $x_1=y$ 和 $x_2=\dot{y}$。在定义相空间的地方，状态空间和状态向量可以与相空间和相向量互换。将它们带入式（13.29）就产生了：

$$\ddot{y}-3x_2+x_1=0 \tag{13.30}$$

用 $\dot{x}_2=\ddot{y}$，那么微分方程变成：

$$\dot{x}_2-3x_2+x_1=0 \tag{13.31}$$

该表达式与式（13.29）等价吗？不完全是。需要多一个约束 $x_2=\dot{x}_1$。在隐含形式下，$\dot{x}_1-x_2=0$。相空间方法的关键在于用导数将某些相变量联系起来。

使用相空间，我们将二阶微分方程（13.29）转化为两个一阶微分方程：

$$\begin{aligned}\dot{x}_1&=x_2\\ \dot{x}_2&=3x_2-x_1\end{aligned} \tag{13.32}$$

那么式（13.32）可以表示为 $\dot{x}=f(x)$，其中 f 是将 $\mathbb{R}^2$ 映射到 $\mathbb{R}^2$ 的函数。

对隐含形式的微分方程 $g(\ddot{y},\dot{y},y)=0$ 也可以使用相同的方法。令 $x_1=y$、$x_2=\dot{y}$ 和 $\dot{x}_2=\ddot{y}$。这就产生了隐含方程 $g(\dot{x}_2,x_2,x_1)=0$ 和 $\dot{x}_1=x_2$。现假设用微分方程表示的一个标量行动 $u\in U=\mathbb{R}$。则在隐含形式中，$g(\ddot{y},\dot{y},y,u)=0$ 可以表示为 $g(\dot{x}_2,x_2,x_1,u)=0$。

假设给定的加速度约束用参数形式表示为 $\ddot{y}=h(\dot{y},y,u)$。它可以转化为 $\dot{x}=f(x,u)$ 形式的相空间转移方程或者状态转移方程，其中 $f:\mathbb{R}^2\times\mathbb{R}\to\mathbb{R}^2$。表示为：

$$\begin{aligned}\dot{x}_1&=x_2\\ \dot{x}_2&=h(x_2,x_1,u)\end{aligned} \tag{13.33}$$

对于二阶微分方程，通常给定两个初始条件。需要 $y(0)$ 和 $\dot{y}(0)$ 的值来确定精确位置 $y(t)$，$t\geqslant 0$。使用相空间表示，不需要高阶初始条件，因为相空间中的点表示 y 和 $\dot{y}$。因此对所有 $t\geqslant 0$，给定相空间中的一个初始点和 $u(t)$，就可以确定 $y(t)$。

例 13.3（二重积分器） 二重积分器是一个简单而又重要的例子，它能够非常好地阐述相空间。假设给定二阶微分方程为 $\ddot{q}=u$，其中 q 和 u 从 $\mathbb{R}$ 中进行选择。这意味着行动就是加速度。积分[1]一次产生速度 $\dot{q}$，积分两次产生位置 q。如果给定 $q(0)$ 和 $\dot{q}(0)$，并指定 $u(t')$，$t'\in[0,t)$，那么就能够确定 $q(t)$ 和 $\dot{q}(t)$。

定义一个二维相空间 $X=\mathbb{R}^2$，其中：

$$x=(x_1,x_2)=(q,\dot{q}) \tag{13.34}$$

状态（或者相）转移方程 $\dot{x}=f(x,u)$ 为：

$$\begin{aligned}\dot{x}_1&=x_2\\ \dot{x}_2&=u\end{aligned} \tag{13.35}$$

为了确定状态轨迹，除了给定行动历史以外，还必须给定初始值 $x_1(0)=q_0$（位置）和 $x_2(0)=\dot{q}_0$（速度）。如果 u 是常数，那么状态轨迹是二次式，因为它是由常数函数二次积分而来。 ■

13.2.1.2 向量情况

向相空间的变换也可以扩展到微分方程，其中微分方程有多个变量对时间进行求导。假设 q 表示一个位形，它用 n 维平滑流形 C 上的坐标邻域来表达。$g(\ddot{q},\dot{q},q)=0$ 或者 $g(\ddot{q},\dot{q},q,u)=0$ 形式的二阶约束可以表示为 $2n$ 维状态空间中的一阶约束。令 x 表示 $2n$ 维相向量。将标量情况下的方法进行扩展，x 定义为 $x=(\dot{q},q)$。对于满足 $1\leqslant i\leqslant n$ 的每个整数 i，有 $x_i=q_i$。对于满足 $n+1\leqslant i\leqslant 2n$ 的每个 i，有 $x_i=\dot{q}_i$。对隐含约束直接进行这种替换将阶数减小到 1。

假设 n 个微分方程的集合用参数的形式表示为 $\ddot{q}=h(q,\dot{q},u)$。在相空间中，有 $2n$ 个微分方程。前 n 个对应于相空间定义 $\dot{x}_i=x_{n+i}$，$1\leqslant i\leqslant n$。这些都是成立的，因为 $x_{n+i}=\dot{q}$ 并且 $\dot{x}_i$ 是 $\dot{q}_i$ 的导数，$i\leqslant n$。$\dot{x}=f(x,u)$ 的其余 n 个分量也采用 h 的形式，只是在 $h(q,\dot{q},u)$ 中 q 的位置上用 x 的前 n 个分量代替、$\dot{q}$ 的位置上用 x 的其余 n 个分量代替。那么结果可以用 $h(x,u)$ 表示（直接从 $h(q,\dot{q},u)$ 得到）。这样所产生的后 n 个方程为 $\dot{x}_i=h_i(x,u)$，$n+1\leqslant i\leqslant 2n$。这 $2n$ 个方程定义了一个 $\dot{x}=f(x,u)$ 形式的相空间（或者状态）转移方程。现在对加速度的约束可以表示成相空间中的速度约束。可以用第 8.3 节的切空间概念来表示涉及加速度的约束，并且状态空间 X 是 C 的切丛（在式（8.9）为 $\mathbb{R}^n$ 和式（15.67）为平滑流形所定义的），因为 q 和 $\dot{q}$ 一起表示了一个切空间 $T_q(C)$ 和一个特定的切向量 $\dot{q}\in T_q(C)$。

13.2.1.3 高阶微分约束

相空间的思路可以应用于高于二阶的微分方程。例如约束可能涉及加速度 $q^{(3)}$ 的导数，这通常被称为加加速度（jerk）。如果微分方程涉及加加速度变量，那么定义 $3n$ 维相空间就可以得到一阶约束。在这种情况下，在诸如 $g(q^{(3)},\ddot{q},\dot{q},q,u)=0$ 的约束中的每个 q_i、$\dot{q}_i$ 和 $\ddot{q}_i$ 都被定义为一个相变量。相似地，通过引入 k n 维相空间，k 阶微分约束可以降为一阶约束。

例 13.4（积分器链） 高阶微分约束的一个简单的例子就是积分器链[2]。这是对例 13.3

1 当进行积分时，假设被积函数是可积的。

2 之所以这样称呼它是因为在系统的结构图中，它被描述成积分器的一个链。

的高阶推广。假设给定 k 阶微分方程 $q^{(k)}=u$，其中 q 和 u 是标量，$q^{(k)}$ 表示 q 对时间的 k 阶导数。

定义一个 k 维相空间 X，其中：

$$x=(q,\dot{q},\ddot{q},q^{(3)},\cdots,q^{(k-1)}) \tag{13.36}$$

状态（或者相空间）转移方程 $\dot{x}=f(x,u)$ 为 $\dot{x}_i=x_{i+1}$，$1\leqslant i\leqslant n-1$，且 $\dot{x}_n=u$。这 n 个方程等价于 $q^{(k)}=u$。

初始状态指定初始位置和所有 $k-1$ 阶导数。使用这些初始条件以及行动 u，由积分器链就可以得到状态轨迹。■

你可能想知道用足够高维数的相空间是否可以完全消除导数。这肯定是可以的。例如，如果有二阶约束，那么可以引入一个三维相空间，其中 $x=(q,\dot{q},\ddot{q})$。这使得诸如 $g(q,\dot{q},\ddot{q})=0$ 这样的约束变成 $g(x)=0$。使用约束的这种表示存在一个问题，那就是状态必须沿着约束面，其运动方式类似于第 4.4 节沿着闭运动链的解集所进行的运动。这就是为什么在那里出现切空间的原因。在这两种情况下，可允许速度的集合在空间中每个点上都有约束。

用相空间所定义的问题都有一个有趣的特性，那就是漂移（drift）。这就是说对于某些 $x\in X$，不存在 $u\in U$ 使得 $f(x,u)=0$。对于第 13.1.2 节中的例子，这种行动总是存在的。这是无漂移（driftless）系统的例子。这是可能的，因为约束没有涉及动力学。在动力系统中，由于存在动量，所以不可能瞬间停止下来，这就是一种形式的漂移。例如如果汽车在 3 米以外向着墙以 100km/ h 的速度行驶，那么汽车将会“漂移”进墙中。任何行动都无法使汽车瞬间停下来。一般情况下，在相空间中也无法实现瞬间停止。

13.2.2 线性系统

既然已经将相空间定义为能够处理动力学的一种特殊形式的状态空间，那么很容易对基于数学形式来定义的微分模型进行分类。线性系统已经得到广泛的研究，特别是在控制论中。原因是可以应用线性代数来产生控制律[194]。该思路可以推广到涉及最优性指标[28,573]、大自然[97,567]，或者多局中人[60]这样的线性系统。

令 $X=\mathbb{R}^n$ 表示一个相空间，令 $U=\mathbb{R}^m$ 表示一个行动空间，$m\leqslant n$。线性系统是一个微分模型，对于它来说状态转移方程可以表示为：

$$\dot{x}=f(x,u)=Ax+Bu \tag{13.37}$$

其中 A 和 B 分别是 $n\times n$ 和 $n\times m$ 维的定常实数矩阵。

例 13.5（线性系统） 对于式（13.37）的一个简单例子，令 $X=\mathbb{R}^3$ 和 $U=\mathbb{R}^2$，令：

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{pmatrix}=\begin{pmatrix}0 & \sqrt{2} & 1\\ 1 & -1 & 4\\ 2 & 0 & 1\end{pmatrix}\begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix}+\begin{pmatrix}1 & 0\\ 0 & 1\\ 1 & 1\end{pmatrix}\begin{pmatrix}u_1\\ u_2\end{pmatrix} \tag{13.38}$$

矩阵乘法表明所有三个方程对状态和行动变量来说是线性的。可以将它与例 11.25 所示的离散时间线性高斯系统进行比较。■

回顾第 13.1.1 节中 k 个线性约束将速度限制在一个 $n-k$ 维超平面上。式（13.37）中的线性模型是参数形式，这意味着每个行动变量可能有一个独立的自由度。在这种情况下，

$m=n-k$。在$m=0$这种极端情况下，没有行动，这样导致了$\dot{x}=Ax$。相速$\dot{x}$对每个点$x\in X$来说都是固定的。如果$m=1$，那么在每个$x\in X$上用u就可以选择一维速度集合。这意味着速度方向是固定的，但是速度的大小由u确定。一般情况下，在一个点$x\in\mathbb{R}^n$上可允许的速度的集合是切空间$T_x(\mathbb{R}^n)$的一个m维线性子空间（如果B是非奇异的）。

尽管有式（13.37），仍然有可能从任意初始状态到达状态空间的所有状态。但是由于被限制在切空间上，所以到达一个邻近点的代价可能很高，而某些方向上的速度是不可能的。对于非线性系统，有时可能很快到达一个状态的一个小的邻域中的任意点，同时始终保持在该状态周围一个小的区域内。这就是可控性问题，这将在第15.1.3节和第15.4.3节介绍。

尽管这里没有介绍系统的可观性，但它也是控制理论中的一个重要内容[194,481]。根据第11章的I-空间的概念，这意味着定义了一个$y=h(x)$形式的传感器，任务是在给定历史I-状态的情况下确定目前的状态。如果系统是可观的，那么意味着非确定的I-状态是一个单点。否则系统可能只是部分可观的。在线性系统情况下，如果感测模型也是线性的：

$$y=h(x)=Cy \tag{13.39}$$

那么可以用简单的矩阵条件就可以确定系统是否可观[194]。也存在非线性可观性理论[481]。

就像离散规划问题一样，也可定义与时间有关的微分模型。在离散情况下，这涉及对阶段的依赖性。对于连续情况，时变线性系统被定义为：

$$\dot{x}=f(x(t),u(t),t)=A(t)x(t)+B(t)u(t) \tag{13.40}$$

在这种情况下，矩阵元素是时间的函数。许多控制技术都可以应用于此，但是这里不予考虑，因为大多数规划问题都是时不变的（定常的）。

13.2.3 非线性系统

尽管能够为线性系统研究许多控制律，但是物理世界中的绝大多数系统都不是线性的。不满足式（13.37）或者式（13.40）的任何微分模型都被称为非线性系统。对于$X=C$这种特殊情况，第13.1.2节给出的所有模型都是非线性系统。

有一族重要的非线性系统在某种意义上是线性的。令X为一个平滑的n维流形，$U\subseteq\mathbb{R}^m$。对于某$m\leqslant n$，令$U=\mathbb{R}^m$。使用坐标邻域，对于平滑函数f和g_i，如下形式的非线性系统：

$$\dot{x}=f(x)+\sum_{i=1}^{m}g_i(x)u_i \tag{13.41}$$

被称为仿射控制系统[1]（control-affine system 或 affine-in-control system）。在非线性理论中这种系统得到了人们的广泛研究[481,848]。它们关于行动是线性的，但是对于状态来说是非线性的。关于仿射控制系统的更多内容参见第15.4.1节。

对于一个仿射控制系统，它不一定能够得到零速度，因为f引起漂移。如果$f\equiv 0$，那

1 不要将 control-affine systems 与 affine control systems 混淆了，affine control systems 的形式是$\dot{x}=Ax+Bu+w$，其中A和B是某常数矩阵，w是一个常数向量。

么会出现无漂移仿射控制系统这种非常重要的特殊情况。它记为：

$$\dot{x} = \sum_{i=1}^{m} g_i(x)u_i \tag{13.42}$$

对于 $1 \sim m$ 的每个 i，置 $u_i = 0$ 就可以得到零速度 $\dot{x} = 0$。

例 13.6（非完整积分器） 无漂移仿射控制系统的最简单的一个例子是非完整积分器，它是 Brockett 在文献[144]中介绍的。因为它出现在量子力学中[114]，所以有时它也被称为 Brochett 系统或者 Heisenberg 系统。令 $X = \mathbb{R}^3$，行动集合 $U = \mathbb{R}^2$。非完整积分器的状态转移方程为：

$$\begin{aligned} \dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= x_1 u_2 - x_2 u_1 \end{aligned} \tag{13.43}$$

■

一些非线性系统可以用第 13.1.1 节的 Pfaffian 约束来隐含表示，并且可以从 C 空间推广到相空间。根据 X，一个 Pfaffian 约束可表示为：

$$g_1(x)\dot{x}_1 + g_2(x)\dot{x}_2 + \cdots + g_n(x)\dot{x}_n = 0 \tag{13.44}$$

即使方程对 $\dot{x}$ 是线性的，也允许它对 x 是非线性的。

完整模型和非完整模型对于相空间来说都可能存在，就像第 13.1.3 节的 C 空间中的情况一样。可以用第 15.4.2 节介绍的 Frobenius 定理来确定仿射控制系统是否完全可积。

13.2.4 通过增加积分器来对模型进行扩展

第 13.1 节的微分模型在许多应用中似乎是实际的，因为它要求行动瞬间发生变化。例如在简易汽车中，转向角和速度可能瞬间变化到任意值。这意味着汽车的加速度能够瞬间变化。如果汽车运动速度很慢，那么这可能是一种合理的近似。但是在高速情况下，这种模型就非常荒谬。

假设给定 $\dot{x} = f(x,u)$ 形式的状态转移方程，其中 X 的维数为 n。模型可以得到强化：

（1）选择一个行动变量 u_i。

（2）将行动变量重新命名为一个新的状态变量 $x_{n+1} = u_i$。

（3）定义一个新的变量 u_i' 来代替 u_i。

（4）通过引入 $\dot{x}_{n+1} = u_i'$，将状态转移方程增加一维。

这种增强是指在 u_i 的前面放置一个积分器。重复进行该过程就产生了一个积分器链。它也可以被应用到不同行动变量的情况。

13.2.4.1 更好的独轮车模型

在行动变量前面放置积分器可以改进第 13.1 节中的模型。例如考虑独轮车模型式(13.18)。不用 u_s 来直接设置速度，假设速度是通过行动 u_a 的积分而来的，行动表示加速度。方程 $\dot{s} = u_a$ 用来代替 $s = u_s$，这意味着行动可设定速度的变化。如果从某有界区间选择 u_a，那么速度是时间的连续函数。

在这种情况下应该如何表示转移方程？u_a 的可能值的集合对 x 和 y 施加了一个二阶约

束，因为需要二重积分来确定它们的值。应用相空间的思路，可以认为 s 是一个相变量。这就导致了一个四维相空间，其中状态是 (x,y,θ,s)。状态（或者相空间）转移方程为：

$$\begin{aligned} \dot{x} &= s\cos\theta & \dot{\theta} &= u_w \\ \dot{y} &= s\sin\theta & \dot{s} &= u_a \end{aligned} \tag{13.45}$$

将它与式（13.18）进行比较。u_s 被 s 代替了，因为现在速度是一个相变量，并且增加一个方程来反映速度和加速度之间的联系。

用相同的思路也可以使独轮车的方向成为时间的连续函数。令 u_α 表示一个角加速度行动。令 ω 表示角速度，它是一个新的状态变量。这产生了一个五维相空间和一个被称为二阶独轮车的模型：

$$\begin{aligned} \dot{x} &= s\cos\theta \\ \dot{y} &= s\sin\theta \\ \dot{\theta} &= \omega \\ \dot{s} &= u_a \\ \dot{\omega} &= u_\alpha \end{aligned} \tag{13.46}$$

其中 $u=(u_a,u_\alpha)$ 是一个二维行动向量。在某些情形下，s 可能固定为一个常值，这就是说 u_a 固定为 $u_a=0$。

13.2.4.2 连续转向汽车

作为另外一个例子，现在考虑简易汽车。就像式（13.15）所表述的那样，允许转向角不连续变化。为简单起见，假设速度固定为 $s=1$。为了使转向角随时间连续变化，令 u_ω 为一个行动，它表示转向角的速度：$\dot{\phi}=u_\omega$。那么产生一个四维状态空间，其中状态被表示为 (x,y,θ,ϕ)。这产生了一个连续转向汽车的模型：

$$\begin{aligned} \dot{x} &= s\cos\theta \\ \dot{y} &= s\sin\theta \\ \dot{\theta} &= \frac{\tan\phi}{L} \\ \dot{\phi} &= u_\omega \end{aligned} \tag{13.47}$$

其中有两个行动变量 u_s 和 u_ω。文献[851]应用这种模型进行规划。

第二个积分器使得转向角成为时间的 C^1 平滑函数。令 ω 为一个状态变量，令 u_α 表示转向角的角加速度。在这种情况下，状态向量为 (x,y,θ,ϕ,ω)，状态转移方程为：

$$\begin{aligned} \dot{x} &= s\cos\theta \\ \dot{y} &= s\sin\theta \\ \dot{\theta} &= \frac{\tan\phi}{L} \\ \dot{\phi} &= \omega \\ \dot{\omega} &= u_\alpha \end{aligned} \tag{13.48}$$

多次应用积分器可以使任意变量足够平滑，并且由于存在对相变量和行动集合的限制，所以变化率是有界的。

13.2.4.3 平滑差动机构

不直接设置发动机的速度，而是通过定义加速发动机的行动 u_l 和 u_r，就可以产生一个二阶差动机构模型。令 ω_l 和 ω_r 分别表示左右发动机的角速度。所产生的状态转移方程为：

$$\begin{aligned}\dot{x} &= \frac{r}{2}(\omega_l + \omega_r)\cos\theta \\ \dot{y} &= \frac{r}{2}(\omega_l + \omega_r)\sin\theta \\ \dot{\theta} &= \frac{r}{L}(\omega_r - \omega_l) \\ \dot{\omega}_l &= u_l \\ \dot{\omega}_r &= u_r\end{aligned} \tag{13.49}$$

总之，使得上述这些模型更加实用的一种方法是在任意行动变量之前插入一个或者多个积分器。每引入一个积分器，相空间的维数就会相应增加。单个积分器迫使原始行动成为时间连续的。如果新的行动是有界的，那么原始行动的变化率在可微分（一般是 Lipschitz 的，如式（8.16））的地方是有界的。用一个二重积分器，使原始行动成为 C^1 平滑的。在一个行动变量上链接更多的积分器会进一步约束变量的值。一般情况下，在一个原始行动前面链接 k 个积分器就可以迫使它成为 C^{k-1} 平滑的且不违反 Lipschitz 的界。

该方法的局限性在于为了构造更加实用的模型，其他变量可能依赖于新的相变量。例如，如果简易汽车快速行驶，那么我们将不能够像慢速运动的汽车那样进行急转弯。好的微分模型应该仔细考虑力学问题。这就是第 13.3 节和第 13.4 节将介绍的内容。

13.3 基本牛顿-欧拉力学

力学是一个纷繁复杂的学科。通过两三节的内容对它进行全面的介绍是不可能的。这里的目的是简要介绍一些主要概念以及在第 14 章规划算法中可能用到的一些模型。希望本节和第 13.4 节的内容能够激发读者对力学领域更深入的研究学习。另一方面，如果只对使用微分模型感兴趣，那么可以跳过推导过程。只需要记住本节产生的所有微分模型都以 $\dot{x} = f(x,u)$ 的形式出现，它们将被用在规划算法中。

对于力学问题，有两点需要记住：

（1）模型应与对物理世界中物体行为的试验观测保持一致。这些观测依赖于试验的类型。在特殊应用中，一些效果可能不够显著，或者不被试验所检测。例如用测量汽车速度的雷达测速仪器很难检测到相对论效应。因此对世界以及试验的类型进行简化假设是非常重要的。

（2）方法通常是表达某些定律，这些定律转换为相空间中对允许速度的约束。这意味着在力学中通常得到的是隐含表示，它们必须被转换为参数形式，并且多数力学处理中并不明确提及行动变量，它们目的是控制物理世界。从力学的观点上看，可以假设行动已经结束。因此约束变成 $g(\dot{x},x)=0$，而不是 $g(\dot{x},x,u)=0$。

几种力学表述可以导致相同的微分约束，但是来自于不同的数学推理。本章的其余部分将简要介绍三种学派，它们每种都比前一种更加完善和先进。最容易理解的是牛顿-欧拉

力学，它遵循著名的牛顿定律。第 13.4.1 节介绍拉格朗日力学，它采用函数空间上的一般优化原理来产生微分约束（例如变分法）。第 13.4.4 节介绍的哈密顿力学定义了更高维数的状态空间，在该空间上仍然通过优化得到微分约束。

13.3.1 牛顿模型

力学最基本的表述可以追溯到牛顿和欧拉时代，其中一部分已经在物理课中学习过。考虑第 3.1 节定义的世界 $\mathcal{W}$，除了允许二维和三维世界以外，还有一维世界 $\mathcal{W}=\mathbb{R}$。仍然需要时间的概念。在空时中得到的运动空间可以被表述为一个 Galilean 群[39]，但是这里的表示利用了时间和欧拉空间的标准直观概念。还假设不存在由于时空弯曲所产生的任何相对论效应（牛顿和欧拉不知道这些，相对论对于地球上或地球附近的多数小尺度机械系统来说并不显著）。

惯性坐标系 牛顿-欧拉力学的核心是用惯性坐标系来表示世界 $\mathcal{W}$ 中的点。假设将世界 $\mathcal{W}$ 的原点和轴选定在宇宙的某个位置。需要将它们以某种方式固定下来，这种方式不能影响我们对基本运动定律的认识。假设我们在室内正在打壁球，当球从一面墙弹到另外一面墙时，我们想对球的运动进行建模。如果坐标系刚性地固连在球上，那么球将不再运动；但是墙、地球以及宇宙的其余部分将绕球疯狂旋转（假设我们有一架摄影机，它指向球的体坐标系的某个轴——如果你观看用这架摄影机拍摄的电影，那么你很快就会感觉很不舒服）。如果坐标系关于场地固定，那么可以对球的位置进行测量。出于现实的考虑，我们可以认为这种固定的坐标系是惯性的。但需要注意的是如果坐标系关于太阳固定，那么球将疯狂弹跳。地球的自转和公转使得球以不可想象的速度进行运动。在现实世界中不存在惯性坐标系；但是假设惯性坐标系固连在地球上，对于基于地球的机械系统来说这是合理的。

惯性坐标系具有如下特性：（1）在任何惯性坐标系中运动定律相同；（2）关于惯性坐标系进行恒速无旋转运动的任何坐标系本身也是惯性的。作为第二个条件的例子，假设壁球试验是在高速公路上行驶的大卡车上进行的。忽略振动，如果在直路上卡车的速度恒定，那么惯性坐标系可以固定在卡车上，球的弹跳好像球场没有运动一样。但是如果公路转弯或者卡车改变速度，那么球的弹跳就会发生变化。如果我们仍然相信固连在卡车上的坐标系是惯性的，那么运动规律将会非常奇怪。在这种情况下，惯性坐标系必须固连到地球上来对卡车和球的行为一起进行建模。

封闭系统 牛顿-欧拉模型的另外一个重要方面是物体系统是封闭的，这意味着除显式建模的物体以外，其他物体对运动没有影响（假设我们不考虑离壁球场几百米远的一个黑洞的效应）。

牛顿定律 物体的运动都基于牛顿的三大定律，这些定律由牛顿进行实验证明，它们在惯性坐标系下成立。

（1）一个静止的物体保持静止，一个运动的物体保持恒定速度运动，除非受到非零合力[1]作用。

（2）一个物体的质量 m、加速度 a 和外力 f 之间的关系是 $f=ma$。

1 这是作用在点上的所有外力的和。

（3）两个物体之间作用力和反作用力的大小相等、方向相反。

基于这些定律，可以对运动物体系统的微分约束进行建模。

13.3.2 质点运动

用质点来描述牛顿-欧拉模型。每个质点可以被认为是一个质量为 m 的点。外力可以作用在任意质点上。通过将物体建模为粘在一起的质点集合，这样就可以确定第 13.3.3 节介绍的刚体的运动。因此首先理解质点的行为是有益的。

13.3.2.1 单质点运动

考虑在 $\mathcal{W}=\mathbb{R}$ 中运动的质量为 m 的单个质点的情况。外力变成一个标量 $f\in\mathbb{R}$。令 $q(t)$ 表示质点在时刻 t 的位置。那么加速度为 $\ddot{q}$，牛顿第二定律变成 $f=m\ddot{q}$。求解 $\ddot{q}$ 产生：

$$\ddot{q}=f/m \tag{13.50}$$

如果将 f 作为一个行动变量 u，并且 $m=1$，那么式（13.50）就是例 13.3 的二重积分器 $\ddot{q}=u$。引入相变量 $x_1=q$ 和 $x_2=\dot{q}$ 就可以得到一个状态向量 $x=(q,\dot{q})$。这就是说对于一个固定的 u，从任意初始状态开始的质点的运动可以由 $\mathbb{R}^2$ 上的一个向量场来获得。状态转移方程为：

$$\begin{aligned}\dot{x}_1&=x_2\\ \dot{x}_2&=u/m\end{aligned} \tag{13.51}$$

其中 $x_1=q$、$x_2=\dot{q}$ 和 $u=f$。令 $U=[-f_{\max},f_{\max}]$，其中 $f_{\max}$ 表示作用在质点上的外力的最大值。外力的值不允许任意大，因为它是不切实际的。

现在将质点的运动推广到 $\mathcal{W}=\mathbb{R}^2$ 和 $\mathcal{W}=\mathbb{R}^3$。令 n 表示 $\mathcal{W}$ 的维数，它可能是 $n=2$ 或者 $n=3$。令 q 表示质点在 $\mathcal{W}$ 中的位置。牛顿第二定律产生 $f=m\ddot{q}$，只是在这种情况下有 n 个独立方程 $f_i=m\ddot{q}_i$。每个方程都可以认为是一个独立的二重积分器。每个外力分量 f_i 可以被认为是一个行动变量 u_i。可以定义一个 $2n$ 维状态空间 $x=(q,\dot{q})$。$n=2$ 时的状态转移方程变成：

$$\begin{aligned}\dot{x}_1&=x_3 & \dot{x}_3&=u_1/m\\ \dot{x}_2&=x_4 & \dot{x}_4&=u_2/m\end{aligned} \tag{13.52}$$

对于 $n=3$，它变成：

$$\begin{aligned}\dot{x}_1&=x_4 & \dot{x}_4&=u_1/m\\ \dot{x}_2&=x_5 & \dot{x}_5&=u_2/m\\ \dot{x}_3&=x_6 & \dot{x}_6&=u_3/m\end{aligned} \tag{13.53}$$

对于一个固定的行动，这些方程分别定义了 $\mathbb{R}^4$ 和 $\mathbb{R}^6$ 上的向量场。像一维情况一样，行动集合也应该是有界的。假设：

$$U=\{u\in\mathbb{R}^n \mid \|u\|\leqslant f_{\max}\} \tag{13.54}$$

现在假设有多个外力作用在一个质点上。在这种情况下，向量和：

$$F=\sum f \tag{13.55}$$

产生所有外力系的合力。合力 F 表示单个外力，根据对质点的作用效果，它等价于组合的力。这使得牛顿第二定律可以表述为 $F=m\ddot{q}$。下面两个例子阐述了外力系所产生的状态转移方程，其中一些对应于行动。

例 13.7（月球车） 用质点的牛顿-欧拉模型可以构造一个 $X=\mathbb{R}^4$ 的例子。一辆月球车建模为在图 13.8 所示的二维世界中质量为 m 的质点。不允许它旋转，也就是说 $C=\mathbb{R}^2$。月球车上有三个推进器，它们分布在月球车的左面、右面和下面，如图 13.8 所示。每个推进器的启动被认为是一个二值开关。每个都有相应的二值行动变量，其中值 1 表示推进器正在点火，值 0 表示推进器关机。当启动以后，左面和右面的横向推进器提供的推力的大小分别为 f_l 和 f_r（左边推进器的推力向右，反之亦然）。向上的推进器安装在月球车的下面，当它启动后提供的推力的大小为 f_u。令 g 表示标量重力加速度常数（对月球来说近似为 1.622m/s^2）。

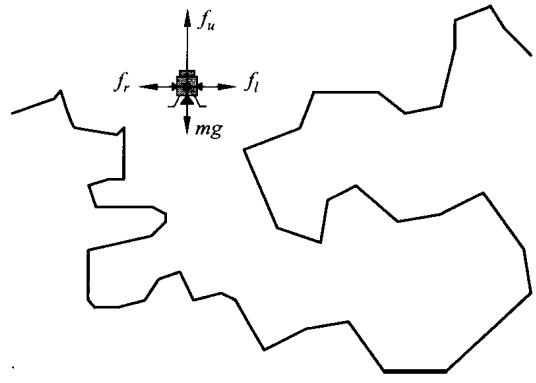


图 13.8　在月球车上有三个推进器，月球车处在月球引力下；它被当作一个质点，因此不可能有旋转；四个正交的力作用在月球车上：其中三个力来自于能够开启和关闭的推进器，剩下的一个来自于引力加速度

从式（13.55）和牛顿第二定律，有 $F=m\ddot{q}$。在水平方向，它变成：

$$m\ddot{q}_1=u_lf_l-u_rf_r \tag{13.56}$$

在垂直方向：

$$m\ddot{q}_2=u_uf_u-mg \tag{13.57}$$

方向相反的力能够进行相减，这是因为只给定力的大小 f_l、f_r、f_u 和 g。如果将它们表示为 $\mathbb{R}^2$ 中的向量，那么它们应该相加。

月球车模型可以变换成一个四维相空间，其中 $x=(q_1,q_2,\dot{q}_1,\dot{q}_2)$。分别用 $\dot{x}_3$ 和 $\dot{x}_4$ 代替 $\ddot{q}_1$ 和 $\ddot{q}_2$，那么式（13.56）和式（13.57）可以写为：

$$\dot{x}_3=\frac{1}{m}(u_lf_l-u_rf_r) \tag{13.58}$$

$$\dot{x}_4=\frac{u_uf_u}{m}-g \tag{13.59}$$

使用 $\dot{x}_1=x_3$ 和 $\dot{x}_2=x_4$，状态转移方程变成：

$$\begin{aligned}\dot{x}_1&=x_3\\ \dot{x}_2&=x_4\\ \dot{x}_3&=\frac{f_s}{m}(u_lf_l-u_rf_r)\\ \dot{x}_4&=\frac{u_uf_u}{m}-g\end{aligned} \tag{13.60}$$

它就是 $\dot{x}=f(x,u)$ 的形式。行动空间 U 有八个元素，它们表示每个推进器是开启的还是关闭的。每个行动向量的形式为 (u_l,u_r,u_u)，其中每个分量为 0 或者 1。 ■

下面几个例子阐述了牛顿第三定律的重要性。

例 13.8（单摆） 一个简单而又重要的模型是如图 13.9 所示的单摆。令 m 表示所系的质点的质量（假设绳子没有质量）。令 g 表示重力加速度常数。令 L 表示单摆的长度。令 θ 表示单摆的角位移，它描述单摆的位形。使用牛顿第二定律并假设单摆在真空中运动（没有风的阻力），得到约束：

$$mL\ddot{\theta}=-mg\sin\theta \tag{13.61}$$

那么可以表述一个二维状态空间，其中 $x_1=\theta$ 和 $x_2=\dot{\theta}$。这产生：

$$\begin{aligned}\dot{x}_1&=x_2\\ \dot{x}_2&=-\frac{g}{L}\sin x_1\end{aligned} \tag{13.62}$$

其中没有行动（式（13.62）的形式是 $\dot{x}=f(x)$）。

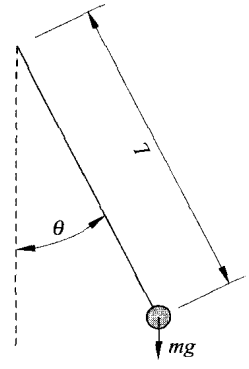


图 13.9 单摆是非线性系统的一个简单而又重要的例子

给模型加入一个线性阻力项 $kL\dot{\theta}$ 就可以考虑风的阻力影响。这产生：

$$mL\ddot{\theta}=-mg\sin\theta-kL\dot{\theta} \tag{13.63}$$

其状态空间的形式为：

$$\begin{aligned}\dot{x}_1&=x_2\\ \dot{x}_2&=-\frac{g}{L}\sin x_1-\frac{k}{m}x_2\end{aligned} \tag{13.64}$$

现在考虑外力 u_f 作用在质点上，方向垂直于绳子。这个行动可以被想象成有一个推进器固定在质点的侧面。这给式（13.63）增加了 u_f 项。它的符号取决于垂直向量的选择（向左推或者向右推）。于是状态转移方程 $\dot{x}=f(x,u)$ 变成：

$$\begin{aligned}\dot{x}_1&=x_2\\ \dot{x}_2&=-\frac{g}{L}\sin x_1-\frac{k}{m}x_2-\frac{1}{mL}u_f\end{aligned} \tag{13.65}$$

■

尽管已经给出足够信息来指定质点的微分模型，仍然还有其他概念值需要介绍，特别是在扩展到多个质点和刚体时。主要思路是可以从牛顿定律中产生守恒定律。质点的线动量（或者动量）d 定义为：

$$d=m\dot{q} \tag{13.66}$$

这可以通过 $f=m\ddot{q}$ 对时间进行积分而来。

将刚体旋转与动量矩（角动量）一起进行介绍。为得到基于矩的动量，首先需要为作用在一个点 $q\in W$ 上的力定义力矩（或者转矩）：

$$n=q\times f \tag{13.67}$$

其中“$\times$”表示 $\mathbb{R}^3$ 上的向量叉乘。对于一个具有线动量 d 的质点，动量矩 e 定义为：

$$e=q\times d \tag{13.68}$$

那么：

$$\frac{de}{dt}=n \tag{13.69}$$

它等价于牛顿第二定律，只不过是用动量的形式表达的。对于在一个封闭系统中的质点运动来说，如果没有外力作用，那么线动量和动量矩是守恒的。本质上，这是牛顿第一定律的另外一种表述。

这个思路也可以用能量来表述，能量所用变量与线动量相同。质点的动能为：

$$T=\frac{1}{2}m\dot{q}\cdot\dot{q} \tag{13.70}$$

其中“·”表述内积（或者点积）。质点系的总动能是单个质点动能之和。

13.3.2.2 质点系的运动

目前介绍的概念可以很自然地扩展到一个封闭系统中的质点系。质点系能够平滑过渡到刚体，刚体建模为“粘在一起”的无穷小质点的集合，“粘在一起”消除了相邻质点之间的力。在目前模型中，质点的运动是独立的。如果两个质点发生碰撞，那么根据牛顿第三定律可以知道它们在冲击瞬间所受的力的大小相等、方向相反。

所有动量表达都可以扩展到多个质点之和[684]。对于质点系，每个线动量的和就产生了系统的线动量：

$$D=\sum d \tag{13.71}$$

总的外力为：

$$F=\sum f_i \tag{13.72}$$

它是整个系统的合力。$dD/dt=F$ 是对单个质点情况的扩展。求和产生总质量：

$$M=\sum m \tag{13.73}$$

系统的质心为：

$$p=\frac{1}{M}\sum mq \tag{13.74}$$

其中 m 和 q 分别为每个质点的质量和位置。$D=Mp$ 和 $F=M\dot{p}$ 类似于单个质点的 $d=m\dot{q}$ 和 $f=m\ddot{q}$。

到现在已经介绍了运动的平移部分；但是系统的旋转也很重要。这里引入力矩概念的目的就是为了研究旋转。令总力矩（或者总转矩）为：

$$N=\sum q\times f \tag{13.75}$$

并令系统的动量矩为：

$$E=\sum q\times d \tag{13.76}$$

那么 $dE/dt=N$，这与单个质点的情况相同。

目前所给的思路使质点系与单个质点很相似。但有时还需要考虑质点间的碰撞，检测碰撞和计算碰撞所产生的冲击力以确保得到正确的运动。

当质点的数量趋近无穷时，考虑刚体的极限情况。在这种情况下，质点被“缝”在一起，这样就抵消了它们的内力。只处理作用在刚体边界上的力就足够了。第 13.3.3 节给出刚体运动的表达式。这些表达式也可以用诸如第 13.4 节中的其他概念来得到。

13.3.3 刚体运动

对于一个自由的三维刚体，第 4.2.2 节指出它的 C-空间 $\mathcal{C}$ 有 6 维。假设行动作为外力而作用于物体。这些力直接导致二阶微分方程的加速度。定义一个状态 $(q,\dot{q})$，得到 12 维相空间 X 中的一阶微分方程。

令 $\mathcal{A}\subseteq\mathbb{R}^3$ 表示一个自由的刚体。令 $\sigma(r)$ 表示 $r\in\mathcal{A}$ 处的密度。令 m 表示 $\mathcal{A}$ 的总质量，它是用密度定义的：

$$m=\int_{\mathcal{A}}\sigma(r)\mathrm{d}r \tag{13.77}$$

其中 $\mathrm{d}r=\mathrm{d}r_1\mathrm{d}r_2\mathrm{d}r_3$ 表示 $\mathbb{R}^3$ 中的体积微元。令 $p\subseteq\mathbb{R}^3$ 表示 $\mathcal{A}$ 的质心，对于 $p=(p_1,p_2,p_3)$ 它定义为：

$$p_i=\frac{1}{m}\int_{\mathcal{A}}r_i\sigma(r)\mathrm{d}r \tag{13.78}$$

假设外力的一个集合作用在 $\mathcal{A}$ 上（假设 $\mathcal{A}$ 中的所有内力相互抵消）。每个力 f 作用在边界的一个点上，如图 13.10 所示（注意到也可以用沿着力的任意点）。力的集合可以合成关于质心 p 的单个力和力矩。令 F 表示作用在 $\mathcal{A}$ 上的总外力。令 N 表示关于 $\mathcal{A}$ 质心的总外力矩。对于外力系来说，它们是：

$$F=\sum f \tag{13.79}$$

$$N=\sum r\times f \tag{13.80}$$

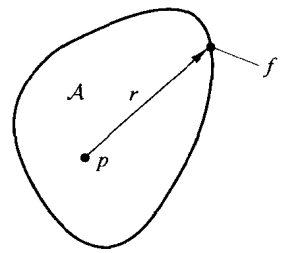


图 13.10 作用在 $\mathcal{A}$ 的 r 处的力 f 产生了一个关于 p 的力矩 $r\times f$

F 和 N 通常被称为力系的合力和合力矩。Poinsot 指出每个力系等价于平行于力的方向上的单个力和力矩。力矩的结果是产生扭转，这类似于拧螺钉，更加详细的讨论见文献[684]。

形式为 $u\in U$ 的行动可以表示为作用在刚体上的外力和/或力矩。例如推进器启动时可以对载体施加作用力。对于给定的 u，计算总的力和力矩可以得到 $F(u)$ 和 $N(u)$。

重要的坐标系 有三种非常重要的坐标系。

（1）惯性坐标系：固连在所感兴趣的运动上的全局坐标系。

（2）平移坐标系：原点在 $\mathcal{A}$ 的质心、轴平行于惯性坐标系的一个动坐标系。

（3）体坐标系：原点在 $\mathcal{A}$ 的质心、轴固连在 $\mathcal{A}$ 上的一个坐标系。它与第 3 章用来定义载体的坐标系相同。

平移 状态转移方程涉及 12 个标量方程。其中 6 个描述线速度。对于这种情况，可以假设物体关于惯性坐标系没有旋转。线动量为 $D=m\dot{p}$，根据牛顿第二定律，那么有：

$$F(u)=\frac{\mathrm{d}D}{\mathrm{d}t}=m\ddot{p} \tag{13.81}$$

解 $\ddot{p}$ 就产生一半状态转移方程。这产生了例 13.3 中二重积分器的三维版本，只是以 m 为刻度因子。令 (p_1,p_2,p_3) 表示 p 的坐标。令 (v_1,v_2,v_3) 表示质心的线速度。状态方程的三个标量方程为 $\dot{p}_i=v_i$，$i=1,2,3$。另外三个方程为 $\dot{v}_i=F_i(u)/m$，$i=1,2,3$。如果没有力矩并且物

体关于惯性坐标系没有旋转，那么这 6 个方程足以描述物体的运动。初始静止并且推进器推力通过质心的航天器会发生这种情况发生。

旋转　即使 $\mathcal{A}$ 关于惯性坐标系发生旋转，上述的 6 个方程也依然有效。它们只是运动的平移部分。用平移坐标系可以使旋转和平移解耦。因为已经讨论过运动的平移，所以假设 $\mathcal{A}$ 只进行旋转，而它的质心保持固定。一旦确定运动的旋转，那么用惯性坐标系描述物体就可以将旋转与平移组合起来。因此现在只考虑 $\mathcal{A}$ 关于平移坐标系的运动，这使得物体只有旋转运动。

刻画旋转比平移和二维旋转复杂得多。那么第 3 章和第 4 章中与三维旋转有关的困难也就不令人惊讶了。

按照牛顿第二定律，动量矩的变化为：

$$N(u)=\frac{\mathrm{d}E}{\mathrm{d}t} \tag{13.82}$$

问题是以能够并入状态方程的形式表达式（13.82）的右边。

微分旋转　为了详细表达动量矩的变化，需要用到微分旋转这个概念。在平面情况下，很容易定义 $\omega=\dot{\theta}$，但是对于 $SO(3)$，情况就非常复杂。一种方法是定义关于航向-俯仰-滚动变量的导数，但是这会导致扭曲和奇异，这是采用牛顿-欧拉表示的问题所在。取而代之的是定义微分旋转，如图 13.11 所示。令 v 表示 $\mathbb{R}^3$ 中的单位向量，令 θ 表示一个旋转。令 ω 表示角速度向量：

$$\omega=v\frac{\mathrm{d}\theta}{\mathrm{d}t} \tag{13.83}$$

这是角速度的一个非常自然的表示方法[1]。旋转矩阵 R 关于时间的变化为：

$$\dot{R}=\omega\times R \tag{13.84}$$

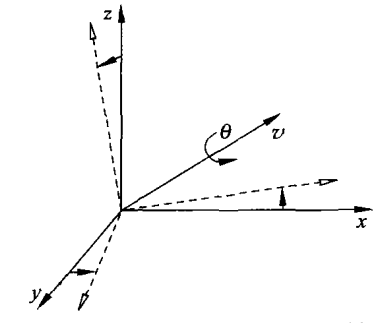


图 13.11　角速度被定义为坐标轴关于一个轴旋转的速率

这可以产生 ω 和航向-俯仰-滚动角或者四元数的关系表达式。例如用航向-俯仰-滚动矩阵（3.42），从 ω 到航向、俯仰、滚动角变化的转换为：

$$\begin{pmatrix}\dot{\gamma}\\ \dot{\beta}\\ \dot{\alpha}\end{pmatrix}=\frac{1}{\cos\beta}\begin{pmatrix}\cos\alpha & \sin\alpha & 0\\ -\sin\alpha\cos\beta & \cos\alpha\cos\beta & 0\\ \cos\alpha\sin\beta & \sin\alpha\sin\beta & -\cos\beta\end{pmatrix}\begin{pmatrix}\omega_1\\ \omega_2\\ \omega_3\end{pmatrix} \tag{13.85}$$

惯性矩阵　将 $\mathcal{A}$ 看作是固连在一起的质点系（由牛顿第三定律知道它们之间的作用力互相抵消）就可以产生惯性矩阵（也被称为惯性张量或者惯性算子）。式（13.77）中的 $\sigma(r)\mathrm{d}r$ 表示 $\mathcal{A}$ 的无穷小质点的质量。其动量矩为 $r\times\dot{r}\sigma(r)\mathrm{d}r$。总的动量矩为：

$$E=\int_{\mathcal{A}(q)}(r\times\dot{r})\sigma(r)\mathrm{d}r \tag{13.86}$$

因为 $\dot{r}=\omega\times r$，那么表达式变成：

$$E=\int_{\mathcal{A}(q)}r\times(\omega\times r)\sigma(r)\mathrm{d}r \tag{13.87}$$

1　意识到一个重要问题是 ω 的积分不是路径不变的（path-invariant）（见文献[994]中的例 2.15）。

其中 r 在被积函数中出现两次。通过代数运算，ω 可以从被积函数中去掉，所得到的函数是 r 的二次型（因为 r 是一个向量，所以函数是二次型的形式）。第一步应用等式 $a\times(b\times c)=(a\bullet c)b-(a\bullet b)c$ 得到：

$$E=\int_{A(q)}((r\bullet r)\omega-(r\bullet\omega)r)\sigma(r)\mathrm{d}r \tag{13.88}$$

将角速度提取出来，得到：

$$E=\left(\int_{A(q)}((r\bullet r)I_3-rr^T)\sigma(r)\mathrm{d}r\right)\omega \tag{13.89}$$

其中积分是在3×3矩阵上进行的，I_3 是3×3的单位阵。

将 I 称为惯性矩阵（inertia matrix），它定义为：

$$I(q)=\int_{A(q)}((r\bullet r)I_3-rr^T)\sigma(r)\mathrm{d}r \tag{13.90}$$

那么：

$$E=I\omega \tag{13.91}$$

这使得式（13.82）可以简化为：

$$N(u)=\frac{\mathrm{d}E}{\mathrm{d}t}=\frac{\mathrm{d}(I\omega)}{\mathrm{d}t}=I\frac{\mathrm{d}\omega}{\mathrm{d}t}+\frac{\mathrm{d}I}{\mathrm{d}t}\omega \tag{13.92}$$

简化惯性矩阵 现在仔细考虑惯性矩阵。它是一个对称的3×3矩阵，可表示为：

$$I(q)=\begin{pmatrix}I_{11}(q) & I_{12}(q) & I_{13}(q)\\ I_{12}(q) & I_{22}(q) & I_{23}(q)\\ I_{13}(q) & I_{23}(q) & I_{33}(q)\end{pmatrix} \tag{13.93}$$

对于每个 $i\in\{1,2,3\}$，元素 $I_{ii}(q)$ 称为转动惯量（moment of inertia）。它们是：

$$I_{11}(q)=\int_{A(q)}(r_2^2+r_3^2)\sigma(r)\mathrm{d}r \tag{13.94}$$

$$I_{22}(q)=\int_{A(q)}(r_1^2+r_3^2)\sigma(r)\mathrm{d}r \tag{13.95}$$

$$I_{33}(q)=\int_{A(q)}(r_1^2+r_2^2)\sigma(r)\mathrm{d}r \tag{13.96}$$

其余的元素定义如下。对于每个 $i,j\in\{1,2,3\}$，并且 $i\neq j$，惯量积（product of inertia）为：

$$H_{ij}(q)=\int_{A(q)}r_ir_j\sigma(r)\mathrm{d}r \tag{13.97}$$

且 $I_{ij}(q)=-H_{ij}(q)$。

这种表述存在的一个问题是当物体旋转时惯性矩阵也会发生变化，因为矩阵的所有元素与方向 q 有关。这是由于将 $\mathcal{A}$ 当作平移坐标系中的质点系而产生的。但是也可能在 $\mathcal{A}$ 的体坐标系中表示惯性矩阵。在这种情况下，惯性矩阵可以表示为 I，因为它不依赖于 $\mathcal{A}$ 关于平移坐标系的方向。通过用体坐标系和平移坐标系之间的旋转来恢复原始惯性矩阵：$I(q)=RI$，其中 R 是一个旋转矩阵。（从方程（2.91）和文献[994]的第3.2节）可以看到：进行替换以后，式（13.92）简化为：

$$N(u)=I\frac{\mathrm{d}\omega}{\mathrm{d}t}+\omega\times(I\omega) \tag{13.98}$$

$\mathcal{A}$ 的体坐标系的原点必须在质心，但是对它的方向没有限制。不同的方向可以产生不

同的惯性矩阵。因为 I 含有 $\mathcal{A}$ 的物理特征，任意两个惯性矩阵仅在旋转上有所不同。这意味着对于给定的 $\mathcal{A}$，用不同体坐标系的方向定义的所有惯性矩阵有相同的特征值和特征向量。考虑正定二次型 $x^T Ix = 1$，它表示椭球体的方程。线性代数中一个基本内容是计算椭球体的主轴，该主轴就是 I 的特征向量。椭球体主轴的长度由特征值给定。通过定义 $x' = Rx$ 可以得到椭球体的一个轴向表示，其中 R 是特征向量的列所构成的矩阵。因此存在体坐标系的一个方向，它的惯性矩阵简化为：

$$I = \begin{pmatrix} I_{11} & 0 & 0 \\ 0 & I_{22} & 0 \\ 0 & 0 & I_{33} \end{pmatrix} \tag{13.99}$$

并且对角线元素为特征值。如果物体恰好是一个椭球体，那么主轴对应于椭球体的轴。许多文献中给出了转动惯量表[693]；在这些情况下，通常选择主轴作为体坐标系的轴，因为它们所产生的 I 的表达式最简单。

完成状态方程　假设 $\mathcal{A}$ 的体坐标系对准主轴。其余六个运动方程可以用更好的形式给出。用式（13.99），表达式（13.98）被简化为[684]：

$$\begin{pmatrix} N_1(u) \\ N_2(u) \\ N_3(u) \end{pmatrix} = \begin{pmatrix} I_{11} & 0 & 0 \\ 0 & I_{22} & 0 \\ 0 & 0 & I_{33} \end{pmatrix} \begin{pmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{pmatrix} + \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \begin{pmatrix} I_{11} & 0 & 0 \\ 0 & I_{22} & 0 \\ 0 & 0 & I_{33} \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} \tag{13.100}$$

那么得到：

$$\begin{aligned} N_1(u) &= I_{11}\dot{\omega}_1 + (I_{33} - I_{22})\omega_2\omega_3 \\ N_2(u) &= I_{22}\dot{\omega}_2 + (I_{11} - I_{33})\omega_3\omega_1 \\ N_3(u) &= I_{33}\dot{\omega}_3 + (I_{22} - I_{11})\omega_1\omega_2 \end{aligned} \tag{13.101}$$

为了得到状态方程形式，求解 $\dot{\omega}$ 产生：

$$\begin{aligned} \dot{\omega}_1 &= \big(N_1(u) + (I_{22} - I_{33})\omega_2\omega_3\big)/I_{11} \\ \dot{\omega}_2 &= \big(N_2(u) + (I_{33} - I_{11})\omega_3\omega_1\big)/I_{22} \\ \dot{\omega}_3 &= \big(N_3(u) + (I_{11} - I_{22})\omega_1\omega_2\big)/I_{33} \end{aligned} \tag{13.102}$$

最后一个问题是需要将 ω 与用来表示 $SO(3)$ 的元素的角度联系起来。它们之间的映射依赖于 $SO(3)$ 的特定参数表示。假设用形式为 (a,b,c,d) 的四元数来表示旋转。我们知道一旦给出 b、c 和 d，那么 a 可以用 $a^2 + b^2 + c^2 + d^2 = 1$ 来确定。ω 与四元数分量的导数之间的关系用式（13.84）得到（见文献[693]的第 433 页）：

$$\begin{aligned} \dot{b} &= \omega_3 c - \omega_2 d \\ \dot{c} &= \omega_1 d - \omega_3 b \\ \dot{d} &= \omega_2 b - \omega_1 c \end{aligned} \tag{13.103}$$

这就建立了 $\dot{x} = f(x,u)$ 形式的状态方程，其中：

$$x = (p_1, p_2, p_3, v_1, v_2, v_3, b, c, d, \omega_1, \omega_2, \omega_3) \tag{13.104}$$

是一个 12 维的相向量。为方便起见，完整的状态方程为：

$$\begin{aligned}
\dot{p}_1 &= v_1 & \dot{b} &= \omega_3 c - \omega_2 d \\
\dot{p}_2 &= v_2 & \dot{c} &= \omega_1 d - \omega_3 b \\
\dot{p}_3 &= v_3 & \dot{d} &= \omega_2 b - \omega_1 c \\
\dot{v}_1 &= F_1(u)/m & \dot{\omega}_1 &= \left(N_1(u) + (I_{22} - I_{33})\omega_2\omega_3\right)/I_{11} \\
\dot{v}_2 &= F_2(u)/m & \dot{\omega}_2 &= \left(N_2(u) + (I_{33} - I_{11})\omega_3\omega_1\right)/I_{22} \\
\dot{v}_3 &= F_3(u)/m & \dot{\omega}_3 &= \left(N_3(u) + (I_{11} - I_{22})\omega_1\omega_2\right)/I_{33}
\end{aligned} \tag{13.105}$$

惯性矩阵与椭球体之间的关系实际上更加密切。由于旋转而产生的动能表示为：

$$T = \frac{1}{2}\omega^T I \omega \tag{13.106}$$

在文献[39,684]中 Poinsot 给出了一种在没有外力的情况下对旋转运动的解释。当物体旋转时，它的运动等价于式（13.106）给出的惯量椭球体沿着以 $\mathbb{R}^3$ 中 $I\omega$ 为法向量的平面向下滚动（没有滑动）。

二维情况 在平面内运动的二维刚体的动力学可以当作三维刚体的特殊情况来处理。令 $\mathcal{A} \subset \mathbb{R}^2$ 为一个二维物体，用体坐标系来表达它。作用在 $\mathcal{A}$ 上的总外力可以根据通过质心的二维总外力和力矩来表示。这种模型的相空间有六维。三个来自于 $SE(2)$ 的自由度，两个来自于线速度，一个来自于角速度。

平移部分仍表示为：

$$F(u) = \frac{\mathrm{d}D}{\mathrm{d}t} = m\ddot{p} \tag{13.107}$$

这给出了状态方程的四个分量。

在二维表述中，所有旋转必须关于 z 轴进行。这意味着角速度 ω 是一个标量。令 θ 表示 $\mathcal{A}$ 的方向。ω 和 θ 的关系为 $\dot{\theta} = \omega$，这产生了状态方程的另外一个分量。

还剩下一个分量。回忆式（13.92），通过式（13.101）可以看到有些基于惯量的项为零。在那里，ω_3 等价于二维情况下的标量 ω。三个方程的最后一项都为零，因为 $\omega_1 = \omega_2 = 0$。前两个方程的第一项也为零，因为 $\dot{\omega}_1 = \dot{\omega}_2 = 0$。只剩下 $N_3(u) = I_{33}\dot{\omega}_3$。在二维情况下，这可以简化为：

$$N(u) = \frac{\mathrm{d}E}{\mathrm{d}t} = \frac{\mathrm{d}(I\omega)}{\mathrm{d}t} = I\frac{\mathrm{d}\omega}{\mathrm{d}t} = I\dot{\omega} \tag{13.108}$$

其中 I 现在是一个标量。注意到对于三维情况，即使 $N(u) = 0$，角速度也可以发生变化。但是在二维情况下，这是不可能的。在这两种情况下，动量矩是守恒的；在二维情况下，这恰好意味着 ω 是固定的。对式（13.108）求解 $\dot{\omega}$ 可以得到状态方程的第六个分量。

平面中的二维刚体的状态方程因此为：

$$\begin{aligned}
\dot{p}_1 &= v_1 & \dot{v}_1 &= F_1(u)/m \\
\dot{p}_2 &= v_2 & \dot{v}_2 &= F_2(u)/m \\
\dot{\theta} &= \omega & \dot{\omega} &= N(u)/I
\end{aligned} \tag{13.109}$$

轮胎打滑的汽车 引入一个汽车模型，将它当作在平面内滑动的刚体。这种模型由 Jim Bernard 提出。C 空间为 $C = \mathbb{R}^2 \times \mathbb{S}^1$，其中 $q = (x, y, \theta)$。假设当汽车高速行驶时，轮胎能够在垂直于汽车主轴的方向上横向滑动（例如平行于后轮轴）。令 ω 表示汽车的角速度。令 v

表示横向滑动速度，它是另外一个状态变量。这样就产生了一个五维的状态空间，其中每个状态都是一个 $q=(x,y,\theta,\omega,v)$ 形式的向量。

后轮轴中心的位置可以表示为：

$$\begin{aligned}\dot{x}&=s\cos\theta-v\sin\theta\\ \dot{y}&=s\sin\theta+v\cos\theta\end{aligned}\tag{13.110}$$

它是状态方程的两个分量。令 $\omega=\dot{\theta}$ 表示角速度，它产生状态方程的另外一维分量。还有两个分量，它们产生于二维刚体力学（将在第 13.3.3 节介绍）。状态方程为：

$$\begin{aligned}\dot{x}&=s\cos\theta-v\sin\theta\\ \dot{y}&=s\sin\theta+v\cos\theta\\ \dot{\theta}&=\omega\\ \dot{\omega}&=(af_f-bf_r)/I\\ \dot{v}&=-s\omega+(f_f+f_r)/m\end{aligned}\tag{13.111}$$

其中 f_f 和 f_r 是前后轮胎受的力，m 是质量，I 是转动惯量，a 和 b 分别为质心到前后轮轴的距离。第一个力为：

$$f_f=c_f((v+a\omega)/s+\phi)\tag{13.112}$$

其中 c_f 是前转弯刚度系数，ϕ 是转向角。第二个力为：

$$f_r=c_r(v-b\omega)/s\tag{13.113}$$

其中 c_r 是后转弯刚度系数。转向角可以指定为一个行动变量：$u_\phi=\phi$。一个积分器放置在速度的前面，这样就允许有加速度。当然状态空间的维数也增加了一维。

汽车中合理的参数值为 $m=1460\text{kg}$，$c_f=17\ 000$，$c_r=20\ 000$，$a=1.2\text{m}$，$b=1.5\text{m}$，$I=2170\text{kg/m}^2$ 和 $s=27\text{m/s}$。这样的状态方程涉及轮胎线性滑动模型，在许多应用中这种模型对滑动的描述并不理想。非线性轮胎模型则能够提供对实际汽车行为的更好近似[93]。对汽车动力学更全面的介绍，参见文献[825]。

13.4 先进的力学概念

牛顿-欧拉力学的优势在于它从最基本的原理出发，其局限性在于很难对复杂的机械系统进行建模。其中一个主要不足是所有定律必须在直角惯性坐标系中进行表达。本节介绍拉格朗日力学和哈密顿力学的基本原理，它们把力学问题变成为用 C 空间的任意坐标邻域来寻找最优路径的问题，这样它们克服了牛顿-欧拉力学的局限性。最优性指标用能量表示。所产生的技术可以应用于一个平滑流形的任意坐标邻域。对于确定一个或者多个物体的运动来说，拉格朗日表述通常是最好的方法。第 13.4.1 节基于变分法来介绍基本拉格朗日概念。第 13.4.2 节提出欧拉-拉格朗日方程的一般形式，这对于确定大量动力系统的运动来说非常有用，其中包括物体的链。拉格朗日力学对于涉及诸如摩擦或者滚动等微分约束的系统来说也很方便。这些情况将在第 13.4.3 节进行简要介绍。第 13.4.4 节的哈密顿表述是基于特殊相空间，它提出了拉格朗日以外的一种可选方法。该方法归纳产生庞特里亚金最小值原理，该原理是第 15.2.3 节介绍的最优控制技术的内容。

13.4.1 拉格朗日力学

13.4.1.1 变分法

拉格朗日力学基于变分法，变分法是在路径空间中进行优化的学科。其中最著名的变分问题涉及限制一个质点沿着一条曲线运动（假设质点沿着一条无摩擦的路径进行滑动）。问题是寻找球从一个点到另外一个点的运动曲线，它从静止开始，仅受重力影响。解是称为最速降线（brachistochrone curve）的摆线函数[844]。在进一步描述该问题之前，回忆寻找函数极值（最小值和最大值）的微积分学中经典优化问题。令 $\tilde{x}$ 表示从 $\mathbb{R}$ 到 $\mathbb{R}$ 的平滑函数，$x(t)$ 表示对于任意 $t\in\mathbb{R}$ 的函数的值。从微积分学，$\tilde{x}$ 的极值是使所有 $t\in\mathbb{R}$ 的 $\dot{x}=0$。假设在某 $t'\in\mathbb{R}$，$\tilde{x}$ 实现了局部最小值。作为一个最小值，t' 的一个小的扰动将造成很大的函数值。因此存在某 $d>0$，对于任意 $\varepsilon\in[-d,d]$，使得 $x(t'+\varepsilon)>x(t')$。每个 ε 表示 t' 的一个可能的扰动。

变分法将讨论一种更难的问题，它要求在函数空间上进行优化。对于每个函数，它是泛函[1]所赋予的值。进行的是类似于求函数的导数并置零的过程。这考虑的是整个函数的小扰动，而不是上面所提及的 ε 扰动。每个扰动自身都是一个函数，它被称为变分（variation）。对于最小化一个泛函的函数来说，任意足够小的扰动必须产生一个更大的泛函值。在优化一个变量的函数时，扰动仅有两个方向：$\pm\varepsilon$（见图 13.12）。在变分法中，有许多不同的“方向”，由于有不可数无限多种方法来构造扰动原始函数的小的变分函数（所有变分的集合是无穷维函数空间，回忆例 8.5）。

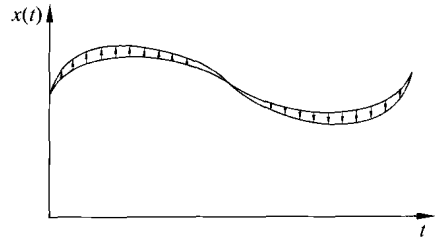


图 13.12 变分是一个“小”的函数，它加到 $\tilde{x}$ 上对 $\tilde{x}$ 进行扰动

令 $\tilde{x}$ 表示从 $T=[t_0,t_1]$ 到 $\mathbb{R}$ 的平滑函数。泛函通过在 $\tilde{x}$ 的定义域上对函数进行积分来定义。令 L 表示变量 a、b 和 c 的一个平滑的实值函数[2]。L 的变量可能是任意 $a,b\in\mathbb{R}$ 和 $c\in T$，即函数是 $L(a,b,c)$，但是每个变量都有其特殊的意义。对于某平滑函数 $\tilde{x}$，用 L 来计算它在特定 $t\in T$ 时的值 $L(x,\dot{x},t)$。用 L 构造泛函 Φ 来计算整个函数 $\tilde{x}$：

$$\Phi(\tilde{x})=\int_T L(x(t),\dot{x}(t),t)\mathrm{d}t \tag{13.114}$$

问题是选择一个优化 Φ 的 $\tilde{x}$。方法是对 Φ 求导并置零，就像标准微积分一样，但是 Φ 关于 $\tilde{x}$ 进行微分并不是标准的微积分。实际上对可能函数的类型（如平滑性）以及变分的向量空间都有特殊要求，对于本节所考虑的问题来说假设这些条件都满足。

例 13.9（最短路径泛函） 考虑一个泛函的例子：

$$L(x,\dot{x},t)=\sqrt{1+\dot{x}^2} \tag{13.115}$$

当在一个函数 $\tilde{x}$ 上计算时，它产生路径的弧长。 ■

1 这就是为什么在本书使用代价泛函的原因。它是函数空间上的一个函数。

2 这里用 L 来表示一个代价函数，泛函 Φ 就是基于该函数。这与用 l 作为代价函数、L 作为运动规划表述中的泛函相抵触。这种符号的冲突以后还会存在，因为 L 是拉格朗日函数的标准符号。注意不要混淆。

另外一个泛函的例子已经在运动规划中出现过。代价泛函（8.39）给经过状态空间的一条路径赋予了一个代价。这给表述最优路径规划提供了一个非常自然的方法。式（7.26）给出了一个离散的近似方法。

令 h 表示 T 上一个平滑函数，令 ε 表示一个小的常数。考虑函数 $x(t)+\varepsilon h(t)$，$t\in[0,1]$。如果 $\varepsilon=0$，那么式（13.114）保持相同。当 ε 增加或者减小时，那么 $\Phi(\tilde{x}+\varepsilon h)$ 可能发生变化。函数 h 就像方向导数中的“方向”。如果对于任意平滑函数 h，存在某 $\varepsilon>0$ 使得值 $\Phi(\tilde{x}+\varepsilon h)$ 增加，那么 $\tilde{x}$ 被称为 Φ 的一个极值点。对 $\tilde{x}$ 的任何小的扰动都会引起 Φ 值增加。因此 $\tilde{x}$ 的行为就像标准优化问题中的一个局部最小值。

对于某 $\varepsilon>0$ 和函数 h，令 $g=\varepsilon h$。可以用下式近似一个泛函的微分[39]：

$$\begin{aligned}\Phi(\tilde{x}+g)-\Phi(\tilde{x}) &= \int_T (L(x(t)+g(t),\dot{x}(t)+\dot{g}(t),t)-L(x(t),\dot{x}(t),t))\,\mathrm{d}t+\cdots \\ &= \int_T\left(\frac{\partial L}{\partial x}g+\frac{\partial L}{\partial \dot{x}}\dot{g}\right)\mathrm{d}t+\cdots \\ &= \int_T\left(\frac{\partial L}{\partial x}g-\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}}g\right)\mathrm{d}t+\left(\frac{\partial L}{\partial \dot{x}}g\right)\Big|_{t_0}^{t_1}+\cdots\end{aligned} \tag{13.116}$$

其中“…”表示将会在极限中成为零的高阶项。最后一步产生于部分积分：

$$\left(\frac{\partial L}{\partial \dot{x}}g\right)\Big|_{t_0}^{t_1}=\int_T\frac{\partial L}{\partial \dot{x}}\dot{g}\mathrm{d}t+\int_T\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}}h\mathrm{d}t \tag{13.117}$$

即 $uv=\int v\mathrm{d}u+\int u\mathrm{d}v$。当 ε 变小时，考虑式（13.116）的值，并假设 $h(t_0)=h(t_1)=0$。为使 $\tilde{x}$ 成为一个极值函数，当变分接近零时，式（13.116）所表达的变化应该趋近于零。基于变分法基本引理（见文献[39]的第 12 节）等更进一步的假设，那么欧拉-拉格朗日方程：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}}-\frac{\partial L}{\partial x}=0 \tag{13.118}$$

是 $\tilde{x}$ 成为极值的必要条件。研究式（13.116）的最后一行可以产生一个直觉。当满足式（13.118）时积分项为零。因为 $h(t_0)=h(t_1)=0$，所以其他项也为零，并且在求取极限的过程中更高阶的项消失。

L 关于 $\dot{x}$ 和 x 的偏导数用标准微积分来定义。导数 $\partial L/\partial\dot{x}$ 的计算是将 $\dot{x}$ 当作普通变量来对待的（例如就像 $\partial L/\partial b$，当变量像 $L(a,b,c)$ 中那样命名时）。顺着这种思路，也可以产生 $\partial L/\partial\dot{x}$ 关于 t 的导数。为了阐述这个过程，考虑如下例子。

例 13.10（简单变分问题） 令 L 为一个泛函，它定义为：

$$L(x,\dot{x},t)=x^3+\dot{x}^2 \tag{13.119}$$

关于 $\dot{x}$ 和 x 的偏导数为：

$$\frac{\partial L}{\partial x}=3x^2 \tag{13.120}$$

$$\frac{\partial L}{\partial \dot{x}}=2\dot{x} \tag{13.121}$$

那么式（13.121）对时间的导数为：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}}=2\ddot{x} \tag{13.122}$$

将它们带入欧拉-拉格朗日方程（13.118）就产生了：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}}-\frac{\partial L}{\partial x}=2\ddot{x}-3x^2=0 \tag{13.123}$$

这表示一个二阶微分约束，它将加速度约束为 $\ddot{x}=3x^2/2$。构造一个二维相空间，可以用一阶微分方程来表示该约束。 ■

13.4.1.2 哈密顿最小作用原理

现在已经介绍了机械系统动力学问题的背景知识。物体系统的 C-空间的路径可以表示为最优化动能和势能之差的变分问题的解。变分法是对 C 的任意坐标邻域的推广。在这种情况下，欧拉-拉格朗日方程为：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}}-\frac{\partial L}{\partial q}=0 \tag{13.124}$$

其中 q 是一个有 n 个坐标的向量。它通常是如下形式的 n 个标量方程：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_i}-\frac{\partial L}{\partial q_i}=0 \tag{13.125}$$

下面用式（13.124）来得到一个相空间转移方程。这通过优化一个定义为动能和势能变化的泛函来产生。第 13.3.1 节定义了质点和刚体的动能。一般情况下，动能函数必须是 $\dot{q}$ 的一个二次函数。它的定义可以解释为 C 上的一个内积，它使 C 变成一个黎曼流形[158]。这给流形提供了一个速度向量之间“角”的概念，并产生了曲率的概念和被称为测地线（geodesics）的最短路径。令 $K(q,\dot{q})$ 表示用流形坐标来表示的动能，其形式总是：

$$K(q,\dot{q})=\frac{1}{2}\dot{q}^T M(q)\dot{q} \tag{13.126}$$

其中 $M(q)$ 是一个 $n\times n$ 矩阵，称为质量矩阵或者惯性矩阵。

下面定义势能。一个系统称为是保守的，如果作用在一个点上的力仅依赖于点的位置，并且力沿着路径所做的功只依赖于路径的终点。在保守系统的运动下总能量是守恒的。在这种情况下，存在一个势函数 $\phi:W\to\mathbb{R}$ 使得对于任意 $p\in W$，有 $F=\partial\phi/\partial p$。令 $V(q)$ 表示物体系在位形 q 下的总势能。

假设动力学是时不变的。哈密顿最小作用原理认为，一个机械系统的轨迹 $\tilde{q}:T\to C$ 就是如下用 C 的任意坐标邻域表示的泛函的极值：

$$\Phi(\tilde{q})=\int_T\big(K(q(t),\dot{q}(t))-V(q(t))\big)\mathrm{d}t \tag{13.127}$$

用式（13.124）的形式表示牛顿第二定律就可以得到 $C=\mathbb{R}^3$ 情况下的哈密顿最小作用原理[39]：

$$\frac{\mathrm{d}}{\mathrm{d}t}(m\dot{q})-\frac{\partial V}{\partial q}=0 \tag{13.128}$$

其中力用势能的导数来代替。那么可以将欧拉-拉格朗日方程应用到泛函：

$$L(q,\dot{q})=K(q,\dot{q})-V(q) \tag{13.129}$$

其中已经假设动力学是时不变的，因此 $L(q,\dot{q},t)=L(q,\dot{q})$。对式（13.127）应用欧拉-拉格朗日方程就产生了极值。

采用拉格朗日表述的优势在于 C-空间不必是在惯性坐标系中描述的 $C=\mathbb{R}^3$。欧拉-拉

格朗日方程给出在一个机械系统的任意 C-空间中运动的必要条件。条件可以用任意坐标邻域来表示，而不一定是牛顿-欧拉表述所必需的直角坐标系。在关于力学的文献中，变量 q 通常被称为广义坐标（generalized coordinates）。这是平滑流形的任意坐标邻域所给出的坐标。

用式（13.129）所得到的式（13.124）的特殊形式产生了对运动的约束：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}}-\frac{\partial L}{\partial q}=\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial K(q,\dot{q})}{\partial \dot{q}}-\frac{\partial K(q,\dot{q})}{\partial q}+\frac{\partial V(q)}{\partial q}=0 \tag{13.130}$$

这表示 n 个方程，每个坐标 q_i 对应于一个方程。因为 $K(q,\dot{q})$ 不依赖于时间，在计算 $\partial K(q,\dot{q})/\partial \dot{q}$ 的表达式中 $\mathrm{d}/\mathrm{d}t$ 算子用 $\ddot{q}$ 代替 $\dot{q}$。$\ddot{q}$ 的出现似乎是适当的，因为所产生的微分方程是二阶的，这与牛顿-欧拉力学相一致。

例 13.11（一个下落的质点） 假设一个质量为 m 的质点在 $\mathbb{R}^3$ 中下落。令 (q_1,q_2,q_3) 表示质点的位置。令 g 表示在 $-q_3$ 方向上的重力加速度。势能为 $V(q)=mgq_3$。动能为：

$$K(q,\dot{q})=\frac{1}{2}m\dot{q}\cdot\dot{q}=\frac{1}{2}m(\dot{q}_1^2+\dot{q}_2^2+\dot{q}_3^2) \tag{13.131}$$

拉格朗日函数为：

$$L(q,\dot{q})=K(q,\dot{q})-V(q)=\frac{1}{2}m(\dot{q}_1^2+\dot{q}_2^2+\dot{q}_3^2)-mgq_3=0 \tag{13.132}$$

为了得到对质点运动的微分约束，使用式（13.130）。对于 1～3 的每个 i，s 有：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}}=\frac{\mathrm{d}}{\mathrm{d}t}(m\dot{q}_i)=m\ddot{q}_i \tag{13.133}$$

因为 $K(q,\dot{q})$ 不依赖于 q，所以对于每个 i，$\partial K/\partial q_i=0$。势能的导数为：

$$\frac{\partial V}{\partial q_1}=0 \qquad \frac{\partial V}{\partial q_2}=0 \qquad \frac{\partial V}{\partial q_3}=mg \tag{13.134}$$

代入（13.130）产生三个方程：

$$m\ddot{q}_1=0 \qquad m\ddot{q}_2=0 \qquad m\ddot{q}_3+mg=0 \tag{13.135}$$

那么加速度仅出现在 $-q_3$ 方向上，并由重力引起。方程与牛顿定律相一致。通常情况下，定义六维相空间就可以得到一阶微分约束。■

哈密顿原理的“最小”其实是不准确的。它实际上只是一个“极小”作用原理，因为式（13.130）也可以产生最大化泛函的运动。

13.4.1.3 应用行动

到目前为止都假设没有行动作用在机械系统上。欧拉-拉格朗日方程通常以这种形式出现，因为目标是对运动进行预测，而不是控制它。令 $u\in\mathbb{R}^n$ 表示一个行动向量。行动作为广义力应用于拉格朗日表述中，广义力“作用”在欧拉-拉格朗日方程的右边。这产生：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}}-\frac{\partial L}{\partial q}=u \tag{13.136}$$

行动迫使系统偏离它的原始行为。在某些实例中，真实的行动可以用其他变量来表达，然后通过变换得到 u（回忆第 13.1.2 节差动机构车辆的行动变量的变换）。在这种情况下，式（13.136）中的 u 可以用 $\phi(u)$ 来代替，ϕ 是某种变换。在这种情况下，u 的维数不一定是 n。

13.4.1.4 产生状态方程的步骤

在一个平滑 n 维流形的坐标邻域上用拉格朗日力学，如下一般步骤可以产生微分模型：

（1）确定系统的自由度以及定义相应的 n 维平滑流形 $\mathcal{C}$。

（2）将动能表示成位形速度分量中的一个二次型：

$$K(q,\dot{q})=\frac{1}{2}\dot{q}^T M(q)\dot{q}=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}m_{ij}(q)\dot{q}_i\dot{q}_j \tag{13.137}$$

（3）表示势能 $V(q)$。

（4）令 $L(q,\dot{q})=K(q,\dot{q})-V(q)$ 为拉格朗日函数，并用欧拉-拉格朗日方程（13.130）来确定微分约束。

（5）令 $x=(q,\dot{q})$，将约束转换到相空间。如果可能，求解 $\dot{x}$ 得到 $\dot{x}=f(x,u)$。

例 13.12（再论二维刚体） 式（13.109）中的方程也可以由欧拉-拉格朗日方程产生。令 $\mathcal{C}=\mathbb{R}^2\times\mathbb{S}^1$ 和 $(q_1,q_2,q_3)=(x,y,\theta)$ 以使符号与拉格朗日函数中所使用的一致。

动能是线速度和角速度动能之和，即：

$$K(q,\dot{q})=\frac{1}{2}m\dot{q}\cdot\dot{q}+\frac{1}{2}I\dot{q}_3^2 \tag{13.138}$$

其中 m 和 I 分别表示质量和转动惯量。假设没有重力；因此 $V(q)=0$ 和 $L(q,\dot{q})=K(q,\dot{q})$。

假设广义力 u_1、u_2 和 u_3 应用于位形变量。对 $L(q,\dot{q})$ 应用欧拉-拉格朗日方程后产生：

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial\dot{q}_1}-\frac{\partial L}{\partial q_1}&=\frac{\mathrm{d}}{\mathrm{d}t}(m\dot{q}_1)=m\ddot{q}_1=u_1\\
\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial\dot{q}_2}-\frac{\partial L}{\partial q_2}&=\frac{\mathrm{d}}{\mathrm{d}t}(m\dot{q}_2)=m\ddot{q}_2=u_2\\
\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial\dot{q}_3}-\frac{\partial L}{\partial q_3}&=\frac{\mathrm{d}}{\mathrm{d}t}(I\dot{q}_3)=I\ddot{q}_3=u_3
\end{aligned} \tag{13.139}$$

它们等价于式（13.109）所给出的方程。第一个不同之处在于需要向相空间转换。第二个不同之处在于式（13.139）中的行动变量并不是指力和力矩，而是作用在位形变量上的广义力。如果需要式（13.109）中的原始行动，那么应该进行转换。 ■

13.4.2 一般拉格朗日表达式

当考虑更加复杂的力学问题时，用一般形式表达微分约束会更方便。例如为物体运动链计算式（13.130）就会导致非常复杂的表达式。但是这些表达式的项可以表示成更加简单的，并赋予分量以某些直观含义的标准形式。

假设动能用式（13.126）表示，并令 $m_{ij}(q)$ 表示 $M(q)$ 的元素。假设势能为 $V(q)$。对式（13.136）求导，那么欧拉-拉格朗日方程可以表示为 n 个标量方程[858]：

$$\sum_{j=1}^{n}m_{ij}(q)\ddot{q}_j+\sum_{j=1}^{n}\sum_{k=1}^{n}h_{ijk}(q)\dot{q}_j\dot{q}_k+g_i(q)=u_i \tag{13.140}$$

其中：

$$h_{ijk}=\frac{\partial m_{ij}}{\partial q_k}-\frac{1}{2}\frac{\partial m_{jk}}{\partial q_i} \tag{13.141}$$

每个 i 对应一个方程。式（13.140）的分量有其物理含义。m_{ii} 系数表示关于 q_i 的惯量。m_{ij} 表示加速 q_i 对 q_j 的影响。$h_{ijj}(q)\dot{q}_j^2$ 表示由 q_j 的速度所产生的对 q_i 的离心效应。$h_{ijk}(q)\dot{q}_j\dot{q}_k$ 表示由 q_j 和 q_k 的速度所引起的对 q_i 的科氏效应。g_i 通常由重力产生。

式（13.140）也可以以矩阵的形式给出。那么欧拉-拉格朗日方程表示为：

$$M(q)\ddot{q}+C(q,\dot{q})\dot{q}+g(q)=u \tag{13.142}$$

这代表 n 个标量方程。其中 $C(q,\dot{q})$ 是一个 $n\times n$ 的科氏矩阵。一些科氏矩阵可能会产生多个等价的约束。关于式（13.140），必须选择科氏矩阵使得：

$$\sum_{j=1}^{n}c_{ij}\dot{q}_j=\sum_{j=1}^{n}\sum_{k=1}^{n}h_{ijk}(q)\dot{q}_j\dot{q}_k \tag{13.143}$$

使用式（13.141），那么：

$$\sum_{j=1}^{n}c_{ij}\dot{q}_j=\sum_{j=1}^{n}\sum_{k=1}^{n}\left(\frac{\partial m_{ij}}{\partial q_k}-\frac{1}{2}\frac{\partial m_{jk}}{\partial q_i}\right)\dot{q}_j\dot{q}_k \tag{13.144}$$

确定 $C(q,\dot{q})$ 的标准方法是通过计算 Christoffel 符号。从式（13.144）中的嵌套和式的内部减去 $\frac{1}{2}\frac{\partial m_{jk}}{\partial q_i}$，方程可以重新记为：

$$\sum_{j=1}^{n}c_{ij}\dot{q}_j=\frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial m_{ij}}{\partial q_k}\dot{q}_j\dot{q}_k+\frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(\frac{\partial m_{ij}}{\partial q_k}-\frac{\partial m_{jk}}{\partial q_i}\right)\dot{q}_j\dot{q}_k \tag{13.145}$$

这样 $C(q,\dot{q})$ 的元素可以记为：

$$c_{ij}=\sum_{k=1}^{n}c_{ijk}\dot{q}_k \tag{13.146}$$

其中

$$c_{ijk}=\frac{1}{2}\left(\frac{\partial m_{ij}}{\partial q_k}+\frac{\partial m_{ik}}{\partial q_j}-\frac{\partial m_{jk}}{\partial q_i}\right) \tag{13.147}$$

这称为 Christoffel 符号，它从式（13.145）中得到。注意到 $c_{ijk}=c_{ikj}$。Christoffel 符号是在微分几何中研究仿射联络时产生的，它通常表示为 Γ^i_{jk}。仿射联络提供了一种不用坐标来表达加速度的方法，这种方法与第 8.3.2 节不用坐标来表达切空间的方法是一样的。对于微分几何中的仿射联络，参见文献[135]；它们在力学中的应用，参见文献[158]。

变换到相空间转移方程

第一步是将方程变换为相空间的形式。引入一个 $2n$ 维的相向量 $x=(q,\dot{q})$。任务是得到 $\dot{x}=f(x,u)$，这表示 $2n$ 个标量方程。前 n 个方程是 $\dot{x}_i=x_{n+i}$，i 从 1 到 n。后 n 个方程是通过求解 $\ddot{q}$ 而得到。

假设用式（13.142）的一般形式，求解 $\ddot{q}$ 就会产生：

$$\ddot{q}=M(q)^{-1}(u-C(q,\dot{q})\dot{q}-g(q)) \tag{13.148}$$

直接代入相变量。对于 i 从 1 到 n，每个 $\ddot{q}_i$ 变成 $\dot{x}_{n+i}$，$M(q)$、$C(q,\dot{q})$ 和 $g(q)$ 都用 x 来表达。这就建立了相空间转移方程。

例 13.13（二连杆操作臂） 图 13.13 显示了一个二连杆操作臂，它有两个旋转关节和两个连杆 $\mathcal{A}_1$ 和 $\mathcal{A}_2$。因此 $\mathcal{C}=\mathbb{S}^1\times\mathbb{S}^1$。令 $q=(\theta_1,\theta_2)$ 表示一个位形。每个关节由电机控制，电

动机产生扭矩 u_i。令 u_1 作用于基座，u_2 作用于 $\mathcal{A}_1$ 和 $\mathcal{A}_2$ 之间的关节。令 d_1 表示 $\mathcal{A}_1$ 的连杆长度。令 ℓ_i 表示从 $\mathcal{A}_i$ 的原点到其质心的距离。对于每个 $\mathcal{A}_i$，令 m_i 和 I_i 分别表示质量和转动惯量。

$\mathcal{A}_1$ 的动能为：

$$K_1(\dot{q})=\frac{1}{2}m_1\ell_1\dot{\theta}_1^2+\frac{1}{2}I_1\dot{\theta}_1^2 \tag{13.149}$$

$\mathcal{A}_1$ 的势能为：

$$V_1(q)=m_1g\ell_1\sin\theta_1 \tag{13.150}$$

$\mathcal{A}_2$ 的动能为：

$$K_2(\dot{q})=\frac{1}{2}p\cdot p+\frac{1}{2}I_2(\dot{\theta}_1+\dot{\theta}_2)^2 \tag{13.151}$$

其中 p 表示 $\mathcal{A}_1$ 质心的位置，它们由式（3.53）给出：

$$\begin{aligned}p_1&=d_1\cos\theta_1+\ell_2\cos\theta_2\\p_2&=d_1\sin\theta_1+\ell_2\sin\theta_2\end{aligned} \tag{13.152}$$

$\mathcal{A}_2$ 的势能为：

$$V_2(q)=m_2g(d_1\sin\theta_1+\ell_2\sin\theta_2) \tag{13.153}$$

这样，拉格朗日函数可以表示为：

$$L(q,\dot{q})=K_1(\dot{\theta}_1)+K_2(\dot{\theta}_1,\dot{\theta}_2)-V_1(\theta_1)-V_2(\theta_1,\theta_2) \tag{13.154}$$

将它代入式（13.118）就得到用 $\ddot{q}$、$\dot{q}$ 和 q 表示的、隐含形式的微分约束。对隐含约束求解 $\ddot{q}$ 并使 $x=(q,\dot{q})$ 就可以转换到相空间，其中 x 是一个四维相向量。

不是直接用式（13.118）来进行计算，而是用式（13.140）直接确定约束。第一项：

$$M(q)=\begin{pmatrix}m_{11}&m_{12}\\m_{21}&m_{22}\end{pmatrix} \tag{13.155}$$

其中：

$$\begin{aligned}m_{11}&=I_1+m_1\ell_1^2+I_2+m_2(d_1^2+\ell_2^2+2d_1\ell_2\cos\theta_2)\\m_{12}&=m_{21}=I_2+m_2(\ell_2^2+d_1\ell_2\cos\theta_2)\\m_{22}&=I_2+m_2\ell_2^2\end{aligned} \tag{13.156}$$

以及：

$$\begin{aligned}c_{111}&=\frac{1}{2}\frac{\partial m_{11}}{\partial\theta_1}=0\\c_{112}&=c_{121}=\frac{1}{2}\frac{\partial m_{11}}{\partial\theta_2}=-m_2\ell_1\ell_2p_2\\c_{122}&=\frac{\partial m_{12}}{\partial\theta_2}-\frac{1}{2}\frac{\partial m_{22}}{\partial\theta_1}=-m_2\ell_1\ell_2p_2\\c_{211}&=\frac{\partial m_{21}}{\partial\theta_1}-\frac{1}{2}\frac{\partial m_{11}}{\partial\theta_2}=m_2\ell_1\ell_2p_2\\c_{212}&=c_{221}=\frac{1}{2}\frac{\partial m_{22}}{\partial\theta_1}=0\\c_{221}&=\frac{1}{2}\frac{\partial m_{22}}{\partial\theta_2}=0\end{aligned} \tag{13.157}$$

最后一项定义为：

$$g_1 = (m_1\ell_1 + m_2 d_1)gp_1 + m_1\ell_2 p_2$$
$$g_2 = m_2\ell_2 gp_2 \tag{13.158}$$

动力学也可以用式(13.142)中的 $M(q)$、$C(q,\dot{q})$ 和 $g(q)$ 来表示。科氏矩阵用式(13.143)来定义：

$$C(q,\dot{q}) = -m_2\ell_1\ell_2 p_2 \begin{pmatrix} \dot{\theta}_2 & \dot{\theta}_1 + \dot{\theta}_2 \\ \dot{\theta}_1 & 0 \end{pmatrix} \tag{13.159}$$

其中 p_2 在式（13.152）中定义，它是 q 的函数。为方便起见，令：

$$r = m_2\ell_1\ell_2 p_2 \tag{13.160}$$

那么现在所产生的表达式是式（13.142）的特殊形式：

$$m_{11}\ddot{\theta}_1 + m_{12}\ddot{\theta}_2 - 2r\dot{\theta}_1\dot{\theta}_2 - r\dot{\theta}_2^2 + g_1(q) = u_1$$
$$m_{22}\ddot{\theta}_1 + m_{21}\ddot{\theta}_2 + r\dot{\theta}_1^2 + g_2(q) = u_2 \tag{13.161}$$

令 $x = (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$，并将状态变量代入式（13.161）就得到相空间转移方程。变量 $\ddot{\theta}_1$ 和 $\ddot{\theta}_2$ 分别变成 $\dot{x}_3$ 和 $\dot{x}_4$。方程必须求解 $\dot{x}_3$ 和 $\dot{x}_4$。在文献[858]中将该模型扩展到有传动比的发动机以及质量不可忽略的情况。■

这里给出的例子并没有深刻触及到可能完美建模的系统。一些关于机器人的教科书所介绍的内容涉及更多连杆、不同类型的关节以及摩擦力[368,728,858,908,994]。

物体链的相空间转移方程也能够用牛顿-欧拉力学表述来产生。即使拉格朗日形式更加完美，但是当用递推方式表达时，牛顿-欧拉方程对模拟多体动力系统来说更有效[368,865,994]。

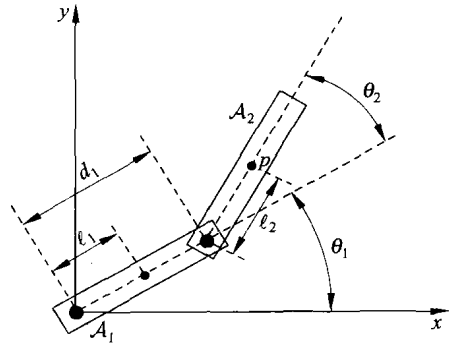


图 13.13　一个有两个旋转关节的二连杆机器人的参数值

13.4.3　欧拉-拉格朗日方程的扩展

可以构造欧拉-拉格朗日方程的多个扩展形式来处理在保守场中出现的除了动能和势能以外的复杂性。每种扩展通常都涉及给式（13.129）加入更多的项来考虑新的复杂性。用这种方法可以进行处理的问题有闭运动链、非完整约束以及非保守力（如摩擦力）。

13.4.3.1　加入速度约束

第 13.4.1 节的拉格朗日表述可以扩展到允许对 q 和 $\dot{q}$ 施加额外的约束。这对建立有闭运动链或者轮式结构的机器人的状态方程来说非常有用。如果有闭链，那么位形可以限制在 C 的一个子集中。如果解集可能用参数表示，那么 C 可以在缩减后的 C-空间上重新进行定义。但是这通常是不可能的，因为这种参数表示很难得到，就像第 4.4 节所提及的那样。如果有第 13.1.3 节那样的轮式结构或者其他接触约束，那么对 q 和 $\dot{q}$ 存在额外约束。通过扩展欧拉-拉格朗日方程，动力学可以加入到第 13.1 节的模型中。

即将介绍的方法都基于拉格朗日乘子。从标准微积分知道：为了最优化一个定义在 $\mathbb{R}^n$

上，并受到 $g(x)=0$ 约束的一个函数 h，只考虑如下表达式的极值就足够了：

$$h(x)+\lambda g(x) \tag{13.162}$$

其中 $\lambda \in \mathbb{R}$ 表示一个拉格朗日乘子[511]。求解：

$$\nabla h(x)+\lambda \nabla g(x)=0 \tag{13.163}$$

得到极值，它表示 n 个方程：

$$\frac{\partial h}{\partial x_i}+\lambda \frac{\partial g}{\partial x_i}=0 \tag{13.164}$$

应用相同的原理可以处理 C 上的速度约束。

假设在 C 上对速度有第 13.1 节所考虑的约束。考虑隐含约束，其中有 k 个形式为 $g_i(q,\dot{q})=0$ 的方程，i 从 1 到 k。通过写成：

$$g_i(q,\dot{q})=\dot{q}_i-f_i(q,u)=0 \tag{13.165}$$

参数约束可以作为隐含约束的特殊情况来处理。对于包含行动 u 的任意约束，不会更难解决。每个 u_i 在下面的分析中都当作一个常数。因此行动变量将不出现在表达式中。

像以前一样，假设动力学是时不变的（时变情况参见文献[792]）。从用式（13.130）所定义的 $L(q,\dot{q})$ 开始，令新的指标为：

$$L_c(q,\dot{q},\lambda)=L(q,\dot{q})+\sum_{i=1}^{k}\lambda_i g_i(q,\dot{q}) \tag{13.166}$$

将 L_c 代替式（13.114）中的 L 以后就定义了一个泛函 Φ_c。

Φ_c 的极值由 n 个方程：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L_c}{\partial \dot{q}_i}-\frac{\partial L_c}{\partial q_i}=0 \tag{13.167}$$

和 k 个方程：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L_c}{\partial \dot{\lambda}_i}-\frac{\partial L_c}{\partial \lambda_i}=0 \tag{13.168}$$

给出。它与式（13.124）的相同，只是这里包括 λ。方程（13.168）等价于 $g_i(q,\dot{q})=0$ 约束。每个方程的第一项为零，因为约束中没有出现 $\dot{\lambda}$，那么：

$$\frac{\partial L_c}{\partial \lambda_i}=0 \tag{13.169}$$

这个结果已经从对 L 极值的约束和 $g_i(q,\dot{q})=0$ 中得出了。在式（13.167）中，有 $n+k$ 个未知数，有 n 个方程。用 k 个约束 $g_i(q,\dot{q})=0$ 可以消除 k 个拉格朗日乘子。这对应于在标准的受约束优化中去除拉格朗日乘子[511]。

式（13.167）中的表达式和约束 $g_i(q,\dot{q})$ 可能十分复杂，这就很难确定状态转移方程。文献[792]的第 3.8 节中给出了一般形式。这里考虑一个非常重要的特殊情况。假设约束是第 13.1 节介绍的 Pfaffian：

$$g_i(q,\dot{q})=\sum_{j=1}^{n}g_{ij}(q)\dot{q}_j=0 \tag{13.170}$$

这其中包括轮式车辆的非完整速度约束，也包括 $g_i(q)=0$ 这种建模闭运动链的特殊情况。对时间求微分得到：

$$\frac{\mathrm{d}}{\mathrm{d}t}g_i(q)=\sum_{j=1}^{n}\frac{\partial g_i}{\partial q_j}\dot{q}_j=\sum_{j=1}^{n}g_{ij}(q)\dot{q}_j=0 \tag{13.171}$$

它是 Pfaffian 的形式。这使得在没有对满足闭包约束的 C 的子集进行参数表示的情况下也能够对第 4.4 节所考虑的封闭链的动力学进行表达。从隐含形式开始，要求用微分法将它们转换到 Pfaffian 的形式。

对于 Pfaffian 约束这种情况，式（13.167）简化为：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_i}-\frac{\partial L}{\partial q_i}+\sum_{j=1}^{k}\lambda_j g_{ji}(q)=0 \tag{13.172}$$

如果需要，Pfaffian 约束可以用来消除拉格朗日乘子。g_{ji} 表示第 j 个 Pfaffian 约束的第 i 项。如果需要，一个行动变量 u_i 可以放置在每个约束的右边。

方程（13.172）通常以如下形式出现：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_i}-\frac{\partial L}{\partial q_i}=\sum_{j=1}^{k}\lambda_j g_{ji}(q,\dot{q}) \tag{13.173}$$

它是约束的一个等价表达式，因为可以忽略拉格朗日乘子而不影响极值的存在性。在这种情况下，可以给出基于达朗贝尔方程的解释。式（13.173）右边的表达式可以认为是第 13.4.1 节所提及的行动。这种行动称为力学中的广义力。对需要用来保持约束的反作用力进行积分就可以得到虚功原理。式（13.173）的右边给出这些反作用力。由于力的抵消，所以约束没有做真实的功（如果没有摩擦）。

例 13.14（球体上的一个质点） 假设一个质点在一个单位球体上做没有摩擦、没有重力的运动。令 $(q_1,q_2,q_3)\in\mathbb{R}^3$ 表示质点的位置。拉格朗日函数是动能：

$$L(q,\dot{q})=\frac{1}{2}m(\dot{q}_1^2+\dot{q}_2^2+\dot{q}_3^2) \tag{13.174}$$

其中 m 是质点的质量。为简单起见，假设 $m=2$。

质点必须沿着球体运动，这个约束产生：

$$g_1(q)=q_1^2+q_2^2+q_3^2-1=0 \tag{13.175}$$

通过时间微分可以变成 Pfaffian 的形式，即：

$$2q_1\dot{q}_1+2q_2\dot{q}_2+2q_3\dot{q}_3=0 \tag{13.176}$$

因为 $k=1$，所以只有一个拉格朗日乘子 λ_1。应用式（13.172）产生三个方程：

$$\ddot{q}_i-2q_i\lambda_1=0 \tag{13.177}$$

其中 i 从 1 到 3。解的一般形式为：

$$c_1q_1+c_2q_2+c_3q_3=0 \tag{13.178}$$

其中 c_i 是实值常数，它们可以用质点的初始位置来进行确定。这表示一个经过原点的平面方程。平面与球面的交集是一个大圆。这意味着质点沿着大圆在两个点之间运动。这是球面上的最短路径（测地线）。■

第 13.4.2 节中的一般形式可以扩展到有约束的情况。例如将式（13.142）推广为：

$$M(q)\ddot{q}+C(q,\dot{q})\dot{q}+g(q)+G(q)^T\lambda=u \tag{13.179}$$

其中 G 是一个由所有 Pfaffian 系数 g_{ji} 构成的 $n\times k$ 矩阵。在这种情况下，拉格朗日乘子可以这样进行计算[728]：

$$\lambda = (G(q)M(q)^{-1}G(q)^T)^{-1}G(q)M(q)^{-1}(u - C(q,\dot{q})\dot{q}) \tag{13.180}$$

其中假设 G 是时不变的。

通过进行所需要的微分、定义 $2n$ 个相变量、求解 $\dot{x}$，这样就可以得到相空间方程。该结果是对式（13.148）的推广。

13.4.3.2 非保守力

已经扩展拉格朗日表述方法来处理 C 上的约束，这些约束降低了切空间的维数。表述还可以扩展到允许非保守力。机械系统中最常见而且最重要的例子就是摩擦力。摩擦模型的详细内容参见文献[684]，这里不予介绍。例如当物体接触时就会有摩擦力，如机器人操作臂的关节、在诸如空气或者水等流体中运动的物体等。非保守力可以表示为附加的广义力，用 $B(q,\dot{q})$ 形式的 $n\times 1$ 向量来表示。假设也允许行动向量。那么修改后的欧拉-拉格朗日方程变成：

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = u - B(q,\dot{q}) \tag{13.181}$$

对式（13.142）的一个常见扩展是：

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q,\dot{q}) = u \tag{13.182}$$

其中 $N(q,\dot{q})$ 是将 $g(q)$ 推广到包括非保守力。还可以进一步推广到包括 Pfaffian 约束和拉格朗日乘子：

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q,\dot{q}) + G(q)^T\lambda = u \tag{13.183}$$

拉格朗日乘子变成[728]：

$$\lambda = (G(q)M(q)^{-1}G(q)^T)^{-1}G(q)M(q)^{-1}(u - C(q,\dot{q})\dot{q} - N(q,\dot{q})) \tag{13.184}$$

再次强调，相空间转移方程从 $2n$ 个相变量产生而来，它是对式（13.148）的推广。

13.4.4 哈密顿力学

力学的拉格朗日表述对于确定物体系的状态转移方程来说是最方便的。一旦确定动能和势能，那么剩余的工作就是导数计算和代数运算。哈密顿力学提出另外一种力学表述方法，它与拉格朗日力学的关系非常紧密。该方法不表示 n 维 C-空间上的二阶微分约束，而是表示 $2n$ 维相空间上的一阶约束。这种思路为我们所熟知。这里所考虑的新的相空间是辛流形（symplectic manifold）的一个例子，它有许多重要的性质，如可定向的和维数是偶数的[39]。标准的相向量定义为 $x(q,\dot{q})$；但是这里不用 $\dot{q}$，而是引入 n 个变量并表示为 p。因此在 $(q,\dot{q})$ 和 (q,p) 之间存在变换。p 变量通过相空间上的一个被称为哈密顿函数的特殊函数而与位形变量产生联系。尽管哈密顿函数通常对于确定 $\dot{x} = f(x,u)$ 并没有多大帮助，但是这里介绍它的原因是它可以推广到最优控制问题。这种推广被称为庞特里亚金最小值原理，将在第 15.2.3 节进行介绍。在力学领域，它提供了对能量守恒定律的一般表示，而能量守恒定律有助于证明许多理论结果[39,399]。

用勒让德（Legendre）变换可以得到 $(q,\dot{q})$ 和 (q,p) 之间的关系。考虑有两个变量 $x,y\in\mathbb{R}$ 的一个实值函数 f。它的全微分[511]为：

$$\mathrm{d}f = u\mathrm{d}x + v\mathrm{d}y \tag{13.185}$$

其中：

$$u=\frac{\partial f}{\partial x}\quad 且\quad v=\frac{\partial f}{\partial y} \tag{13.186}$$

考虑构造一个依赖于 du 和 dy，而不依赖于 dx 和 dy 的全微分。令 g 为 u 和 y 的一个函数，它定义为：

$$g(u,y)=ux-f \tag{13.187}$$

那么 g 的全微分为：

$$\mathrm{d}g=x\mathrm{d}u+u\mathrm{d}x-\mathrm{d}f \tag{13.188}$$

用式（13.185）来表示 df，那么：

$$\mathrm{d}g=x\mathrm{d}u-v\mathrm{d}y \tag{13.189}$$

那么变量 x 和 v 现在为：

$$x=\frac{\partial g}{\partial u}\quad 且\quad v=-\frac{\partial g}{\partial y} \tag{13.190}$$

它是对式（13.186）的一种反演。将这种思路扩展到向量形式就得到哈密顿表述。

假设动力学并不依赖于特定时间（扩展到时变动力学并不困难，可参见文献[39,399]）。令 $L(q,\dot{q})$ 为式（13.129）所定义的拉格朗日函数。令 $p\in\mathbb{R}^n$ 表示一个广义动量向量（或者伴随变量），它的目的与式（13.185）中的相同。每个 p_i 定义为：

$$p_i=\frac{\partial L}{\partial \dot{q}_i} \tag{13.191}$$

在一些文献中，用 p 被表示为 λ，因为它也能够解释为一个拉格朗日乘子向量。哈密顿函数（Hamiltonian function）定义为：

$$H(p,q)=p\cdot\dot{q}-L(q,\dot{q})=\sum_{i=1}^{n}p_i\dot{q}_i-L(q,\dot{q}) \tag{13.192}$$

并且可以解释为一个保守系统的总能量[399]。这是对式（13.187）的一种向量扩展，其中 L 和 H 分别代替 f 和 g。p 和 q 分别是 u 和 x 的向量形式。

H 被当作 p 和 q 的函数，那么它的全微分为：

$$\mathrm{d}H=\sum_{i=1}^{n}\frac{\partial H}{\partial p_i}\mathrm{d}p_i+\sum_{i=1}^{n}\frac{\partial H}{\partial q_i}\mathrm{d}q_i \tag{13.193}$$

用式（13.192），dH 可以表示为：

$$\mathrm{d}H=\sum_{i=1}^{n}\dot{q}_i\mathrm{d}p_i+\sum_{i=1}^{n}p_i\mathrm{d}\dot{q}_i-\sum_{i=1}^{n}\frac{\partial L}{\partial \dot{q}_i}\mathrm{d}\dot{q}_i-\sum_{i=1}^{n}\frac{\partial L}{\partial q_i}\mathrm{d}q_i \tag{13.194}$$

用式（13.191）可以去掉所有 d$\dot{q}_i$ 项，得到：

$$\mathrm{d}H=\sum_{i=1}^{n}\dot{q}_i\mathrm{d}p_i-\sum_{i=1}^{n}\frac{\partial L}{\partial q_i}\mathrm{d}q_i \tag{13.195}$$

用式（13.118），那么有：

$$\dot{p}=\frac{\partial L}{\partial q_i} \tag{13.196}$$

这意味着：

$$\mathrm{d}H = \sum_{i=1}^{n} \dot{q}_i \mathrm{d}p_i - \sum_{i=1}^{n} \dot{p}_i \mathrm{d}q_i \tag{13.197}$$

使式（13.197）和式（13.193）相等就产生了 $2n$ 个方程，它们被称为哈密顿方程：

$$\dot{q}_i = \frac{\partial H}{\partial p_i} \qquad \dot{p}_i = \frac{\partial H}{\partial q_i} \tag{13.198}$$

其中 i 从 1 到 n。这些方程与式（13.190）类似。

哈密顿方程等价于欧拉-拉格朗日方程。这两种情况下的极值产生了等价的微分约束。不同之处在于拉格朗日表述用 $(q,\dot{q})$，而哈密顿函数用 (p,q)。哈密顿函数产生一阶偏微分方程。这里假设动力学是时不变的，并且运动发生在保守场中。在这种情况下，$\mathrm{d}H = 0$，这就是总能量守恒。在时变情况下，与哈密顿方程一同出现的还有附加方程 $\partial H / \partial t = -\partial L / \partial t$。哈密顿方程最初的目的是建立理论力学的基本结论，而不是为了确定特定系统的运动。例如哈密顿函数用来建立 Louisville 定理，该定理论述到：相变量向保护体积的方向流动，也就是说哈密顿系统不是渐进稳定的[39]。第 15.1.1 节将介绍渐进稳定。庞特里亚金最小值原理是将哈密顿方程扩展到最优控制理论，我们将在第 15.2.3 节对它进行介绍。

13.5 多决策者

微分模型可以扩展到对多个决策者的相互作用进行建模。这就将表述 10.1 的序贯决策、表述 10.4 的序贯对策都扩展到连续时间情况。可以为这些扩展建立状态方程的微分版本。

13.5.1 微分决策

微分对策是将表述 10.1 扩展到连续时间的情况，为了建立微分对策，假设大自然行动 $\theta(t)$ 从 Θ 中选择。微分模型定义为：

$$\dot{x} = f(x,u,\theta) \tag{13.199}$$

状态空间 X 和行动空间 U 与本章其他地方所使用的方式相同。仅有的不同是状态转移方程。也可能会用到与状态有关的大自然行动空间。

就像在第III部分中所见，可以对大自然进行非确定的或者是概率的建模。在非确定的情况下，式（13.199）等价于一个微分包含（differential inclusion）[54]：

$$\dot{x} \in \{\dot{x}' \,|\, \exists \theta \in \Theta, \dot{x}' = f(x,u,\theta)\} \tag{13.200}$$

用前向投影可以计算 $\dot{x}$ 将来可能的值。可以定义将在第 14.2.1 节介绍的可达集，它用于刻画将来可能的状态随时间的演化过程。这种模型下构造的规划通常采用最坏情况分析方法。

例 13.15（非确定的前向投影） 作为使用式（13.199）的一个简单例子，考虑用第 12.5.1 节的原像规划框架来表达不确定性模型。

在时刻 $t \geq 0$，大自然选择某 $\theta \in \Theta(t)$。状态转移方程为：

$$\dot{x} = x + u + \theta \tag{13.201}$$

图 12.45 所显示的圆锥只是在应用常数 $u \in U$ 下的非确定的前向投影。 ■

在概率情况下，对大自然行动轨迹（例如维纳过程[911]）必须仔细地进行限制。在这种条件下，式（13.199）变成一个随机微分方程。这种情况下的规划变成连续时间随机控制[570]，并且任务是最优化期望代价。

例 13.16（一辆简易汽车和大自然） 本章的任何模型都可以引入不确定性。例如简易汽车（见式（13.15））。假设大自然干扰汽车的驾驶使得它不能准确地知道汽车将要前进的方向。令 $\Theta = [-\theta_{\max}, \theta_{\max}]$，其中 $\theta_{\max} \in (0, \pi/2)$ 表示大自然所引起的转向角误差的最大值。修改简易汽车模型以考虑这种误差：

$$\begin{aligned} \dot{x} &= u_s \cos\theta \\ \dot{y} &= u_s \sin\theta \\ \dot{\theta} &= \frac{u_s}{L} \tan(u_\phi + \gamma) \end{aligned} \tag{13.202}$$

其中 tan 的定义域必须扩展到 $\mathbb{R}$ 或者必须附加其他合适的限制。在每个 t 时，大自然行动[1] $\gamma \in \Theta$ 对真实航向产生扰动使之偏离命令方向 u_ϕ。在非确定的不确定性下，规划算法必须确定汽车偏离命令方向的最大值。指定一个概率密度函数 $p(\gamma)$ 作为概率模型。当随时间进行积分时，式（13.202）在将来的汽车位形上产生概率分布函数[1004]。 ■

类似地，在本章的任何模型中都可以引入考虑大自然影响的参数。某些误差可能是系统性的，它反映的是建模过程中所产生的错误或者简化。这些误差对应于开始时大自然的一个常数行动。在这种情况下，不允许大自然随时间改变其行动。其他误差则对应于噪声，它们产生随时间而不同的大自然行动。

13.5.2 微分对策论

序贯对策理论扩展到连续时间情况就被称为微分对策论（或者动态对策论[60]），它是由 Isaacs 提出的一门学科[480]。也可能有第 9.3 节、第 9.4 节和第 10.5 节中所考虑的变化形式：

（1）有任意数量局中人。

（2）对策可能是零和的或者非零和的。

（3）状态可能是已知的，也可能是未知的。如果状态未知，那么会出现 I-空间，这与第 11.7 节中的相似。

（4）大自然干扰对策。

（5）可以定义不同的平衡点概念，诸如鞍点和 Nash 平衡点。

关于微分对策的完整介绍，参见文献[60]。两个局中人 P_1 和 P_2 参加微分对策，它们各自都有一个连续行动集合。令 U 和 V 分别表示 P_1 和 P_2 的行动空间。状态转移方程定义为：

$$\dot{x} = f(x, u, v) \tag{13.203}$$

其中 x 是状态，$u \in U$，$v \in V$。

线性微分对策是一族重要的对策，因为最优控制中的许多技术都可以用来解决这种对策[60]。

1 符号 γ 用来代替 θ 以避免在这个特定例子中与汽车定向变量 θ 相抵触。

例 13.17（线性微分对策） 线性系统模型（13.37）可以扩展到两个局中人。令 $X=\mathbb{R}^n$ 为一个相空间。令 $U=\mathbb{R}^{m_1}$ 和 $V=\mathbb{R}^{m_2}$ 为行动空间，$m_1, m_2 \leqslant n$。一个线性微分对策表示为：

$$\dot{x}=Ax+Bu+Cv \tag{13.204}$$

其中 A、B 和 C 分别是 $n\times n$、$n\times m_1$、$n\times m_2$ 的定常矩阵。这种对策的解依赖于代价泛函以及所期望的平衡点。对于二次型代价，存在解析解。这些都是对一个决策者情况的扩展，参见第 15.2.2 节和文献[60]。 ■

Isaacs[480]最初的工作包括一些关于追逃微分对策的有趣的例子。其中最著名的一个描述如下。

例 13.18（Homicidal Chauffeur） 在 Homicidal Chauffeur 对策中，追赶者是 Dubins 车，而逃跑者是一个能够向任意方向平移的点机器人。它们都在 $\mathcal{W}=\mathbb{R}^2$ 中。汽车和机器人的速度分别为 s_1 和 s_2。假设 $|s_1|>|s_2|$，这意味着追赶者的速度大于逃跑者的速度。扩展式（13.15）使之包括两个机器人位置的状态变量，方程为：

$$\begin{aligned} \dot{x}_1 &= s_1\cos\theta_1 \qquad & \dot{x}_2 &= s_2\cos v \\ \dot{y}_1 &= s_1\sin\theta_1 & \dot{y}_2 &= s_2\sin v \\ \dot{\theta} &= \frac{s_1}{L}\tan u_\phi \end{aligned} \tag{13.205}$$

状态空间 X 为 $\mathbb{R}^4\times\mathbb{S}^1$，行动空间为 $U=[-\phi_{\max},\phi_{\max}]$ 和 $V=[0,2\pi)$。

任务是确定追赶者是否能够到达与逃跑者相距一个预先指定的距离 ε 的范围内：

$$(x_1-x_2)^2+(y_1-y_2)^2<\varepsilon^2 \tag{13.206}$$

如果发生这种情况，那么追赶者赢；否则逃跑者赢。问题的解依赖于 L、s_1、s_2、ε 和初始状态。即使追赶者运动得更快，逃跑者仍然可能逃掉，因为它没有转弯半径的限制。对于给定的 L、s_1、s_2、ε 的值，可以将状态空间划分成两个区域，它们分别对应于追赶者赢和逃跑者赢[60,480]。为了得到关于如何进行划分的某种直觉，假设西班牙斗牛士为避开一头快速冲过来的牛而进行的运动（发怒的牛的行为就像一辆快速行驶的 Dubins 汽车）。

■

另外一个有趣的追逃对策是一辆汽车试图拦截另外一辆汽车的情况[697]。

例 13.19（两辆汽车的对策） 假设有两辆简易汽车，它们都在 $\mathcal{W}=\mathbb{R}^2$ 中。每个都有式（13.15）给出的转移方程。这个对策的状态转移方程为：

$$\begin{aligned} \dot{x}_1 &= u_s\cos\theta_1 \qquad & \dot{x}_2 &= v_s\cos\theta_2 \\ \dot{y}_1 &= u_s\sin\theta_1 & \dot{y}_2 &= v_s\sin\theta_2 \\ \dot{\theta}_1 &= \frac{u_s}{L_1}\tan u_\phi & \dot{\theta}_2 &= s\frac{v_s}{L_2}\tan v_\phi \end{aligned} \tag{13.207}$$

如果两个局中人都被限制为 Dubins 车，那么追逃对策就变得非常有趣。 ■

进一步阅读

本章综合了大量的资料来源。一些重要的相关学科被省略了。对于在接触和操作中的多体力学，参见文献[684]。没有讨论三维车辆模型，因为它们的 $SO(3)$ 太复杂，参见文献

[436]。对于与模拟动力系统有关的计算问题，参见文献[249,865]。

对于 C-空间上的速度约束，参见文献[599,728]和第 15.3 节～第 15.5 节。对于更多涉及滚动球体的问题，参见文献[530]及其中的参考文献。滚动球的问题有时被称为 Chaplygin 球。文献[732]研究和分析了由球形关节所构造的一个非完整操作臂。文献[635,719]研究了相互接触中的弯曲物体的运动学问题。对于这种情形中的运动规划，参见文献[103,105,225,679]。其他一些有趣的非完整系统包括 snakeboard[476]、滚筒赛车[559]、滚筒推土机[216]、Trikke[215]和文献[114]中的例子（例如 Chaplygin 雪橇）。

相空间是微分方程、物理和控制论的基础，参见文献[44,194]。

进一步关于力学的阅读材料因两种不同的处理而有所复杂。典型的力学书籍并不基于微分几何，因为它产生烦琐的表述和不寻常的术语（例如广义坐标）。通过用黎曼流形上的测地线来清晰地表述事物，现代力学就能够克服该问题；但是对于没有微分几何知识背景的读者来说，这可能更加难以理解。文献[39]是关于现代力学的一个非常好的书籍。著名的经典力学书籍之一是文献[399]。文献[793]是一个介绍变分法的在线图书，它还包括受约束拉格朗日函数。受约束拉格朗日函数是基于文献[792]的第 3 章、文献[399]的第 2.4 节和文献[407]的一部分。除了代数约束和微分约束以外，文献[793]还介绍了关于对拉格朗日函数的积分约束。文献[792]介绍了不等式约束下的拉格朗日力学。第 13.4.4 节中的哈密顿函数的表示是基于文献[399]的第 7 章和文献[39]的第 15 节。对于用仿射联络和 Christoffel 符号所论述的现代力学，见文献[3,158,680]。另外一个得到详细阐述的资料是文献[361]。进一步阅读机器人动力学，见文献[30,206,728,858,908,994]。对于机动车动力学，参见文献[391]。

关于微分对策的进一步资料有文献[60,426,480]；也参见文献[34,58,786,985,991,992,993,997]。追逃微分对策的算法复杂性的下界由文献[824]提出。

习　题

1. 令 $C=\mathbb{R}^4$。有两个 Pfaffian 约束 $\dot{q}_1+\dot{q}_2+\dot{q}_3+\dot{q}_4=0$ 和 $\dot{q}_2-\dot{q}_4=0$。确定行动变量的近似数量，并用 $\dot{q}=f(q,u)$ 的形式表达微分约束。

2. 引入一个相空间，将 $2\ddot{y}-10\dot{y}^2+5y=0$ 转换为 $\dot{x}=f(x)$ 的形式。

3. 引入一个相空间，将 $y^{(4)}+y=0$ 转换为 $\dot{x}=f(x)$ 的形式。

4. 推导牵引拖车的一辆汽车的位形转移方程（13.19）。

5. 用第 13.2.4 节的思路，对牵引拖车的汽车式（13.19）进行平滑操控扩展。

6. 假设两个相同的差动机构机器人的中心用一个长度为 d 的刚性杆连接在一起。机器人在杆的两端用螺旋关节连接。有四个车轮需要控制，但是车轮旋转的某些组合可能会引起侧滑。假设不允许侧滑，设计一个 $\dot{q}=f(q,u)$ 形式的运动模型，其中 C 和 U 反映真实自由度。

7. 将月球车模型扩展到带一个推进器的一般刚体，推进器的推力不经过质心。

8. 为在水平面上方、以一定角度进行点火的一个三维旋转刚体设计一个模型。假设推进器被置于刚体上，使得在它落到地面之前能够对其进行控制。给出一般相空间转移方程。

9. 在例 13.12 中关于 q_2 施加重力，并用欧拉-拉格朗日方程产生新的状态转移方程。

10. 用受约束拉格朗日函数来产生例 13.8 中单摆的运动方程。

11. 定义一个相空间，为如图 13.14 所示的双摆确定一个 $\dot{x} = f(x)$ 形式的方程。

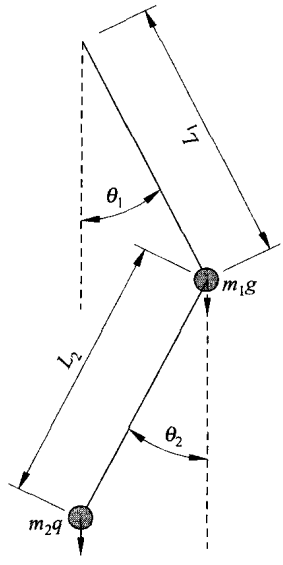


图 13.14 一个双摆

12. 将例 13.13 进行扩展以得到三连杆操作臂的动力学。第三个连杆 A_3 通过一个螺旋关节连接到其他两个连杆。新的参数是 θ_3、d_2、l_3、m_3 和 I_3。

13. 通过将有标准球坐标的球参数化并用无约束拉格朗日函数来解决例 13.14。验证能够得到相同的答案。

14. 将式（13.161）转换为相空间形式，得到 $\dot{x} = f(x,u)$ 形式的相空间转移方程。用诸如质量、转动惯量、长度等这些基本参数来表达方程的右边。

15. 定义在重力影响下自由二维刚体的哈密顿函数，并建立哈密顿方程。

实现

16. 设计一个三维航天器（刚体）模拟器，它有任意数量的二进制推进器，推进器要么开启，要么关闭，这些推进器可以被放置在任意位置和方向上。

17. 为例 13.13 中的二连杆操作臂设计一个模拟器。

第 14 章　微分约束条件下基于采样的规划

在第 13 章以后的内容中，微分约束似乎无处不在。例如当车轮滚动、飞机飞行以及当考虑任何机械系统的动力学时都会出现微分约束。这使得第 II 部分的运动规划所使用基本模型在一些应用中失效，因为它们忽略了微分约束。例如表述 4.1 只关注 C 空间中的障碍物。

本章将第 13 章的微分约束融入基于采样的运动规划中。第 13 章的详细模型（例如拉格朗日力学）在这里并不重要。本章直接采用 $\dot{x} = f(x,u)$ 形式的系统。注意力集中在基于采样的方法，因为如果微分约束存在，组合方法很难有所作为。但是如果没有障碍物，那么可以应用分析技术。这是对有障碍物情况下运动规划的补充，这将在第 15 章介绍。

第 14.1 节提出微分约束下运动规划的基本定义和概念。这对于说明非完整规划、动态规划、轨迹规划之间的不同之处是至关重要的，它们都是微分约束条件下的规划。另外重要的一点是障碍物在相空间中可能会更加复杂，这在第 13.2 节介绍过。第 14.2 节将介绍在行动轨迹的空间上进行的采样，这是后续规划算法的本质所在。

第 14.3 节将再次讨论第 5.4 节的递增采样和搜索架构，并将它们进行扩展以处理微分约束。这引出了将在第 14.4 节介绍的几种基于采样的规划方法。将会再次出现诸如动态规划或者第 5.5 节的 RDT 等我们所熟悉的内容。所产生的规划方法可以应用于涉及 C 空间或者相空间上微分约束的各种问题中。

第 14.5 节简要介绍了微分约束下的反馈运动规划。对第 8.5.2 节值迭代方法进行适应性修改就可以得到近似最优规划。第 14.6 节将描述解耦方法，它从忽略微分约束的无碰撞路径开始，然后进行改进得到所希望的轨迹。这种方法通常会损失完备性和最优性，但是它们在许多情况下能够大量减小计算量。第 14.7 节简要介绍微分约束下最优轨迹的数值计算技术，该技术可以改进规划算法计算的解。

14.1　简　　介

14.1.1　问题表述

可以将微分约束下的运动规划认为是两点边值问题（two-point boundary value problems，BVP）[443]的一种变形。在两点边值问题中，给定初始和目标状态，任务是计算状态空间中一条连接初始状态和目标状态的路径，同时还要满足微分约束。运动规划还涉及在状态空间中避免碰撞障碍物等复杂因素。遗憾的是，解决 BVP 的方法并不完全适用于运动规划，因为它们不是为处理障碍区域而设计的。对一些方法可能需要进行适应性修改，但是障碍物约束通常使得这些经典方法失去有效性或者完备性。在本章中，BVP 被认为是有微分约束、没有障碍物的运动规划。也存在涉及多于两点的 BVP，但是本书不予

考虑。

假设微分约束用平滑流形 X 上的一个状态转移方程 $\dot{x}=f(x,u)$ 表示，该流形被称为状态空间，它可能是一个 C-空间 $\mathcal{C}$ 或者一个 C-空间的一个相空间。路径解不像第 II 部分那样直接表示，而是从行动轨迹通过状态转移方程的积分来产生。

令行动空间 U 为 $\mathbb{R}^m$ 的一个有界子集。一个规划算法计算一个行动轨迹 $\tilde{u}$，它是一个 $\tilde{u}:[0,\infty)\rightarrow U$ 形式的函数。特定时刻 t 的行动表示为 $u(t)$。为了与函数的符号一致，这似乎应该表示为 $\tilde{u}(t)$。但是这里没有这样做，这是有目的的：一是为了与离散情况之间的联系更加清晰；二是为了对行动 $u\in U$ 和行动轨迹 $\tilde{u}$ 加以区别。如果行动空间与状态有关，那么 $u(t)$ 应该满足 $u(t)\in U(x(t))\subseteq U$。对于与状态有关的模型，默认情况下都做这种假设。还假设使用终止行动 u_T，因为知道在某个 t_F 时刻会应用终止行动，所以有可能指定 $[0,\infty)$ 上所有行动轨迹。

需要表述行动和状态轨迹之间的联系。从 $t=0$ 时刻某初始状态 $x(0)$ 开始，从一个行动轨迹 $\tilde{u}$ 产生状态轨迹：

$$x(t)=x(0)+\int_0^t f(x(t'),u(t'))\mathrm{d}t' \qquad (14.1)$$

它是状态转移方程从初始条件 $x(0)$ 进行的积分。令 $\tilde{x}(x(0),\tilde{u})$ 表示式（14.1）积分所得到的状态轨迹。式（14.1）的微分就是状态转移方程。回忆第 13.1.1 节，如果 u 固定，那么状态转移方程则定义了一个向量场。状态转移方程是第 8.3 节中式（8.14）的另外一种表达式，在那里表达的是一个向量场的积分曲线。在目前情况下状态轨迹就是积分曲线。

微分约束条件下的运动规划问题可以表示为对表述 4.1 的钢琴搬运工问题的扩展。主要不同在于：（1）时间；（2）状态或者相空间；（3）状态转移方程。结果如下。

表述 14.1（微分约束下的运动规划）

1. 一个世界 $\mathcal{W}$、一个机器人 $\mathcal{A}$（或者 $\mathcal{A}_1,\cdots,\mathcal{A}_m$ 连杆组）、一个障碍区域 $\mathcal{O}$ 和一个位形空间 $\mathcal{C}$，它们的定义与表述 4.1 相同。

2. 一个无界时间区间 $T=[0,\infty)$。

3. 一个被称为状态空间的平滑流形 X，它可能是 $X=\mathcal{C}$ 或者考虑动态时从 $\mathcal{C}$ 产生的相空间，参见第 13.2 节。令 $\kappa:X\rightarrow\mathcal{C}$ 表示一个函数，它返回与 $x\in X$ 有关的位形 $q\in\mathcal{C}$。这里 $q=\kappa(x)$。

4. 为状态空间定义一个障碍区域 X_{obs}。如果 $X=\mathcal{C}$，那么 $X_{obs}=\mathcal{C}_{obs}$。对于一般相空间，第 14.1.3 节详细描述了 X_{obs}。符号 $X_{free}=X\setminus X_{obs}$ 表示能够避免碰撞和满足任意附加全局约束的状态。

5. 对于每个状态 $x\in X$，一个有界行动空间 $U(x)\subseteq\mathbb{R}^m\cup\{u_T\}$，它包括一个终止行动 u_T，m 是某固定的整数，它被称为行动变量数。如果应用终止行动，那么假设 $f(x,u_T)=0$（如果使用代价泛函，那么没有代价积累）。令 U 表示在所有 $x\in X$ 上 $U(x)$ 的并集。

6. 用为每个 $x\in X$ 和 $u\in U(x)$ 定义的状态转移方程 $\dot{x}=f(x,u)$ 来指定一个系统。这可以从第 13 章的任意微分模型中产生。

7. 将一个状态 $x_I\in X_{free}$ 指定为初始状态。

8. 将一个状态 $x_G\in X_{free}$ 指定为目标状态。

9. 一个完整的算法必须计算出一个行动轨迹 $\tilde{u}: T \rightarrow U$，对于该行动轨迹来说，从式（14.1）产生的状态轨迹满足：（1）$x(0) = x_I$；（2）存在某 $t > 0$，有 $u(t) = u_T$，$x(t) \in x_G$。

对 $\tilde{u}$ 可能还有附加约束，如连续性或者平滑性等。至少选择的 $\tilde{u}$ 必须使得式（14.1）的被积函数是可积的。令 $\mathcal{U}$ 表示 $T = [0, \infty)$ 上所有可允许行动轨迹的集合。在默认情况下，假设 $\mathcal{U}$ 包括任意可积行动轨迹。如果需要，还可以通过引入新的相变量来保证连续性和平滑性。第 13.2.4 节介绍的在行动变量前面放置积分器的方法通常可以实现想要的约束。如果还需要对指标进行优化，那么可以使用式（8.39）给出的代价泛函。最优解的存在性要求 U 除了是有界的以外，还必须是闭集合。

由于有终止行动 u_T，所以最后时刻不需要进行论述。通常一旦应用 u_T，代价就不再进一步积累，并且状态保持固定。对于涉及动力学的问题，这似乎有点奇怪，因为动量应该保持运动状态。需要记住的是终止行动只是使得表述正常工作所使用的一个技巧。在许多情况下，目标对应于 X 的一个速度分量为零的子集。在这种情况下就没有动量，也就不会有问题。如果目标区域包括有非零速度的状态，那么物理系统在应用 u_T 以后应该仍然保持运动，但是代价泛函将不会测量到任何进一步的代价。在应用 u_T 以后，系统就认为任务完成了。如果机械系统由于动量而发生碰撞，那么这是用户的问题，因为他指定的目标状态涉及动量。

绝大多数求解技术都是基于采样的方法。这是由于在微分约束下进行规划极其困难。来自于表述 4.1 的标准钢琴搬运工问题是表述 14.1 的特殊情况，它是 PSPACE-复杂度的[820]。甚至对于在没有微分约束的三维多面体环境下的一个点来说，最优规划也是 NP-复杂度的[174]。对于 $\mathcal{C} = \mathbb{R}$ [750]和 $\mathcal{C} = \mathbb{R}^2$ [173]，在出现障碍物、有微分约束情况下进行规划的方法只是为二重积分器 $\ddot{q} = u$。

第 14.1.2 节对在表述 14.1 所描述的微分约束下运动规划问题的前景提出了一些展望，它假设给定初始状态并且未来的状态是可以预测的。第 14.5 节简要讨论了微分约束下反馈运动规划问题。

14.1.2　不同类型的规划问题

有许多方法来对微分约束下的运动规划问题进行分类。一些规划方法依赖于系统的特性，因此分类有利于刻画这些系统之间的差别。这里所描述的不同类型的问题都是表述 14.1 的具体化。尽管下面所描述的模型类型不同，但是它们可以统一到微分约束条件下的规划这个课题中。

影响微分模型的一个因素是任务分解的方法。例如使机器人移动这个任务通常要求考虑力学问题。在图 1.19 所示的经典机器人学方法下，运动规划问题从机器人力学中被抽取出来。这使得可以应用第 II 部分的运动规划的思路。这种分解具有随意性。机器人力学可以直接在规划过程中予以考虑。另外一种方法是只考虑部分约束。例如在规划过程中只考虑运载器的滚动约束，而动力学用另外规划模块来处理。因此重要的是记住：出现在规划过程中的微分约束的类型不只依赖于特定的机械系统，而且还依赖于任务是如何分解的。

14.1.2.1 规划文献中的术语

非完整规划 “非完整规划”这个术语由 Laumond[596]提出用来描述轮式机器人运动规划问题（见文献[598,636]）。在第 13.1 节对该术语进行了非正式介绍，在那里非完整性是指不能完全可积的微分约束。这意味着它们不能转换为不涉及导数的约束。非完整性的一个更加正式的定义将在第 15.4 节给出。大多数规划方面的研究都集中在对 C 上的速度进行约束，而不是对相空间 X 上的速度进行约束。这包括第 13.1 节给出的大多数模型，给它们施加了 C-空间 C 上不可积速度约束。这些约束通常被称为运动约束，以区别于由动力学产生的约束。

在力学和控制领域，非完整性也应用于相空间上不可积速度约束[114,115]。因此非完整性规划这个术语也指那些涉及动力学的问题。但是目前在大多数应用中，非完整性规划指的都是只有运动约束这种问题。这是受到早期的轮式机器人规划的启发。本书中的非完整性规划指的是在 C 上或者相空间 X 上不可积速度约束下的规划。

对于基于采样的规划，完全可积实际上并不重要。在许多情况下，即使理论上可以建立可积约束，也并不意味着实际上能够进行积分，并且即使能够进行积分，约束也可能是隐含的，所以并不是轻易可参数化的。例如假设约束从闭运动链产生。通常无法进行参数化。将闭包约束进行微分，在 C 上得到速度约束。在基于采样的规划器中可以这样处理，就好像它是一个非完整性约束，即使它很容易进行积分。

Kinodynamics 规划 Kinodynamics 规划是由 Canny、Donald、Reif 和 Xavier[292]提出的，它指的是必须满足速度和加速度限制的运动规划问题。这意味着在 C 上有二阶约束。对于 $C=\mathbb{R}^2$ 和 $C=\mathbb{R}^3$，最初的工作采用二重积分器模型 $\ddot{q}=u$。例 13.3 是该模型的标量版本。最近该术语被许多作者应用于涉及动力学的运动规划问题中。因此涉及二阶（或者更高阶）微分约束的任何问题都可以被认为是一种形式的 Kinodynamics 规划。因此如果 x 包括速度变量，那么 Kinodynamics 规划包括任何系统 $\dot{x}=f(x,u)$。

注意到 Kinodynamics 规划不一定是一种形式的非完整性规划，而且目前所考虑的大多数情况都不是。问题可能会涉及非完整性规划和 Kinodynamics 规划。这就要求微分约束不只是不可积的，并且至少是二阶的。这种情形通常产生于第 13.4.3 节介绍的受限拉格朗日分析。第 13.3.3 节的汽车既是 Kinodynamics 的，又是非完整性的。

轨迹规划 轨迹规划这个术语在机器人领域已经用了数十年，它指的是确定机械臂（如 PUMA 560）的路径和速度函数的问题。即在相空间 X 中寻找一条路径，其中 $x\in X$ 定义为 $x=(q,\dot{q})$。通常使用第 1.4 节提到的改进方法来解决问题，改进方法首先计算 C_{free} 中的一条路径。对于沿着路径的每个位形 q 来说，速度 $\dot{q}$ 必须满足微分约束。可能也存在逆控制问题，它涉及计算每个时刻 t 导致想要的 $\dot{q}(t)$ 的行动 $u(t)$。改进方法通常作为 C 中一条路径的时标[459]。近年来，轨迹规划和 Kinodynamics 规划似乎是同义的，即约束都是二阶的（x 只包括位形和速度变量）。一个区别在于轨迹规划仍然保持方法的历史内涵，即首先规划 C_{free} 的一条路径。

14.1.2.2 控制论中的术语

大量的适合规划的术语是在控制领域发展起来的。在某些情况下，可能与规划中的术语发生冲突。例如“运动规划”已经用来表示没有障碍物的情况下的非完整性规划[158,730]。

这可以认为是一种 BVP。在某些情况下，这种形式的规划指的是操控问题（见文献[599,728]），将在第 15.5 节进行介绍。对于涉及避开障碍物以及其他约束的问题，本书中的运动规划这个术语依然保持其原来的含义。

开环控制律　诸如第 13 章中的那些微分模型通常都被称为控制系统或者系统，这是我们已经使用过的概念。它们分为线性系统和非线性系统，这分别在第 13.2.2 节和第 13.2.3 节介绍过。表述 14.1 在控制论中可以认为是为系统设计一个开环控制律（受状态空间上非凸约束）。开环是指没有使用反馈。只需要确定随时间变化的行动轨迹（反馈情况称为是闭环的，回忆第 8.1 节）。一旦给定初始状态，状态轨迹可以从行动轨迹中产生。它也可以作为可行开环控制律，因为它满足所有约束，但不一定是最优的。考虑设计一个最优开环控制律是有趣的。甚至对于看起来非常简单的问题来说，这也是极其具有挑战性的。对于某些受限系统，这其中包括线性系统和一些轮式车辆模型，会存在非常良好的解，但都是在没有障碍物的情况下。这将在第 15 章进行介绍。

漂移　漂移的概念曾经出现第 13.2.1 节，它意味着从某些状态不可能瞬间停止下来。这种困难出现在有动量的机械系统中。要求用无穷大的加速度，即需要无穷大能量来在瞬间内抵消一个机械系统的所有动能。Kinodynamics 规划和轨迹规划一般情况下会涉及漂移。非完整性规划问题可能是无漂移的，如果在 $\mathcal{C}$-空间上只存在速度约束。第 13.1.2 节的模型是无漂移的。从规划的角度来看，有漂移的系统通常比无漂移的系统更难以处理。

欠驱动　作为 u 的分量，行动变量通常被称为执行机构（actuators），如果执行机构的数量严格小于 $\mathcal{C}$ 的维数，那么称系统是欠驱动的（underactuated）。也就是说，独立的行动变量少于机械系统的自由度。欠驱动非线性系统是典型的非完整系统。因此大量的非完整性系统理论和规划都涉及欠驱动系统的应用问题。作为欠驱动系统的一个例子，假设有一个在 $\mathbb{R}^3$ 中自由漂浮的航天器，它有三个推进器。每个推进器所产生的力都是一个行动变量；但是系统是欠驱动的，因为只有三个推进器，而 $\mathcal{C}$ 的维数为六。其他例子出现在第 13.1.2 节。如果系统不是欠驱动的，那我们称之为全驱动的（fully actuated），即执行机构的数量等于 $\mathcal{C}$ 的维数。Kinodynamics 规划讨论的几乎都是全驱动系统。

对称系统　最后，在一些规划算法中一个非常重要的系统特征是对称性[1]。如果如下条件成立，那么一个系统 $\dot{x} = f(x,u)$ 是对称的。如果存在一个行动轨迹使得系统从某 x_I 到达 x_G，那么也存在另外一个行动轨迹，它通过访问 X 中相同的点使得系统从 x_G 到达 x_I，但是时间过程相反。在沿着路径的每个点上，这意味着不同的行动可以使速度取反。因此对于一个对称系统来说有可能翻转任何运动。对有漂移的系统来说这通常是不可能的。一个对称系统的例子是第 13.1.2 节的差动机构。对于简单汽车来说，Reed-Shepp 车是对称的，但是 Dubins 车不是，因为它不能倒车。

14.1.3　相空间中的障碍物

现在详细论述表述 14.1 中第 4 项的障碍区域。如果 $X = \mathcal{C}$，那么 $X_{obs} = \mathcal{C}_{obs}$，对于刚体机器人，它是在式（4.34）中定义的；对于多连杆机器人，它是在式（4.36）中定义的。如

1　有时在控制论中，将“对称性”这个术语应用于李群。这是一个不同的概念，它意味着系统关于诸如 $SE(3)$这样的群中的变换是不变的。例如一辆汽车的动力学应该不依赖于汽车所面向的方向。

果 X 是一个不仅包括位形信息而且还包括速度变量的相空间，那么会发生更有趣的情况。

相关位形落在 $\mathcal{C}_{obs}$ 中的任何状态也必须是 X_{obs} 的成员。如果在 $\mathcal{W}$ 中发生碰撞，那么速度是不相关的。在大多数涉及相空间的情况中，障碍区域 X_{obs} 被定义为：

$$X_{obs} = \{x \in X | \kappa(x) \in \mathcal{C}_{obs}\} \qquad (14.2)$$

其中 $\kappa(x)$ 是与状态 $x \in X$ 有关的位形。如果 X 的前 n 个变量是位形参数，那么 X_{obs} 关于其他变量有圆柱形结构，如图 14.1 所示。如果 κ 是一个复杂映射，而不是简单地选择位形坐标，那么结构可能不是圆柱的。在这些情况下，式（14.2）仍然能够表明 X 中正确的障碍区域。

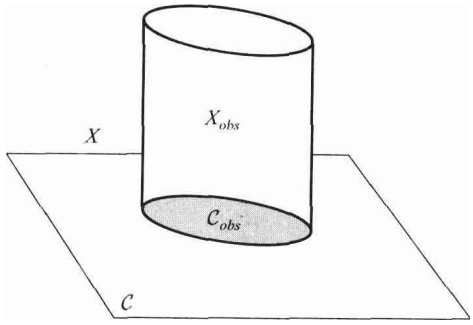
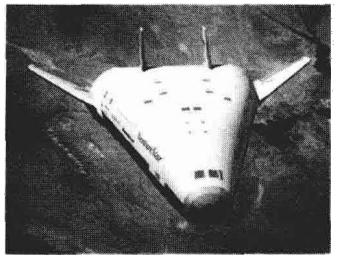


图 14.1 一个障碍区域 $\mathcal{C}_{obs} \subset \mathcal{C}$ 产生关于相变量的一个圆柱障碍区域 $X_{obs} \subset X$

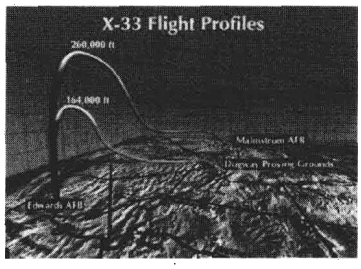
14.1.3.1 关于相变量的附加约束

在许多应用中，对相变量可能存在附加约束。这些约束被称为相约束，它们的一般形式是 $h_i(x) \leq 0$。例如基于安全原因，一辆车或者气垫船可能有一个最大速度。因此对速度变量存在简单的界限。例如可指定 $\|\dot{q}\| \leq \dot{q}_{\max}$，$\dot{q}_{\max}$ 为某常数 $\dot{q}_{\max} \in (0, \infty)$。这种简单的界限通过对速度变量进行限制而直接被合并到 X 的定义中。

但是在其他情况下，对速度的约束可能十分复杂。例如计算 NASA/洛克希德马丁的 X-33 可重复使用航天器[1]（见图 14.2）的再入轨迹问题时要求保持在相空间中的一个复杂的狭窄区域内。即使没有传统意义上的障碍物，但是进入错误的相空间时也会发生很糟糕的情况。例如航天器可能过热或者不可控地振动[162,203,665]。有一个更简单的例子，假设对 X 进行约束以保证一个 SUV 或者一个双层观光巴士在转弯时不会翻倒。



NASA/洛克希德 -马丁公司的 X-33



再入轨迹

图 14.2 在 NASA/洛克希德-马丁公司的 X-33 再入问题中，对相变量有复杂的约束，它避开引起飞行器过热或者不可控地振动的状态（由 NASA 提供）

附加约束可以隐含表示为 $h_i(x) \leq 0$。确定某状态是否在 X_{free} 或者 X_{obs} 中，必须将它代入每个约束中来确定是否满足附加约束。如果一个状态在 X_{free} 中，那么一般情况下称它为无违规的（violation-free），也就是说它既不会发生碰撞，也没有违背任何附加相约束。

1 该工程在 2001 年被取消，但是相似的飞行器正在研制中。

14.1.3.2　不可避免的碰撞区域

规划的最具挑战性的一个方面可以用不可避免的碰撞区域 X_{ric} 来形象描述。这是最终进入 X_{obs} 的那些状态的集合，无论采取什么行动。有一个简单的例子，假设一个机器人以 100km/h 的速度向 2m 外的一堵墙驶去。由于有动量，那么一定会发生碰撞，无论采取什么行动试图使之停止或者转弯。在足够低的速度下，X_{ric} 和 X_{obs} 近似相同，但是在速度增加的情况下 X_{ric} 会显著增大。

令 U_∞ 表示从来不用终止行动 u_T 的所有轨迹 $\tilde{u}:[0,\infty)\rightarrow U$ 的集合（我们不认为用 u_T 就可以避免那些不可避免的碰撞）。不可避免的碰撞区域被定义为：

$$X_{ric}=\left\{x(0)\in X \,\middle|\, \tilde{u}\in \mathrm{U}_\infty, \exists t>0, x(t)\in X_{obs}\right\} \tag{14.3}$$

其中 $x(t)$ 是从 $x(0)$ 应用式（14.1）所得到的 t 时刻的状态。这不包括运动最终受到阻挡这种情况，但是它可能将系统带入零速状态。假设用第 13.1.2 节的 Dubins 车，并且车不能在小路尽头以外的地方调头。在这种情况下，它可以通过停车来避免发生碰撞。如果继续行驶，那么它最终必然会发生碰撞。这种情况就好像落入一个陷阱中一样，所以 X_{ric} 的定义不适于这种情况。对于无漂移系统，$X_{ric}=X_{obs}$。

例 14.1（不可避免的碰撞区域） 图 14.3 显示了对 X_{ric} 的一个简单阐述。假设 $\mathcal{W}=\mathbb{R}$，并且机器人是一个按照二重积分模型 $\ddot{q}=u$ 运动的质点（假设质量 $m=1$）。为简单起见，假设 u 表示必须从 $U=[-1,1]$ 中选择的一个力。$\mathcal{C}$-空间为 $\mathcal{C}=\mathbb{R}$，相空间为 $X=\mathbb{R}^2$，相变量（状态）表示为 $x=(q,\dot{q})$。假设在 $\mathcal{C}$ 上有两个障碍物：一个点和一个区间。它们沿着 q 轴，如图 14.3 所示。在它们上面的圆柱体中出现了 X_{obs}。在 $\dot{q}=0$ 处，$X_{ric}=X_{obs}=\mathcal{C}_{obs}$。当 $\dot{q}$ 增加时，即使 X_{obs} 保持固定，X_{ric} 也会增大。注意到 X_{ric} 只向左发展，因为 $\dot{q}>0$ 表示一个正速度，即在正 q 方向上产生动量。当动量增加时，所需要的停车距离呈二次方增加。对于速度 $\dot{q}=v$，最小停车距离为 $v^2/2$，这可以用行动 $u=-1$、对 $\ddot{q}=u$ 积分两次得到。如果 $\dot{q}>0$ 以及 q 在障碍物的右边，那么它将能够安全地避开障碍物，无论速度多大。如果 $\dot{q}<0$，那么 X_{ric} 向右延伸。这还是由于所需要的停车距离所造成的。 ■

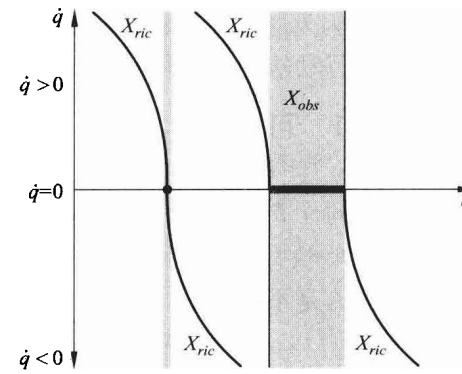


图 14.3　不可避免发生碰撞的区域随速度呈二次方增长

在高维和更一般的系统中，问题会变得更加复杂。例如在 $\mathbb{R}^2$ 中机器人可以通过转向来避开小的障碍物。一般情况下，运动的特定方向十分重要，并且 X_{ric} 的拓扑与 X_{obs} 的拓扑可能十分不同。假设一个小型飞机飞进一个由复杂的走廊网络组成的洞穴中。一旦飞机进入这个洞穴，那么就没有什么行动能够避免发生碰撞。在洞穴中飞机的整个状态空间都包含在 X_{ric} 中，并且还必须包括不能避免使飞机进洞的那些状态。

在微分约束下的采样规划中，不计算 X_{ric}，因为它过于复杂[1]。甚至不知道如何对 X_{ric} 构

1　但是有可能计算 X_{ric} 的粗略近似，并将它们用在规划中。

造“碰撞检测器”。如果用 X_{obs}，那么由于有动量而会出现困难。状态空间的绝大部分可能都不值得探测，因为它们都在 X_{ric} 中。遗憾的是，目前还没有比较实用的方法来准确地确定状态是否在 X_{ric} 中。当动量和杂乱程度增加时，这将越来越成问题。

14.2 可达性与完备性

本节介绍基于采样的规划算法的基本概念。在第 5 章中，在 C 上进行采样是非常重要的。最重要的是采样序列应该是稠密的（dense）以使采样点接近于 C_{free} 中的任意点。行动轨迹使得微分约束下规划的解更加复杂。为了使基于采样的规划算法是分辨率完备的，在行动轨迹的空间上进行的采样和搜索必须以某种方式产生 X_{free} 中的一个稠密集。

14.2.1 可达集合

对于第 5 章的算法，分辨率完备性和概率完备性依赖于 C 上的一个稠密采样序列。在目前情形下，这要求在 X 上进行稠密采样。但是微分约束使得采样过程异常复杂。一般情况下，在 X 上规定必须到达的采样点是不合理的，因为到达它们也许是不可能的或者需要求解 BVP。因为通过行动轨迹间接得到 X 中的路径，所以完备性分析开始于考虑哪些点是可以通过对行动轨迹进行积分而到达的。

14.2.1.1 可达集合

暂时假设没有障碍物：$X_{free}=X$。令 $\mathcal{U}$ 为在时间区间 $[0,\infty)$ 上所有可允许的行动轨迹的集合。从每个 $\tilde{u}\in\mathcal{U}$，用式（14.1）来定义一个状态轨迹 $\tilde{x}(x_0,\tilde{u})$。那么这些轨迹能够访问 X 中的哪些状态？X 中的所有状态都可能被访问到，但在一般情况下，由于有微分约束，某些状态也许是不可达的。

令 $R(x_0,\mathcal{U})\subseteq X$ 表示 x_0 的可达集，它是从 x_0 开始的任意轨迹都能够访问到的，从某 $\tilde{u}\in\mathcal{U}$ 积分所得到的所有状态的集合。这可以正式地表示为：

$$R(x_0,\mathcal{U})=\left\{x_1\in X \middle| \exists\tilde{u}\in\mathcal{U},\exists t\in[0,\infty), x(t)=x_1\right\} \tag{14.4}$$

其中 $x(t)$ 由式（14.1）给出，并要求 $x(0)=x_0$。

以下几个例子阐述了某些简单情况。

例 14.2（简单不等式约束的可达集） 假设 $X=C=\mathbb{R}^2$，回忆第 13.1.1 节的某些简单约束。令 $\mathbb{R}^2$ 中的一个点为 $q=(x,y)$。令状态方程 $\dot{x}=u_1$ 和 $\dot{y}=u_2$，其中 $(u_1,u_2)\in U=\mathbb{R}^2$。

对 U 施加不同约束就可以定义不同的行动空间。假设要求 $u_1>0$（这是第 13.1.1 节中的 $\dot{x}>0$）。任意 $q_0=(x_0,y_0)\in\mathbb{R}^2$ 的可达集是一个半开平面，它是垂线 $x=x_0$ 右边所有点的集合。在 $u_1\leqslant 0$ 情况下，那么 $R(q_0,\mathcal{U})$ 是同一条垂线左边的一个半闭平面。如果 U 定义为使得 $u_1>0$、$u_2>0$ 的所有 $(u_1,u_2)\in\mathbb{R}^2$ 的集合，那么任意点的可达集是一个象限。

对于约束 $au_1+bu_2=0$，任意点的可达集是 $\mathbb{R}^2$ 中的一条法向量为 (a,b) 的直线。直线的位置依赖于 q_0。因此不同初始状态的可达状态形成了一族平行线。这是微分几何中一个叶

层结构（foliation），这些直线被称为叶子（leaves）[874]。

在 $u_1^2+u_2^2 \leqslant 1$ 情况下，任意 (x_0,y_0) 的可达集是 $\mathbb{R}^2$。因此任何状态都能够到达其他状态。

■

到目前为止都没有考虑障碍区域。令 $\mathcal{U}_{free} \subseteq \mathcal{U}$ 表示所有行动轨迹的集合，这些轨迹产生映射到 X_{free} 的状态轨迹。也就是说，$\mathcal{U}_{free}$ 是从 $\mathcal{U}$ 中排除掉所有导致在某 $t>0$ 进入 X_{obs} 的行动轨迹以后所得到。考虑障碍区域的可达集表示为 $R(x_0,\mathcal{U}_{free})$，即将式（14.4）中用的 $\mathcal{U}$ 替换为 $\mathcal{U}_{free}$。那么对于 $\mathcal{U}_{free}$ 中的轨迹来说，可以应用终止行动来避免由于动量而产生的不可避免的碰撞。这样就定义了一个排除掉不用 u_T 就一定发生碰撞的那些轨迹后所产生的一个更小的可达集。

算法的完备性可用可达集来表达。对于任意 $x_I, x_G \in X_{free}$，如果 $x_G \in R(x_I,\mathcal{U}_{free})$，那么一个完备的算法必须报告行动轨迹的解；否则报告失败。完备性很难实现，除了受到特殊限制的情况以外[173,750]，因此基于采样的完备性概念显得更有价值。

14.2.1.2　时间受限的可达集

考虑能够在某固定的时间内到达的所有状态的集合。令时间受限可达集（time-limited reachable set）$R(x_0,\mathcal{U},t)$ 是 $R(x_0,\mathcal{U})$ 的子集，它是到时间 t 为止能够到达的所有状态的集合。正式地，这是：

$$R(x_0,\mathcal{U},t)=\left\{x_1 \in X \,\middle|\, \exists \tilde{u} \in \mathcal{U}, \exists t' \in [0,t],\ x(t')=x_1\right\} \tag{14.5}$$

对于例 14.2 中的最后一种情况，时间受限可达集是以 t 为半径、以 (x_0,y_0) 为中心的封闭圆面。式（14.5）的一个考虑障碍区域的版本被定义为 $R(x_0,\mathcal{U}_{free},t)$。

假设从 $t=0$ 开始并逐渐增加的 $R(x_0,\mathcal{U},t)$ 的动画。将 $R(x_0,\mathcal{U},t)$ 的边界假设为从 t_0 开始的传播波前。它最后到达 $R(x_0,\mathcal{U})$ 的边界（假设它有一个边界；如果 $R(x_0,\mathcal{U})=X$，那么它就没有边界）。$R(x_0,\mathcal{U},t)$ 的边界实际上可以被阐述为测量流逝时间的代价泛函从 x_0 开始的最优尚需代价的一个水平集。边界也是一种前向投影，就像第 10.1.2 节为离散空间所考虑的那样。在那里由于有大自然的影响，所以将来可能的状态可在前向投影中加以确定。在目前情形下，将来可能的状态由机器人的未定行动来确定，并不是向前看 k 阶段，时间受限可达集将对未来 t 时间进行展望。从本质上讲，存在阶段的一个连续统。

例 14.3（简单汽车的可达集） 用第 13.1.2 节的简单汽车模型可以对可达集进行非常好的阐述。假设 $X=\mathcal{C}=\mathbb{R}^2 \times \mathbb{S}^1$ 和 $X_{obs}=\phi$。

回忆到 Dubins 车只能够向前行驶。从任意一个位形，时间受限可达集如图 14.4（a）所示。时间界限 t 足够小以使汽车的旋转不超过 $\pi/2$。注意图 14.4（a）显示了只给出平移的可达集的一个二维投影。真正的可达集是 $\mathcal{C}$ 中的一个三维区域。如果 $t>2\pi$，那么汽车将能够在一个圆圈中行驶。对于任意 q，考虑 t 趋近于无穷时的情况，这产生 $R(q,\mathcal{U})$。假设一辆汽车在一个无穷大的平坦表面上一直向前行驶。通过沿着一个圆行驶，然后直线行驶一段时间，然后再沿着一个圆行驶，这样它有可能到达任何希望的位形。因此对于任意 $q \in \mathcal{C}$，有 $R(q,\mathcal{U})=\mathcal{C}$。没有倒车装置意味着如果想到达某些位形，那么需要某些额外的机动空间。

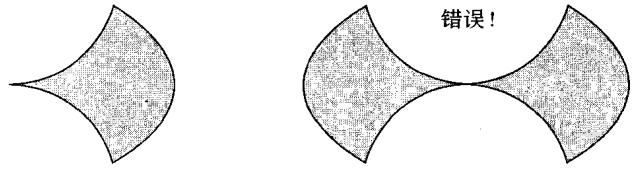


图 14.4 （a）面向右的 Dubins 车的时间受限可达集；（b）这不是 Reeds-Shepp 车时间受限可达集

现在考虑 Reeds-Shepp 车，它能够倒车。对于这种汽车来说，任何时间受限的可达集必须包括 Dubins 车的可达集的所有点，因为虽然加入了新的行动，但没有行动被删除。人们倾向于断言时间受限可达集是如图 14.4（b）所示的样子，但这是错误的。在一个任意小的时间（或者空间）内，能够倒车的汽车可以实现侧向移动。实际上这是通过我们所熟悉的平行移位停车这种机动方式来实现的。在这种情况下，$R(q,\mathcal{U},t)$ 总是包含 q 的一个开集，这意味着它向各个方向增长（见第 15.3.2 节）。正式地，这种特性被称为短时可控性，这将在第 15.4 节进行介绍。 ■

14.2.1.3 后向可达集

可达集的定义关于时间是对称的。不用描述从某 $x \in X$ 所能够到达的所有点，而描述能够到达 $x \in X$ 的所有点会更容易。这与第 10.1.2 节中可选用前向和后向投影的情况相似。

后向可达集（backward reachable set）被定义为：

$$R(x_f,\mathcal{U}) = \left\{ x_0 \in X \,\middle|\, \exists \tilde{u} \in \mathcal{U}, \exists t \in [0,\infty), x(t) = x_f \right\} \tag{14.6}$$

其中 $x(t)$ 由式（14.1）给出，并要求 $x(0) = x_0$。注意到与式（14.4）的相似性。时间受限后向可达集定义为：

$$R(x_f,\mathcal{U},t) = \left\{ x_0 \in X \,\middle|\, \exists \tilde{u} \in \mathcal{U}, \exists t' \in [0,t], x(t') = x_f \right\} \tag{14.7}$$

其中仍然要求式（14.1）中的 $x(0) = x_0$。甚至可以用 $\mathcal{U}$ 的后向时间版本定义的后向可达集来定义完备性。

第 2 章的前向、后向和双向搜索之间是并列关系。在微分约束下的基于采样的规划也存在这种情况。前向和后向可达集指的是能够到达的可能状态。算法就是对这些可达集的子集进行探索。

14.2.2 离散时间模型

本节将介绍一个简单而又有效的方法对行动轨迹空间进行采样。第 14.2.3 节将介绍更加一般的情况。在微分约束条件下，基于采样的运动规划算法都采用对行动轨迹空间进行采样的方法。这就产生了可能行动轨迹的一个简约集合。为了确保某种形式的完备性，运动规划算法应该仔细构造和改进采样集合。就像第 5 章所介绍的，一个采样集合的质量可以用离散度和稠密性来表达。主要的不同之处在于这里的算法所用的是 $\mathcal{U}$ 上的一个采样序列，而不是第 5 章 C 上的采样序列。这是因为路径解不再直接表示在 C 上（或者 X 上）。

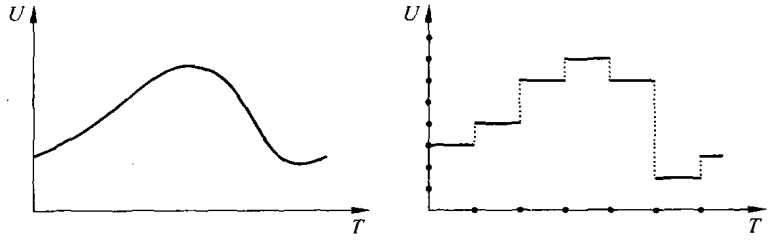
离散时间模型如图 14.5 所示，可以从三个方面对它进行描述：

（1）时间 T 划分为长度为 Δt 的区间。这样就有了阶段的概念，其中阶段 k 表示已经流

逝的时间是 $(k-1)\Delta t$。

（2）选择行动空间 U 的一个有限子集 U_d。如果 U 已经是有限的，那么可以选择 $U_d = U$。

（3）在每个时间区间内行动 $u(t) \in U_d$ 必须保持固定常数值。



U 中的一个轨迹　　U_d 中的一个轨迹

图 14.5　离散时间模型产生 $\mathcal{U}_d \subset \mathcal{U}$，它是将时间划分为等长区间并在每个区间上应用一个常数行动；行动从 U 的一个有限子集 U_d 中进行选择

前两个方面是对时间和行动空间进行离散化。第三个方面将离散时间与行动轨迹空间联系起来。令 $\mathcal{U}_d$ 表示在给定离散时间下所允许的所有行动轨迹的集合。注意到 $\mathcal{U}_d$ 完全确定了离散时间模型。

对于某些问题，U 可能已经是有限的。例如，假设启动一个自由漂浮航天器上多个推进器（开启或者关闭着的）中的一个。在这种情况下不需要 U 的离散化。在某些更一般的情况下，U 可能是连续集合。可以用第 5.2 节的采样方法来确定一个有限子集 $U_d \subseteq U$。

$\mathcal{U}_d$ 中的任意行动轨迹可以表示为一个行动序列 $(u_1, u_2, \cdots, u_k)$，其中 $u_i \in U_d$ 给出从 $(i-1)\Delta t \sim i\Delta t$ 这段时间内所采取的行动。假定 k 阶段以后终止行动。

14.2.2.1　可达性的图

时间离散化以后，可达集为适应 $\mathcal{U}_d$ 而被修改为 $R(x_0, \mathcal{U}_d)$。一个问题是：在可达集上进行采样的效果如何？也就是说，$R(x_0, \mathcal{U})$ 和 $R(x_0, \mathcal{U}_d)$ 有什么不同？这可以通过定义一个可达性的图来进行讨论，图将通过一个规划算法而逐渐显现出来。

令 $T_r(x_0, \mathcal{U}_d)$ 表示一个可达性的树（reachability tree），它对从 x_0 开始的所有轨迹的集合进行编码，而 x_0 是通过应用 $\mathcal{U}_d$ 中的轨迹得到的。$T_r(x_0, \mathcal{U}_d)$ 的每个顶点都是一个可达状态 $x \in R(x_0, \mathcal{U}_d)$。$T_r(x_0, \mathcal{U}_d)$ 的每条边都是有向的。边的起点代表一个起始状态，它的终点代表在时间 Δt 时间内应用常数行动 $u \in U_d$ 所得到的状态。每条边 e 表示一个行动轨迹片段 $e: [0, \Delta t] \to U$。通过用式（14.1）对 $f(x,u)$ 从 0 到 Δt 进行积分就可以得到一个状态轨迹 $\tilde{x}_e$。

因此根据 $\tilde{x}_e$，可以认为 T_r 是 X 中的一个拓扑图（T_r 是 $T_r(x_0, \mathcal{U}_d)$ 的缩写）。T_r 的一个刈幅 $S(T_r)$ 是：

$$S(T_r) = \bigcup_{e \in E} \bigcup_{t \in [0, \Delta t]} x_e(t) \tag{14.8}$$

其中 $x_e(t)$ 表示 t 时刻从边 e 所得到的状态。回忆例 4.6 的拓扑图和第 5.5.1 节介绍的刈幅。

例 14.4（Dubins 车的可达性的树） Dubins 车的可达性树的几个阶段如图 14.6 所示。假设有三个行动（直行、右转、左转），选择 Δt 使得如果向右转或者向左转，那么汽车足够旋转 $\pi/2$。第二阶段以后，树中有 9 个叶子，如图 14.6（a）所示。每个阶段产生 3^k 个

新叶子。在图 14.6（b）中，在 $k=4$ 阶段增加了 81 个新叶子，那么总共有 $81+27+9+3+1$ 个顶点。在某些情况下，不同的行动序列能够到达相同的状态。完成前 4 个阶段以后的刈幅是访问过的所有点的集合。这是 C 的一个子集，它是所有顶点和每个 $e \in E$ 的 $\tilde{x}_e$ 所绘出的所有点的并集。 ■

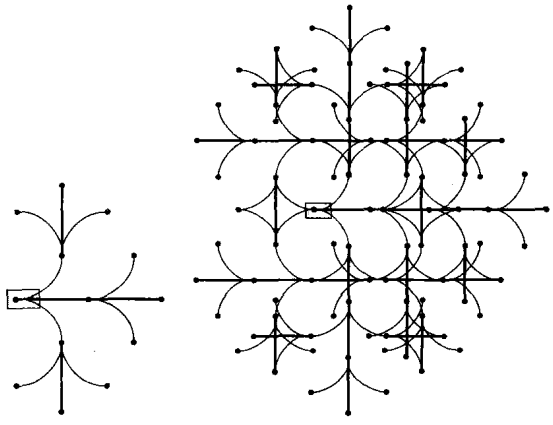


图 14.6 有三个行动的 Dubins 车的一个可达性的树，第 k 阶段产生 3^k 个新的顶点

从例 14.4 可以看出有两个或者两个以上行动轨迹可能到达相同的状态。因为可以将每个行动轨迹表示为一个行动序列，为检验从不同行动序列是否到达相同状态所进行的经典人工智能搜索中会出现我们所熟悉的问题。对于某些系统，利用这些信息可以使得可达性问题得到显著简化。如果从多个行动序列到达相同状态，那么只需要用一个顶点来表示。

这就产生了一个可达性的有向图 $\mathcal{G}_r(x_0, \mathcal{U}_d)$，它是通过合并 $T_r(x_0, \mathcal{U}_d)$ 中的相同状态而得到的。如果每个行动序列到达唯一状态，那么可达性的图成为可达性的树。但是如果多个行动序列到达相同状态，那么这用单个顶点 $\mathcal{G}_r$ 来表示。这里首先使用可达性的图。至于一个可达性的树，一个可达性的图可以被阐述为 X 中的一个拓扑图，它的刈幅 $S(\mathcal{G}_r)$ 通过修改式（14.8）来定义。

到达相同状态的最简单的情况可以从例 2.1 看到。根据第 13 章中所介绍的，平面上的离散栅格可以建模为 $\dot{x}=u_1$ 和 $\dot{y}=u_2$ 形式的一个系统，其中状态空间 $X=\mathbb{R}^2$。行动的离散集合 U_d 为 $\{(1,0),(0,1),(-1,0),(0,-1)\}$。令 $\Delta t=1$。在这种情况下，可达性的图变成我们熟悉的二维栅格。如果初始状态为 $(0,0)$，那么栅格顶点由两个坐标都是整数的所有状态组成。

对任何系统随意地进行离散化，通常不会出现这种良好的栅格。但是在许多情况下可以对离散化进行仔细选择以使状态落在一个栅格或网格中。在基于采样的规划中这有许多优越性。对系统 $\ddot{q}=u$，第 14.4.1 节介绍了利用这种结构的方法。它甚至可以扩展到更一般的系统，只要系统可以表示为 $\ddot{q}=g(q,\dot{q},u)$ 且不是欠驱动的。通过巧妙的离散化，还可以迫使一大类非完整性系统[1]落在一个网格上[765]。这通常很难实现，并且在大多数离散化下，可达性图的顶点在可达集中是稠密的。

也可能定义可达性树和可达性图的后向版本，这与得到后向可达集的方法相同。它们

1 这类系统都是无漂移的幂零系统。“幂零”这个术语将在第 15.5 节进行定义。

给出的初始状态和行动序列能够到达一个给定的目标状态，并且并不比前向情况更难定义或者更难计算。它们可能看起来更困难，但需要记住的是初始状态并不固定，因此不会出现 BVP。初始状态可以用状态转移方程的逆时间积分来得到，参见第 14.3.2 节。

14.2.2.2　$\dot{x}=u$ 的分辨率的完备性

基于采样的完备性概念可以用可达集和可达性的图来表示。要求对 U 的采样方式是使可达性图的顶点最终在可达集中变得稠密，同时也确保可达性的图能被系统地搜索。所有的完备性概念都可以根据前向或者后向可达性的图来表示。但这里只描述前向情况，因为后向情况是相似的。

为了有助于衔接第 II 部分中关于运动规划的内容，首先假设：（1）$X=C=\mathbb{R}^2$；（2）一个状态表示为 $q=(x,y)$；（3）$U=[-1,1]^2$；（4）状态转移方程是 $\dot{x}=u_1$ 和 $\dot{y}=u_2$。假设 U 用离散时间模型。令 $\Delta t=1$ 且：

$$U_d=\{(-1,0),(0,-1),(1,0),(0,1)\} \tag{14.9}$$

这产生了例 7.4 的 Manhattan 运动模型。产生图 7.40 所示的阶梯形路径。在目前情形下，这些路径可通过对行动轨迹进行积分来得到。从某状态 x_I，可达性图表示所有可能从式（14.1）得到的单位步长的阶梯形路径的集合。

假设在这种模型下，X_{free} 是 $\mathbb{R}^2$ 的一个有界的开子集。在这种情况下可以清楚地表达与第 5 章的分辨率完备性之间的联系。对于任意固定的 Δt，一个有确定分辨率的栅格是通过可达性的图来隐式定义的。任务是寻找一个导致目标（或者可达性图中接近于它的一个顶点）的行动序列，同时保持在 X_{free} 中。这种序列可以通过系统搜索来发现，就像第 2.2 节介绍的那样。如果搜索是系统进行的，那么它将正确地确定可达性的图是否对解进行了编码。如果解不存在，那么规划算法将 Δt 降低一个固定因子（例如 2），然后继续进行系统搜索。该过程重复进行直到发现问题的解。如果没有解，那么算法将永远进行下去。这种方法在第 5 章的意义上是分辨率完备的，即使所有路径都是用行动序列表达的。

这种简单模型与普通运动规划之间的联系是非常清楚的，因为行动轨迹积分产生沿着栅格的运动。当时间离散化分辨率提高时，阶梯形路径可以任意逼近某路径解。观察图 14.5 可以看出，当关于 U 和 T 提高采样分辨率时，通过离散时间近似所得到的轨迹收敛于对某 $\tilde{u}$ 进行积分所得到的任意轨迹。一般情况下，当 Δt 以及在 U 中进行采样的离散度趋近于零时就会收敛。这对 $\dot{x}=u$ 并且 X 是任意平滑流形这种更加一般的情况也是成立的。假设将栅格放置在 X 上并通过减小 Δt 来任意细分之。

14.2.2.3　$\dot{x}=f(x,u)$ 的分辨率的完备性

除了 $\dot{x}=u$ 这种平凡情况以外，可达性的图通常不是一个简单的栅格。即使 X 是有界的，可达性的图也可能有无限多个顶点，尽管 Δt 是固定的和 U_d 是有限的。例如考虑 $\Delta t=1$ 离散化下的 Dubins 车。对所有 $t\in T$，固定 $u_\phi=-\phi_{\max}$（左转）。这在可达性的图中独自地生成了可数无限多个顶点。圆的周长为 $2\pi\rho_{\min}$，其中 $\rho_{\min}$ 是最小转弯半径。令 $\rho_{\min}=1$。因为周长是一个无理数，所以不可能在运动 k 秒后再次访问初始点，k 是某整数。当然也不可能再次访问圆上的任意一个点。可达性的图中顶点的集合实际上在圆上是稠密的。图 14.6 不会发生这种情况，因为 Δt 和周长的关系是有理的。考虑在目前情形下当 $\rho_{\min}=1/\pi$ 且 $\Delta t=1$ 时

会发生什么。

假设 $\dot{x}=f(x,u)$ 并采用离散时间模型。为了确保离散时间近似的收敛性，f 必须是良好行为的。这可以通过要求 f 关于 u 和 x 的所有导数的上界和下界都由常数来加以限制。更一般情况下，假设 f 是 Lipschitz 的，这是导数存在的一个等价条件，但是它也应用在不可微的点上。如果 U 是有限的，那么 Lipschitz 条件为——对于所有 $x,x' \in X$ 和所有 $u \in U$，存在某 $c \in (0,\infty)$ 使得：

$$\|f(x,u)-f(x',u)\| \leqslant c\|x-x'\| \tag{14.10}$$

其中 $\|\cdot\|$ 表示 X 上的范数。如果 U 是无限的，那么条件为——对于所有 $x,x' \in X$ 和所有 $u,u' \in U$，必须存在某 $c \in (0,\infty)$ 使得：

$$\|f(x,u)-f(x',u')\| \leqslant c(\|x-x'\|+\|u-u'\|) \tag{14.11}$$

从直觉上看，Lipschitz 条件指出的是：如果 x' 和 u' 逼近 x 和 u，那么它们代入 f 时所产生的误差是可控的。如果需要收敛到关于一个代价泛函的最优轨迹，那么 $l(x,u)$ 也需要 Lipschitz 条件。在这种假设下，如果 Δt 和 U_d 的采样点离散度趋近于零，那么对离散行动序列进行积分所得到的轨迹可以任意逼近解轨迹。也就是说，行动序列可以任意逼近任何 $\tilde{u} \in U$。如果 f 是 Lipschitz 的，那么对式（14.14）进行积分所产生的结果与逼近 $\tilde{u}$ 的行动序列所产生的结果近似相同。

在 Δt 和 U_d 的离散度趋近于零这种极限情况下，可达性的图在可达集 $R(x_I,\mathcal{U})$ 中变成稠密的。保证对一个栅格进行系统搜索并不困难，因为在每个分辨率下只有有限个顶点。遗憾的是，对于某固定的离散时间模型来说，其可达性的图在一般情况下可能有可数无限多个顶点，即使 X 是有界的。

在可达性的图是可数无限的情况下，分辨率完备性算法仍然存在。为了理解它，可以采用三角形枚举法（triangular enumeration），它能够证明 $\mathbb{N}\times\mathbb{N}$ 是可数的，其中 $\mathbb{N}$ 是自然数的集合。证明过程是通过给出一个序列来进行的：序列从 (0,0) 开始，并对角地前后扫过第一象限。在极限状况下将会覆盖所有的点。应用相同的思路可以得到分辨率完备的算法。可以构造一个离散时间模型的序列，对该序列来说时间步长 Δt 和 U 的采样离散度均接近于零。每次离散化都产生一个有可数个顶点的可达性图。

通过进行相同的之字形搜索也可以建立一个分辨率完备的算法。之字形搜索在上面用来显示 $\mathbb{N}\times\mathbb{N}$ 是可数的。如图 14.7 所示，假设 U 是有限的且 $U_d=U$。沿着横轴的是一个改进的离散时间模型序列。每个模型生成它自己的可达性的图，对于这些图来说，系统的搜索最终会探索所有的顶点。假设这种探索每次只进行一步，每步到达一个新的顶点。图 14.7 中的纵轴指示搜索算法到达的顶点数。计算机用一个可数无限集合并行地探索所有可达性的图。即使只有一个计算机，也确信能够用之字形搜索探索到所有点。因此如果 U 是有限的，那么一个分辨率完备的算法总是存在的。如果 U 不是有限的，那么当时间步减小时，U_d 必须能够进一步细分。当然，也有许多其他方法来系统地探索所有可达性的图。更难的是寻找一个在实际中能够产生良好性能的方法。

目前的讨论都假设基于采样的算法能够揭示可达性图的一个子图。这忽略了算术精度、数值积分误差等数值问题。这些问题可以融入分辨率完备性分析中[198]。

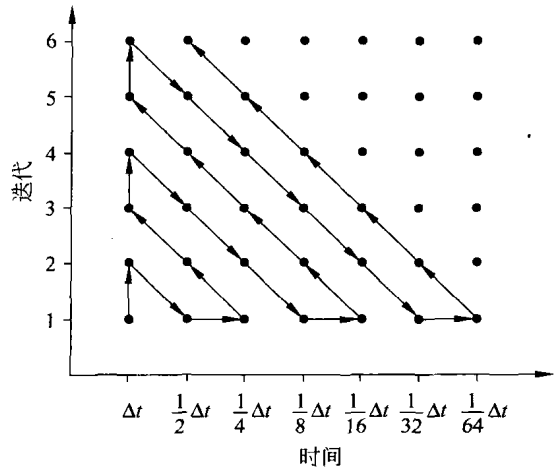


图 14.7　通过系统地交替探索不同的可达性图，能够实现分辨率完备性，即使每个可达性图有可数无限多个顶点

14.2.3　运动本原

第 14.2.2 节的离散时间模型只是对行动轨迹空间进行离散化的许多可能方法中的一种。现在可以认为它是确定运动本原（motion primitives）的一种特殊情况。如果在固定时间区间上将行动限制为常数，这对许多应用来说可能过于苛刻。假设我们想使视频游戏或者电影中数字演员的运动自动化。假设我们有一个运动本原数据库。例如这些本原能够从运动数据中获取[35,556]。例如，如果设计演员进行功夫格斗，那么每个运动序列可能对应于一个基本运动，如踢或者用拳击打。这种运动本原就不可能对应于固定时间区间上的常数行动。运动本原的持续时间通常是变化的。

定义一个更一般的离散化通常就可以处理这种模型。离散时间模型可以用来表述一个离散时间状态转移方程：

$$x_{k+1} = f_d(x_k, u_k) \tag{14.12}$$

其中 $x_k = x((k-1)\Delta t)$，$x_{k+1} = x(k\Delta t)$，u_k 是 U_d 中的行动，它在 $(k-1)\Delta t$ 到 $k\Delta t$ 这段时间内得到应用。因此 f_d 是一个函数 $f_d : X \times U_d \to X$，它代表对原始状态转移方程 f 的一种近似。在 Δt 内所应用的每个常数行动 $u \in U_d$ 都可以认为是一个运动本原。

现在将上述构造过程推广到允许更一般的运动本原的情况。令 $\tilde{u}^p$ 表示一个运动本原，它是从一个时间区间到 U 的一个函数。令时间区间为 $0 \sim t_F(\tilde{u}^p)$，$t_F(\tilde{u}^p)$ 是一个依赖于特定本原的最后时刻。从任意状态 $x \in X_{free}$，假设有一个运动本原的集合 $\mathcal{U}^p(x)$。该集合可能是无限的，在这种情况下必须用一个局部规划方法在运动本原空间上最后进行某种附加采样。在离散阶段上的状态转移方程可以被定义为：

$$x_{k+1} = f_p(x_k, \tilde{u}_k^p) \tag{14.13}$$

其中 $\tilde{u}_k^p$ 是一个必须从 $\mathcal{U}^p(x_k)$ 中选择的运动本原。时间离散模型和式（14.12）可以认为是运动本原在固定时间区间 $[0, \Delta t)$ 上都是常数这种特殊情况。注意到在式（14.13）中，阶段索引 k 不一定对应于时间 $(k-1)\Delta t$。k 只表示到目前为止已经应用了 $k-1$ 个运动本原，并

且现在是决定第 k 个运动本原的时候。目前的时间是通过对到目前为止所有 $k-1$ 个运动本原的持续时间进行求和得到的。如果给出所有 $x \in X$ 的本原的一个集合 $U^p(x)$，那么可达性的图及其刈幅可以用对离散时间情况的简单扩展来定义。离散时间模型 U_d 现在可以阐述为运动本原的一个特殊集合。

对于某些运动本原来说，如果没有转移运动，那么也许不可能立刻对其进行排列。例如在文献[364]中，为直升机的自主飞行定义了两种不同类型的运动本原，它们分别被称为平衡轨迹和机动。平衡轨迹对应于稳定运动，机动对应于用来在两种稳定运动之间进行转移的不稳定运动。从一种平衡轨迹到另外一种平衡轨迹的转移只能进行一次机动。该问题可以建模为一个混合系统，其中运动本原表示一种模式[362]（回忆第 7.3 节、第 8.3.1 节和第 10.6 节的混合系统的概念）。增广状态空间为 $X \times M$，其中 M 是模式的集合。转移方程（14.13）可以扩展到增广状态空间以使运动本原除了能够改变原始状态以外还能够改变模式。直升机可能的轨迹跟踪机动自动机中的路径。文献[362]的一个例子如图 14.8 所示。每条边和每个顶点对应于机动自动机中的一种模式。每条边或者顶点实际上对应于本原的一个参数族，根据状态来选择其中一个特定本原。文献[455]为动画人物提出了一个相似的状态机，运动本原被称为行为。

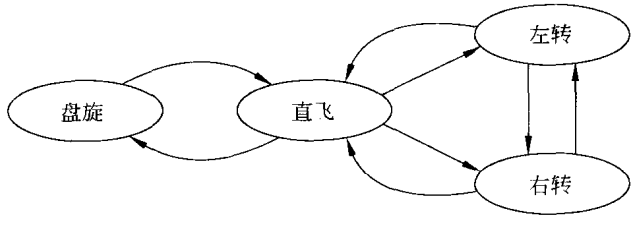


图 14.8 Frazzoli 提出的一个机动自动机[362]获取关于对可允许运动本原序列的约束

基于一般运动本原的离散化具有很大的灵活性，并且在许多情况下，基于采样的规划算法的性能可以得到显著的提高。主要的不足之处在于它增加了建立分辨率完备性的负担。

14.3 再论基于采样的运动规划

既然已经定义了微分约束下运动规划的基本概念，那么将焦点转到对第 5 章的基于采样的规划方法进行扩展。这首先涉及给第 5.4 节的递增采样和搜索框架加入微分约束。按照一般框架，第 14.4 节介绍了几种常用的方法，它们是框架的特殊情况。如果可以用 BVP 进行有效求解，那么就可能将第 5.6 节的基于采样的路线图进行扩展以处理微分约束。

14.3.1 基本组成

本节描述如何将第 5.1 节～第 5.3 节的内容进行适应性修改来处理相空间和微分约束。

14.3.1.1 X 中的距离与体积

回忆第 5 章，一些基于采样的规划算法依赖于测量 C 中的距离或者体积。如果 $X=C$，例如第 13.1.2 节中的轮式系统，那么可以直接应用第 5.1 节的概念。对于除了包括位形变

量以外可能还包括相变量的一般状态空间 X 来说，需要用到等价性。在多数情况下，相变量的拓扑是平凡的。例如，如果 $x=(q,\dot{q})$，那么每个 $\dot{q}_i$ 分量被限制在 $\mathbb{R}$ 一个区间上。在这种情况下，如果 n 是 X 的维数，那么速度分量只是 $\mathbb{R}^{n/2}$ 中的一个轴向矩形区域。在这种情况下用笛卡儿积将定义在 C 上的测度和度量扩展到 X。

用式（5.4）给出的笛卡儿积的方法可以定义一个度量。问题在于对不同分量进行加权并组合到一个标量函数这个过程具有随意性。在 C 的情况下，这已经涉及平移和旋转的组合。对于 X 的情况，这还包括速度分量，这使得很难选择权值。

黎曼度量　在平滑流形上定义一个度量的一个严格的方法是定义一个度量张量（或者黎曼张量），它是两个切向量的二次函数。可以认为它是 X 上的一个内积，可以用来测量角度。这就产生了黎曼度量的定义，它基于 X 上的最短路径（测地线）[135]。在第 13.4.1 节中的拉格朗日力学中有它的一个例子。动能（见式（13.70））就是所要求的度量张量，测地线就是动力系统为能量守恒所进行的运动。度量可以被定义为连接两个点的测地线的长度。如果所选择的黎曼度量有某种物理意义，如拉格朗日力学中的情况，那么所产生的度量能够提供非常有意义的信息。然而遗憾的是，很难计算它的值或者计算所付出的代价太大。

理想距离函数　定义 X 上的距离的理想方法是用代价泛函并将从 $x\in X_{free}$ 到 $x'\in X_{free}$ 的距离定义为从 x 到 x' 最优尚需代价。在某些情况下，它也被称为非完整度量、Carnot-Caratheodory 度量或者子黎曼度量[599]。注意到这不是真正的度量，因为代价不可能是对称的。例如 Dubins 车向前行驶一个短的距离比向后行驶一个短的距离更短。如果有障碍物，那么甚至不可能到达汽车后面的位形。

这种距离的概念有点令人感到为难，因为它要求求解表述 14.1 的运动规划问题的最优解。因此它在运动规划算法中并不十分有效。尽管如此，理解该理想距离对于设计 X 上实用的距离函数是有用的。例如 Dubins 车不用加权欧氏度量（也被称为 Mahalanobis 度量），基于两个位形之间最短路径的长度就可以定义距离函数。这些长度很容易计算，它们都基于将在第 15.3 节介绍的最优曲线族。该距离函数忽略了障碍物，但是它依然应该能够提供比加权欧氏度量更好的距离信息。对涉及动力学的汽车模型来说它也是非常有用的。

一般的思路是不用代价昂贵的计算就能够尽可能接近最优尚需代价。忽略某些约束，如障碍物或者动力学等，通常有可能计算最优尚需代价的一个有用的低估值。这可能有助于应用 A^* 方法。

定义测度　如果与相变量有关的拓扑是平凡的，那么 C 上的测度扩展到 X 是很容易的。但是不可能得到一个不变测度。在多数情况下，C 是一个其中存在 Haar 测度的变换群，从对 C 的参数化不敏感这种意义上，它产生“真实的”体积。从第 5.1.4 节中可以观察到 $SO(3)$ 的情况。对于一个一般的状态空间 X，Haar 测度可能不存在。如果定义了一个黎曼度量，那么存在曲面积分和体积的概念[135]，但是这些可能很难用在基于采样的规划算法中。

14.3.1.2　采样理论

第 14.2.2 节已经介绍了关于采样的一些内容。至少有两个连续空间：X 和时间区间 T。在多数情况下，行动空间也是连续的。必须用某种方式对每个连续空间进行采样。在极限状况下，重要的是任何采样序列在进行采样的空间上是稠密的。第 14.2.2 节的分辨率完备

性也要求如此。

对 T 和 U 的采样可以直接用第 5.2 节的随机的或者确定性的方法来实现。时间只是 $\mathbb{R}$ 的一个区间，U 典型地表示为 $\mathbb{R}^m$ 的一个凸 m 维子集。例如 U 通常是 $\mathbb{R}^m$ 的一个轴向矩形子集。

一些规划方法可能要求对 X 进行采样。第 5.2 节偏差和离散度的定义很容易适应性地修改到任意测度空间和度量空间。即使它很容易定义一个良好的指标，也许仍然很难或者不可能产生能够最优化性能指标的采样点。

避免这种问题的一个简便方法是用 X 的坐标邻域。这使流形以 $\mathbb{R}^n$ 中的一个 n 维区域出现，在一些情况下该区域是矩形的。这就可以以一种非常简单的方式应用第 5.2 节的采样概念。采样的质量取决于用来定义坐标邻域的特定参数表示。注意到当用坐标邻域时（假设 X 是一个方体），必须考虑相应的等同问题。

14.3.1.3 碰撞检测

如第 5 章所述，有效的碰撞检测算法是基于采样的规划的关键。如果 $X=C$，那么直接应用第 5.3 节的方法。如果 X 包括相约束，那么必须进行附加检验。通常给定这些约束，所以计算就很简单。回忆第 4.3 节，这对于 C 上的障碍物约束来说并不很有效，这是由于 $\mathcal{W}$ 中的障碍物和 C 中的障碍物之间存在复杂的映射。

如果只进行逐点检验，那么并不能保证点与点之间的轨迹片段都处在 X_{free} 中。该问题将在第 5.3.4 节用碰撞检测算法中的距离信息予以论述。对 $h_i(x)\leqslant 0$ 形式的相约束来说同样存在这个问题。在这种一般形式中，没有附加信息可以用来确保 x 的某邻域包含在 X_{free} 中。幸运的是在多数应用中相约束并不复杂，并且有可能确保 x 与约束边界之间至少保持某距离。一般情况下，需要对每个相约束进行仔细分析以保证状态轨迹片段没有违规。

总之，确定是否 $x\in X_{free}$ 会涉及：

（1）用第 5.3 节中的一个碰撞检测算法来确保 $\kappa(x)\in C_{free}$。

（2）检查 x 以确保已经满足 $h_i(x)\leqslant 0$ 这种形式的约束。

从理论上讲，所有轨迹片段都应该进行检查。但是在实际上只检查个别点，这样做虽然效率很高，但是从技术上这并不能保证完全正确。

14.3.2 系统模拟器

因为存在式（14.1），所以微分约束条件下的基于采样的规划需要一个新的分量。现在用一个行动轨迹来表达运动，但是碰撞检测和检验约束必须在 X 中进行。因此在规划过程中，需要频繁地对系统 $\dot{x}=f(x,u)$ 进行积分。类似于将碰撞检测建模为一个“黑箱”，积分过程被建模为一个称为“系统模拟器”（system simulator）的模块，如图 14.9 所示。因为本章所考虑的系统是时不变的，那么任何积分的开始时间总可以移到从 $t=0$ 开始。积分可以认为是通过计算状态轨迹来实现式（14.1）的一个模块，状态轨迹由给定的初始状态 $x(0)$、行动轨迹 $\tilde{u}_t$ 和时间 t 产生而来。逐步增长的模拟器封装了积分状态转移方程的详细内容，所以在规划器的设计中不需要详细讨论它们。但是来自于特定状态转移方程的信息在规划算法的设计中仍然非常重要。

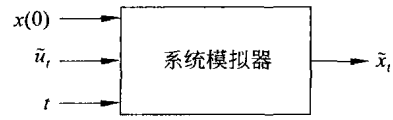


图 14.9　使用一个系统模拟器，用 $\tilde{u}_t:[0,t]\to U$ 从 $x(0)$ 对系统 $\dot{x}=f(x,u)$ 进行积分产生一个状态轨迹 $\tilde{x}_t:[0,t]\to X$；有时指定 $\tilde{x}$ 为一个参数化的路径，但是通常它被近似为 X 中的一个采样序列

封闭形式的解　按照式（14.1），必须积分行动轨迹来产生一个状态轨迹。在某些情况下，这种积分产生了封闭形式的解。例如，如果系统是一个积分器链，那么很容易得到 $x(t)$ 的多项式表示。例如，假设 q 是一个标量，且 $\ddot{q}=u$。如果 $q(0)=\dot{q}(0)=0$ 且应用常数行动 $u=1$，那么 $x(t)=t^2/2$。如果 $\dot{x}=f(x,u)$ 是一个线性系统（包括积分器链），那么总能够得到状态轨迹的封闭表达式。这是基于矩阵指数，许多控制方面的书籍中都给出了它的表达式（例如文献[194]）。

欧拉方法　对于多数系统，必须进行数值积分。将 t 分成更小的区间，并将计算数值解的经典方法迭代应用到微分方程，这样就可以构造基于数值积分的系统模拟器。欧拉法是这些方法中最简单的。令 Δt 表示一个小的时间区间，在该区间上进行近似。这可以认为是系统模拟器的内部参数。实际上，Δt 通常比第 14.2.2 节中离散时间模型所用的 Δt 更小。假设给定 $x(0)$ 和 $u(0)$，任务是估计 $x(\Delta t)$。

随时间进行积分，经过某固定的 Δt 以后，状态转移方程可以用来确定状态。例如，如果给定 $x(0)$，并且在区间 $t'\in[0,\Delta t]$ 上已知 $u(t')$，那么 Δt 时刻的状态为：

$$x(\Delta t)=x(0)+\int_0^{\Delta t} f(x(t),u(t))\mathrm{d}t \tag{14.14}$$

积分不能直接求得，因为 $x(t)$ 出现在被积函数中，且对于时间 $t>0$，$x(t)$ 未知。

用：

$$f(x,u)=\dot{x}=\frac{\mathrm{d}x}{\mathrm{d}t}\approx\frac{x(\Delta t)-x(0)}{\Delta t} \tag{14.15}$$

求解 $x(\Delta t)$ 就产生了经典的欧拉积分法：

$$x(\Delta t)\approx x(0)+\Delta t\cdot f(x(0),u(0)) \tag{14.16}$$

近似误差取决于 $x(t)$ 变化的快慢以及区间 Δt 的长度。如果规划算法应用运动本原 $\tilde{u}^p$，那么它将 $t_F(\tilde{u}^p)$ 作为时间输入，并且系统模拟器可能进一步细分时间区间以保持更高精度。这允许规划算法的研究人员可以忽略数值精度问题。

龙格-库塔法　尽管欧拉积分非常有效，也非常容易理解，但是一般情况下它的近似精度差。$\tilde{x}$ 在 $t=0$ 处进行泰勒级数展开，产生：

$$x(\Delta t)=x(0)+\Delta t\cdot\dot{x}(0)+\frac{(\Delta t)^2}{2!}\ddot{x}(0)+\frac{(\Delta t)^3}{3!}x^{(3)}(0)+\cdots \tag{14.17}$$

将它与式（14.16）进行比较可以看出，泰勒级数是一种精确表示（如果 $\tilde{x}$ 是解析的），欧拉法只用到泰勒级数的第一项。因此被忽略的项就是近似误差。如果 $x(t)$ 大概是线性的，那么误差可能很小；但是如果 $\dot{x}(t)$ 或者更高阶的导数变化很快，那么近似的效果会很差。

龙格-库塔法是基于使用泰勒级数的更高阶的项。应用最广泛而且非常有效的一种数值积分方法是四阶龙格-库塔法。它很容易实现，而且在多数应用中的效果很好，并且它比欧

拉法更受欢迎。在 $x\left(\frac{1}{2}\Delta t\right)$ 处进行泰勒级数展开就可以产生该方法。其状态在近似过程进行估计。

四阶龙格-库塔法为：

$$x(\Delta t) \approx x(0) + \frac{\Delta t}{6}(w_1 + 2w_2 + 2w_3 + w_4) \tag{14.18}$$

其中：

$$\begin{aligned} w_1 &= f(x(0), u(0)) \\ w_2 &= f\left(x(0) + \frac{1}{2}\Delta t w_1, u\left(\frac{1}{2}\Delta t\right)\right) \\ w_3 &= f\left(x(0) + \frac{1}{2}\Delta t w_2, u\left(\frac{1}{2}\Delta t\right)\right) \\ w_4 &= f(x(0) + \Delta t w_3, u(\Delta t)) \end{aligned} \tag{14.19}$$

尽管它比欧拉法的计算量更大，但是它在实际应用中是值得的，因为它对精度有很大提高。注意到用到三个不同时刻的行动：0、$\frac{1}{2}\Delta t$ 和 Δt。如果行动在 $[0, \Delta t)$ 上是常数，那么三个时刻用相同的值。

近似误差取决于 $\tilde{x}$ 的高阶导数的变化快慢。这可以用泰勒级数的剩余项来表达。实际上，在龙格-库塔积分的后续的迭代中适应性修改 Δt 是很有利的。例如文献[249]中建议 Δt 刻度为 $(\Delta t / \Delta x)^{1/5}$，其中 $\Delta x = \|x(\Delta t) - x(0)\|$ 为 $\mathbb{R}^n$ 中的欧氏距离。

多步法 龙格-库塔法体现了在简单性和有效性之间一种非常常见的权衡折中。但是通过仔细关注积分问题，还可能进一步提高有效性。欧拉法和龙格-库塔法通常被称为单步法。也存在多步法，它依赖于将会进行一系列积分这样的事实，所用的方式类似于第 5.3.3 节中的增量碰撞检测。关键的问题是确保方法正确地初始化、确保数值稳定性、对误差进行估计以适应性地调整步长。许多关于数值分析的书籍介绍了多步法[52,443,865]。其中最常用的是 Adams 法。

多步法要求将更多的注意力投向方法的理解和实现上。对于一个特定应用，如果决定沿着这个思路进行工作，那么应该基于规划、碰撞检测、数值积分所花费的代价。如果主要目的是积分并且效率至关重要，那么多步法能够比龙格-库塔法节省大量运行时间。

黑箱模拟器 对于某些应用，也许没有状态转移方程，但是在给定目前状态和行动轨迹的情况仍然可能计算将来的状态。例如这可能发生在一个复杂的软件系统中，它模拟一辆机动车或者一些在桌子上弹跳的零件的动力学。在计算机图形学的应用中，模拟可能来自于捕获的运动数据。许多模拟器可能用的是形如 $g_i(x, \dot{x}, u) = 0$ 的隐含微分约束，而不是 $\dot{x} = f(x, u)$。在这种情形下，许多基于采样的规划器都可以应用，因为它们只依赖于系统模拟器的存在性。因此规划算法从系统是如何表示和积分的等这些特定的细节中解脱出来。

逆时间系统模拟 一些规划算法要求从逆时间方向进行积分。对于某给定的 $x(0)$ 和从 $-\Delta t$ 到 0 的行动轨迹，后向系统模拟器计算一个状态轨迹 $\tilde{x}:[-t, 0] \to X$，当用 $\tilde{u}_t$ 从 $-\Delta t$ 到 0 对其进行积分时产生 $x(0)$。这似乎就像一个逆控制问题[858]或者如图 14.10 所示的一个 BVP，但是它更加简单。为给定的初始状态和目标状态确定行动轨迹则更加复杂；但是在逆时间积分中，给定行动轨迹和最终状态，初始状态不需要固定。

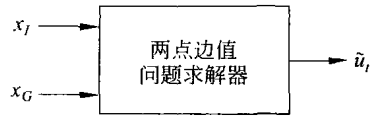


图 14.10　第 15 章中的某种方法可以解决没有 X_{obs} 的两点边值问题；对于多数系统来说两点边值问题很难解决，但是它比系统模拟器更有力；例如在构造满足微分约束的基于采样的路线图中，它很有价值

式（14.14）的逆时间版本为：

$$x(-\Delta t) = x(0) + \int_0^{-\Delta t} f(x(t),u(t))\mathrm{d}t = x(0) + \int_0^{\Delta t} -f(x(t),u(t))\mathrm{d}t \qquad (14.20)$$

它依赖于 $\dot{x} = f(x,u)$ 是时不变的这个事实。因此只对状态转移方程取反就可以得到逆时间积分。欧拉法和龙格-库塔法能够以通常的方式应用于 $-f(x(t),u(t))$。

14.3.3　局部规划

第 5 章的方法是基于一个既简单又有效的局部规划方法（LPM）的存在性。它是第 5.4 节的递增采样和搜索框架和第 5.6 节的基于采样的路线图框架中的一个重要部分。在没有障碍物和微分约束的情况下，定义联系两个位形的 LPM 是很平凡的。例如它们能够用 C 中最短路径（测地线）来连接。第 5.6 节的基于采样的路线图方法依赖于这种简单的 LPM。

在有微分约束的情况下，构造一个连接两个位形或者状态的 LPM 的问题是很难的。回忆第 14.1 节可知，这是经典的 BVP，对于多数系统来说是很难求解的。在基于采样的规划算法中有两种主要方法来解决这种困难：

（1）设计采样方案，即仔细选择运动本原使得 BVP 容易得到解决。

（2）设计规划算法以使需要解决的 BVP 尽可能少。这种情况下的 LPM 不指定必须到达的目标状态。

对于前一种方法，求解 BVP 可以认为是图 14.10 所示的一个黑箱，在没有障碍物的情况下它有效地连接 x_I 和 x_G。在钢琴搬运工问题中，这通过沿着 C 中最短路径运动就可以得到。对于第 13.1.2 节的许多轮式车辆系统，存在可以有效解决 BVP 的操控方法，参见第 15.5 节。对线性系统也存在有效的解决方法，这将在第 15.2.2 节进行介绍。

如果有效地解决了 BVP，那么第 5 章任何基于采样的规划算法都可以适应性修改到有微分约束的情况。这通过使用图 14.10 中的模块作为 LPM 来实现。例如一个基于采样的路线图可以在经过 C 的最短路径的地方使用计算所得到的解。如果 BVP 不能有效解决，那么这种方法就变得不切实际，因为必须成千上万次地使用这种方法来建立一个路线图。如图 14.10 所示的一个有效模块能够消除与微分约束下规划有关的大多数复杂性。剩余的问题是：与没有微分约束情况下的最短路径比较而言，BVP 的解可能很长（例如 Dubins 车必须行驶多远才能向后稍微运动一点）。

对于第二种方法，假设求解 BVP 的开销很高。这种情况下规划方法应该尽可能避免对 BVP 进行求解。一些规划算法可能只要求 LPM 接近中间目标状态，这些状态对于某些系统来说更加简单。其他规划方法可能不要求 LPM 进行任何类型的连接。LPM 可能返回一个运动本原，该本原在搜索中有所进步，但是对它的设计并不是为了连接一个预定的状态。

这通常涉及递增规划方法，递增规划方法将在第 14.4 节进行介绍，它对第 5.4 节和第 5.5 节中的方法进行了扩展以处理微分约束。

14.3.4 微分约束下的一般框架

这里提出的框架是对第 5.4.1 节中的采样和搜索框架的直接扩展，它包括将第 5.5 节的内容扩展到允许在搜索图的刈幅中对任意点进行选择。这就用刈幅-点选择方法（SSM）代替了凸选择方法（VSM）。框架也就很自然地扩展了第 2.2.4 节的离散搜索框架。其中的步骤有：

（1）**初始化**：令 $G(V,E)$ 代表一个无方向的搜索图，它的顶点集合 V 包含 x_I 的一个顶点和 X_{free} 中其他可能的状态，边的集合 E 是空的。图可以被阐述为一个有刈幅 $S(G)$ 的拓扑图。

（2）**刈幅-点选择方法（SSM）**：为扩展而选择一个顶点 $x_{cur} \in S(G)$。

（3）**局部规划方法（LPM）**：生成一个运动本原 $\tilde{u}^p:[0,t_F] \rightarrow X_{free}$ 使得对于某 $x_r \in X_{free}$，有 $u(0)=x_{cur}$ 和 $u(t_F)=x_r$，x_r 可能是 G 中的顶点，也可能不是。使用系统模拟器、碰撞检测算法以及通过对相约束进行检测，必须验证 $\tilde{u}^p$ 是没有违规的。如果这一步失败，那么返回步骤（2）。

（4）**在图中插入一条边**：将 $\tilde{u}^p$ 插入 E。通过积分，$\tilde{u}^p$ 产生了一条从 x_{cur} 到 x_r 的状态轨迹。如果 x_r 不在 V 中，那么将它加入。如果对于某 $e \in E$，x_{cur} 处在一条边轨迹的内部，那么在 x_{cur} 引入一个新的顶点来分割 e。

（5）**对解进行检查**：确定 G 是否已经对一条解路径进行了编码。在一些应用中，可能容忍在状态轨迹中存在小的间隙。

（6）**返回步骤（2）**：迭代继续进行，除非找到问题的解或者某终止条件得到满足。在后一种情况下，算法报告失败。

以与第 5.4.1 节相同的方式应用一般框架可以得到单向、双向和多向搜索。搬运工问题扩展到微分约束下的运动规划。例如 bug 陷阱引起同样的困难，并且当树的数量增加时，很难协调搜索过程。

新的复杂性来自于 BVP（见图 14.11）。回忆第 14.1.1 节，对于大多数系统来说，减少规划过程中必须解决的 BVP 是非常重要的。假设很难精确连接到一个预定状态。图 14.11 显示了一个最好的情形，其中前向单向搜索用来进入一个很大的目标区域。在这种情况下，不需要解决任何 BVP。当目标区域减小时，问题就复杂了。图 14.11（b）显示了 X_G 是一个点 $\{x_G\}$ 这种极限情况。这要求规划算法至少解决一个 BVP。

图 14.11（c）显示后向单向搜索情况。这也就是将 BVP 移动到 x_I。因为 x_I 精确给定（没有“初始区域”），不可能像前向情况那样避免 BVP。如果算法产生了一个解 $\tilde{u}$，对于这个解来说 $x(0)$ 非常接近于 x_I 这种情况，并且如果 X_G 很大，那么有可能解决求解困难。可以从 x_I 对 $\tilde{u}$ 应用系统模拟器，而不是从 $x(0)$。已知 $\tilde{x}(x(0),\tilde{u})$ 是无违规的，并且 $\tilde{x}(x_I,\tilde{u})$ 可以运动到接近 $\tilde{x}(x(0),\tilde{u})$ 的地方。这要求 f 关于 x 的变化只产生很小的变化（一个小的 Lipschitz 常数隐含着这种含义），也就是对于 $\|x_I - x(0)\|$ 来说很小。一个问题是当时间增加

时两条轨迹上的点之间的差别通常也会增加。如果系统模拟器已经证实 $\tilde{x}(x_I,\tilde{u})$ 是无违规的，并且最后状态仍然处在 X_G 中，那么可以宣布得到一个解。

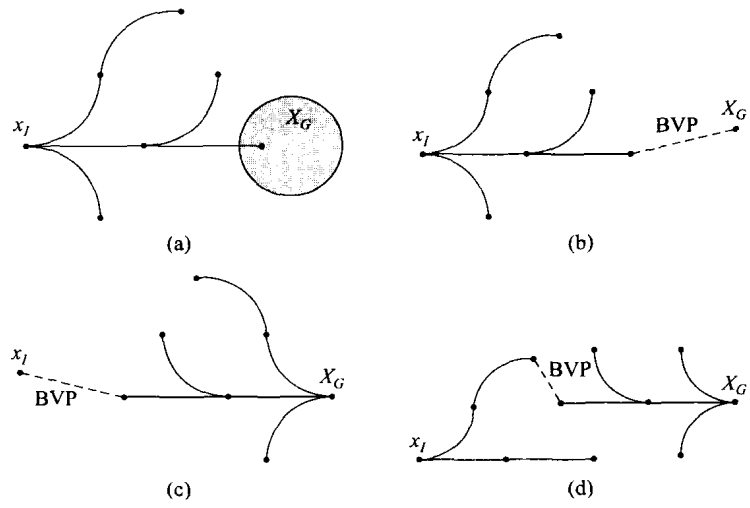


图 14.11　（a）避免进行 BVP 的前向、单向搜索；（b）精确到达目标会引起一个 BVP；（c）后向、单向搜索也会引起一个 BVP；（d）对于双向搜索，当连接树时会出现 BVP

对于双向搜索，必须在轨迹的中间某处求解一个 BVP，如图 14.11（d）所示。这使得确定两棵树是否连接上的问题更加复杂。如果目标区域很大，那么通过移动由后向树所产生轨迹的开始状态，也许能够消除轨迹中间的间隙。令 $\tilde{u}_1$ 和 $\tilde{u}_2$ 分别表示前向树和后向树产生的行动轨迹。假设它们的终止时间分别为 t_1 和 t_2。通过将 $\tilde{u}_2$ 的定义域从 $[0,t_2]$ 移到 $[t_1,t_1+t_2]$ 就可以连接行动轨迹产生一个函数 $\tilde{u}:[0,t_1+t_2]\rightarrow U$。如果 $t\leqslant t_1$，那么 $u(t)=u_1(t)$；否则 $u(t)=u_2(t-t_1)$。如果两条轨迹之间有间隙，那么必须用系统模拟器检查新的状态轨迹 $\tilde{x}(x_I,\tilde{u})$ 以确定它是否是无违规的以及是否终止在 X_G 中。因为生成了更多 BVP，所以多向搜索非常复杂。从原理上讲，有可能将上述思路扩展到连接一系列行动轨迹，这试图消除所有间隙。

考虑搜索图和可达性图之间的关系。在单向搜索中，搜索图总是可达性图的一个子集（假设精度很高且没有数值积分误差）。在前向搜索情况下，可达性图从 x_I 开始，而在后向搜索情况下，可达性图从 x_G 开始。在双向搜索情况下，有两个可达性图。来自于这两个图的顶点可能是完全重合，这是避免 BVP 的另外一种方法。然而遗憾的是这种情况非常少，除非有意地选择 x_I 和 x_G 使得出现这种情况。例如可以选择 x_G 的精确位置，因为已经知道它是来自于 x_I 的可达性图的一个顶点。对于多数系统来说很难如此。因此一般情况下，BVP 总会出现，因为可达性图没有公共顶点。在多向搜索中，大量的可达性图被探测，而每个可达性图的顶点与其他图的顶点不一致。

14.4　递增采样和搜索方法

第 14.3.4 节的一般框架将被具体化来为微分约束下的规划产生三种重要的方法。

14.4.1 在网格上进行搜索

本节按照第 5.4.2 节的思路，将栅格搜索技术适应性修改到运动规划。在目前情形下的困难在于选择一种离散化，它所产生的网格能够用第 2.2 节的任何搜索技术来搜索。本节主要是受到 kinodynamic 规划工作的启发[290,292,444]。

14.4.1.1 二重积分器网格

首先考虑例 13.3 的二重积分器。令 $\mathcal{C}=\mathcal{C}_{free}=\mathbb{R}$ 且 $\ddot{q}=u$。这对 $\mathbb{R}$ 中一个自由质点的运动进行了建模，如第 13.3.2 节所描述。相空间为 $X=\mathbb{R}^2$ 且 $x=(q,\dot{q})$。令 $U=[-1,1]$。如果令 $U=[-a_{\max},a_{\max}]$，那么即将介绍的思路也很容易推广到任意加速度界 $a_{\max}>0$；但是选择 $a_{\max}=1$ 以简化表达。

微分方程 $\ddot{q}=u$ 进行一次积分就产生：

$$\dot{q}(t)=\dot{q}(0)+ut \tag{14.21}$$

其中 $\dot{q}(0)$ 是一个初始速度。对式（14.21）进行积分，得到位置：

$$q(t)=q(0)+\dot{q}(0)t+\frac{1}{2}ut^2 \tag{14.22}$$

它用到两个初始条件 $q(0)$ 和 $\dot{q}(0)$。

存在一个离散时间模型，对于它来说可达性图落在一个网格上。令 $U_d=\{-1,0,1\}$ 且 Δt 为任意正实数就可以得到该模型。X 上的对应于 $u=-1$、$u=0$ 和 $u=1$ 的向量场如图 14.12 所示。在每个 Δt 上在这些场之间进行切换且进行积分就产生了如图 14.13 所示的可达性图。

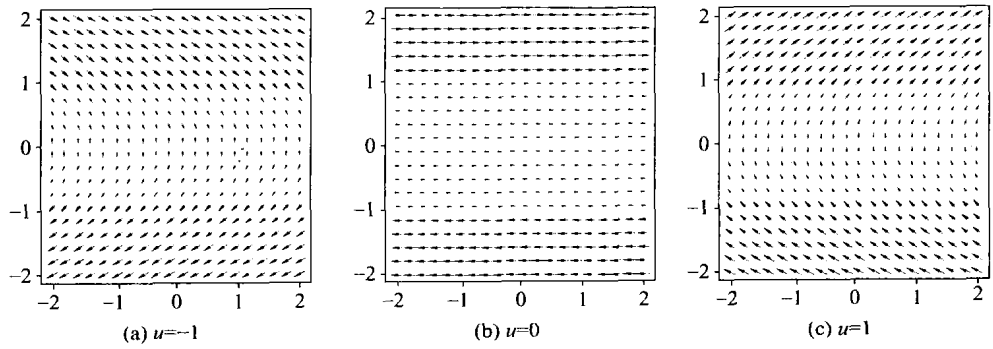


图 14.12 在每个 Δt 上在这些向量场之间进行切换就可以得到可达性图；中间的一个图产生水平相轨迹，其他的图产生抛物线

这产生了一个 $x_{k+1}=f_d(x_k,u_k)$ 形式的离散时间转移方程，其中 $u_k\in U_d$ 且 k 表示时间 $t=(k-1)\Delta t$。任何行动轨迹都可以作为一个行动序列来确定，例如一个 6 阶段行动序列可以用 $(-1,1,0,0,-1,1)$ 给出。从 $x_1=x(0)=(q_1,\dot{q}_1)$ 开始。在任意阶段 k 和对于任意行动序列，所产生的状态 $x_k=(q_k,\dot{q}_k)$ 可以表示为：

$$\begin{aligned} q_k &= q_k+\frac{1}{2}i(\Delta t)^2 \\ \dot{q}_k &= \dot{q}_1+j\Delta t \end{aligned} \tag{14.23}$$

其中 i 和 j 是可以从行动序列中计算而来的整数。因此任意行动序列产生了一个可以用平

面中的整数坐标 (i,j) 表示的状态。从 $x_1=(0,0)$ 开始，这形成了如图 14.13 所示的点格。网格是倾斜的（斜率为 1），因为变化速度要求某种运动。如果允许加速度无穷大，那么 $\dot{q}$ 能够瞬时发生改变，这对应于在 X 中垂直运动。就像在式（14.21）中所见，$\dot{q}$ 随时间线性变化。如果 $q\neq 0$，那么位形呈平方规律变化。如果 $u=0$，那么它线性变化，除了当 $\dot{q}=0$ 时；在 $\dot{q}=0$ 的情况下，不发生运动。

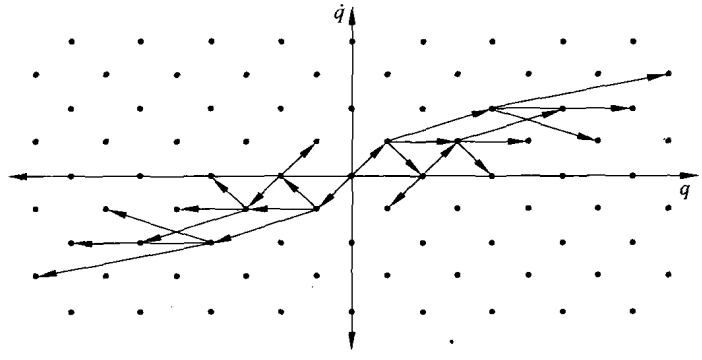


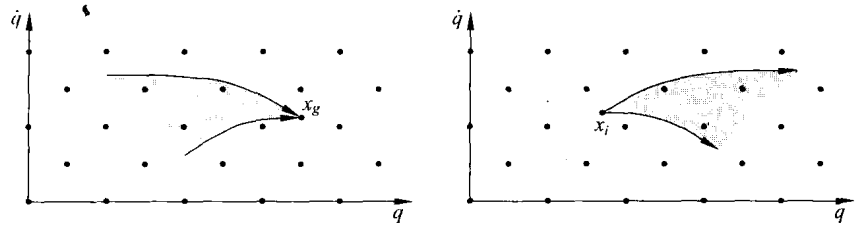
图 14.13　显示三个阶段以后，从原点的可达性图（当加速或者减速时真实状态轨迹实际上是抛物线）；得到一个网格，但是当 $|\dot{q}|$ 增加时，一个阶段内运动的距离也会增加

因为有漂移，所以这里的邻域结构与第 5.4.2 节中的不同。对于 $u=0$，假设有许多水平传送带，如果点在 q 轴的上方，那么传送带将这些点向右输送；如果这些点在 q 轴的下方，那么传送带将这些点向左输送（见图 14.12（b））。传送带的速度给定为 $\dot{q}$。如果 $u=0$，那么沿着 q 传送的距离为 $\dot{q}\Delta t$。如果 $\dot{q}>0$，那么在相平面中这引起向右的水平运动；如果 $\dot{q}<0$，那么这引起向左的水平运动。从图 14.13 可以观察到：当 $|\dot{q}|$ 增加时会产生更大的运动。如果 $\dot{q}=0$，那么没有水平运动发生。如果 $q\neq 0$，那么 $\dot{q}$ 坐标改变 $\pm\frac{1}{2}u(\Delta t)^2$。这种减速或者加速也会影响沿着 q 方向的位置。

对于大多数实际问题来说，对速度都有一个上限。令 $v_{\max}>0$ 为一个正常数，并假设 $|\dot{q}|\leq v_{\max}$，并且假设 C 是有界的（所有 $q\in C$ 都包含在 $\mathbb{R}$ 的一个区间内）。因为可达性图是一个网格，并且现在状态被限制在 $\mathbb{R}^2$ 的一个有界子集内，所以可达性图中顶点的数量是有限的。对于任意固定的 Δt，网格可以用第 2.2 节的任何算法来进行搜索。搜索开始于一个可达性图，该图的初始顶点为 x_I。近似时间最优的轨迹可以用广度优先搜索来得到（也可以采用 Dijkstra 算法，但是开销更大）。每次搜索失败时减小 Δt 一个固定因子，这样就可以得到分辨率完备性。就像第 5.4.2 节所提及的，不要求立刻构造整个栅格分辨率。采样可以逐步进行，用 union-find 算法[245,826]可以有效地更新连通性。稍后将介绍一种近似算法框架，它指出解离最优有多近，这是用算法的输入参数表达的。

回忆图 5.14（b）所阐述的连接格点的问题。如果目标区域 X_G 包含网格点，那么一定会准确到达目标。如果没有包含网格点，例如 X_G 为一个单点，那么需要一些附加工作来将目标状态连接到附近的网格点。这实际上对应一个 BVP，但是对于二重积分器来说，它容易解决。在 Δt 时间内从某状态 x_G 所能够到达的状态的集合处在一个锥中，如图 14.14(a) 所示。应用 U 中的一个常数行动，使得落在锥中的网格点很容易连接到 x_G。同样地，x_I 甚

至不必正好是一个网格点。因此很容易将 x_I 连接到一个网格点，得到一条到达 x_G 附近的一个点的轨迹，然后将给定点连接到 x_G。



(a) 从 x_G 的后向可达集　　(b) 从 x_I 的前向可达集

图 14.14　初始和目标状态可以连接到网格点，这些点在表示时间受限可达集 X 中的圆锥体内

14.4.1.2　扩展与其他考虑

二重积分器的可选网格　在 X 上可以构造多种网格。也可以对 U 和时间进行不同的离散化。如果只注重可行性，而不是最优性，那么灵活性就更大。因为 $\mathcal{C}=\mathbb{R}$，所以很难定义一个规避障碍物的问题，但是概念很快将推广到更高维。在这种情况下，发现一个连接某初始状态和目标状态的可行轨迹是我们主要关心的问题。但是如果 x_I 和 x_G 是零速状态，那么状态在 q 轴周围徘徊，并且速度太小以至于动量可以被忽略。这为尽可能减小运动时间的目的提供了某种动力，即使最终结果不是最优的。作为另外一种选择，初始状态和目标状态也可能不是零速的，在这种情况下，任何可行解都是令人满意的。例如为进行安全性分析，假设目标是倾覆一辆运动型汽车（SUV）。

为了得到如何构造网格的一种感性认识，回忆类似于传送带的例子。在 $\dot{q}$ 的各个值上水平排列采样点，这样设计一个网格。例如在 $\dot{q}$ 方向上这些可以均匀地间隔，如图 14.13 所示。假设状态处在一个传送带上。如果需要，通过应用一个非零行动持续某指定的时间，就可以产生运动到达另外一个传送带，记为 $\dot{q}'$。如果 $\dot{q}'>\dot{q}$，那么 $u>0$；否则 $u<0$。如果行动是常数，那么经过 $|\dot{q}'-\dot{q}|/u$ 时间以后，状态将到达 $\dot{q}'$。到达时，在传送带上的位置 q 可能不在某个采样点上。这没有问题，因为可以应用行动 $u=0$ 直到状态漂移到下一个采样点。另外一种方法是从 U 中选择行动，行动直接驱动到达它的前向、时间受限可达集中的一个网格点。回忆图 14.14，锥可以被放置在一个网格点上来定位其他网格点，它们是某时间区间上应用 U 中一个常数行动所能够到达的点。

回忆图 14.13，当 $|\dot{q}|$ 增加时，在 Δt 时间内运动的距离更远。实际上可能不希望这种行为，因为更快速度下的分辨率很差。当 $|\dot{q}|$ 增加时，这可以通过将传送带很近地放置在一起来进行补偿。当速度增加时，需要更短的时间区间来改变传送带，并且所运动的距离可以大致保持相同。这符合在更快的速度下需要更快的反应时间这种直觉。

也可以产生一个多分辨率版本[819]。实际上目前所考虑的简单问题都能够没有近似误差地得到解决[750]，但是这里之所以介绍基于网格的方法是因为它可以扩展到更加棘手的问题，就像下面将要阐述的情况。

多个独立的二重积分器　现在考虑推广到 n 个积分器的向量。在这种情况下，$\mathcal{C}=\mathbb{R}^n$ 且每个 $q\in\mathcal{C}$ 是一个 n 维向量。有 n 个行动变量和 n 个 $\ddot{q}_i=u_i$ 形式的二重积分器。每个变量的行动空间是 $U_i=[-1,1]$（也可以用任意加速度界）。相空间 X 是 $\mathbb{R}^{2n}$，并且每个点为

$x=(q_1,\cdots,q_n,\dot{q}_1,\cdots,\dot{q}_n)$。第 i 个二重积分器产生相空间转移方程的两个标量方程，它们是 $\dot{x}_i=x_{n+i}$ 和 $\dot{x}_{n+i}=u_i$。

即使有 n 个二重积分器，它们也可以在状态方程中进行解耦。一个积分器的相不依赖于其他积分器的相。因此目前所表达的思路可以用一种简单的方式进行扩展以得到 $\mathbb{R}^{2n}$ 上的一个网格。每个行动是一个 n 维向量 u。每个 U_i 离散化后产生值−1、0、1。从任意网格点发出 3^n 条边，其中对于所有 i，有 $\dot{q}_i\neq 0$。对于 $\dot{q}_i=0$ 的任意二重积分器来说，只有两种选择，因为 $u_i=0$ 不产生运动。对于 i 从 1 到 n，可达性图到 (x_i,x_{n+i}) 的投影看起来非常像图 14.13，它刻画了第 i 个积分器的行为。

标准搜索算法可以应用到 $\mathbb{R}^{2n}$ 上的网格。广度优先搜索产生近似时间最优解。通过包围 X 并允许 Δt 收敛到零就可以得到分辨率完备性。既然有更多维数，那么从 X 中可以删除一个复杂的障碍区域 X_{obs}。那么遍历每条边就要求沿着图的每条边进行碰撞检测。注意到状态轨迹是线性的或者抛物线的。那么不需要进行数值积分，因为式（14.22）已经给出了状态轨迹的封闭表达式。

无约束机械系统　为一个一般情形的全驱动机械系统可以得到一个网格，例如它可以包括大多数机械臂。回忆式（13.4），如果对于一个固定的 q，$U(q)$ 可以定义为 f 的像，那么 $\dot{q}=f(q,u)$ 形式的任何系统也可以表示为 $\dot{q}=u$。使用 f 的主要目的在于使得比较容易指定一个固定的行动空间 U，对于每个 $q\in C$，U 映射到不同的切空间。

关于 $\ddot{q}=h(q,\dot{q},u)$ 形式的方程也可以观察到相似的情形，其中 $u\in U$ 且 U 是 $\mathbb{R}^n$ 的一个开子集。第 13.3 节和第 13.4 节中一般的无约束机械系统可以得到这种形式。例如式（13.148）用于表达开链机械臂的动力学。通过直接指定可允许加速度的集合，这种方程可以表示为 $\ddot{q}=u'$。每个 u 将会映射到下式给出的一个行动空间中一个新的行动 u'：

$$U'(q,\dot{q})=\left\{\ddot{q}\in\mathbb{R}^n \,\middle|\, \exists u\in U,\ddot{q}=h(q,\dot{q},u)\right\} \tag{14.24}$$

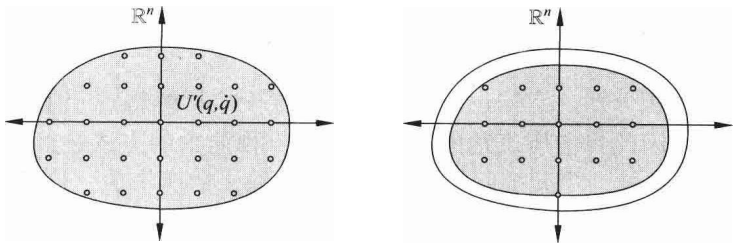
其中 $q\in C$ 和 $\dot{q}\in\mathbb{R}^n$。

每个 $u'\in U'(q,\dot{q})$ 直接表达 $\mathbb{R}^n$ 中的一个加速度向量。因此使用 $u'\in U(q,\dot{q})$，用 h 表示的原始方程现在可以记为 $\ddot{q}=u'$。在这种新的形式下，它看起来像多个独立的二重积分器。主要的不同之处在于：

（1）集合 $U'(q,\dot{q})$ 可以描述 $\mathbb{R}^n$ 中的一个复杂区域，然而在真正二重积分器的情况下 U 是一个以原点为中心的方体。

（2）集合 $U'(q,\dot{q})$ 关于 q 和 $\dot{q}$ 发生变化。对于时间采样区间 Δt 上的这种变化必须给予特别的关注。在真正二重积分器的情况下 U 是固定的。

在 $\mathbb{R}^n$ 上进行栅格采样，并在可达性图中为每个落入 $U'(q,\dot{q})$ 的栅格点构造一条边，这样就可以处理第一个不同（见图 14.15（a））。随 Δt 提高栅格分辨率可以得到分辨率完备性。为了讨论第二个问题，将 $U'(q(t),\dot{q}(t))$ 考虑为 $\mathbb{R}^n$ 中随时间运动的一个形状。将 u' 选择在接近 $U'(q(t),\dot{q}(t))$ 的边界的地方是很危险的，因为当 t 增加时，u' 可能落在新的行动集合的外面。通常可能得到关于 $U'(q(t),\dot{q}(t))$ 的边界能够改变多快的界限（例如这可以通过对 h 关于 q 和 $\dot{q}$ 进行微分）。基于这个界限，$U'(q,\dot{q})$ 的边界附近的一个薄层可以不予考虑，这样的目的是确保所有想采取的行动在整个 Δt 时间内都在 $U'(q(t),\dot{q}(t))$ 中，见图 14.15（b）。



(a) (b)

图 14.15 （a）新的行动集合$U'(q,\dot{q})$和基于栅格的采样；（b）减小某安全边缘来说明它随时间的变化

应用这些思路将近似算法框架扩展到开链机械臂的情况，其中 h 由式（13.148）给定。假设 U 是一个轴向矩形，它通常也是如此，因为每个 u_i 的界限对应于每个电动机的极限扭矩。如果 q 和 $\dot{q}$ 固定，那么式（13.140）应用一个线性变换来从 u 得到 $\ddot{q}$。矩形通常被剪切成一个平行六面体（一个平行四边形的 n 维扩展）。回忆第 3.5 节或者线性代数中的这种变换。

近似算法框架 本节所述的网格是文献[292]介绍的，它在 NP 复杂度问题的严格近似算法框架中分析 Kinodynamic 规划问题[768]。假设有两个或者三个独立的二重积分器。分析表明：计算的解在如下意义上是近似最优的。令 c_0 和 c_1 为两个正常数，它们定义了一个函数：

$$\delta(c_0,c_1)(\dot{q}) = c_0 + c_1\|\dot{q}\| \tag{14.25}$$

令 t_F 表示应用终止行动的时刻。一个状态轨迹被称为 $\delta(c_0,c_1)$-安全的，如果对于所有 $t\in[0,t_F]$，半径为 $\delta(c_0,c_1)(\dot{q})$、圆心在 $q(t)$ 的一个球在 $\mathcal{W}$ 中不会引起与障碍物的碰撞。注意到球的半径随速度线性增长。可以将机器人想象为一个半径由速度来确定的圆盘。给定 x_I、x_G、c_0 和 c_1（只允许点目标区域）。假设对于一个给定的问题，存在一个 $\delta(c_0,c_1)$-安全的状态轨迹（对任意 $\tilde{u}\in\mathcal{U}$ 进行积分产生），该轨迹在时间 t_{opt} 以后终止在 x_G 中。通过选择适宜的 Δt（由文献[292]中的公式给出），将广度优先搜索应用于可达性网格将会发现一个 $(1-\varepsilon)\delta(c_0,c_1)$-安全轨迹，它最多花费 $(1-\varepsilon)t_{opt}$ 时间将 x_I 近似连接到 x_G（在 X 中接近的程度依赖于 ε），并且算法的运行时间是关于 $1/\varepsilon$ 和用来定义多边形障碍物的本原的数量的多项式[1]。在分析中一个关键的步骤是显示能够通过只使用 U_d 中的行动并在 Δt 上保持它不变就可以对任意状态轨迹进行跟踪。但这并不一定意味着计算所得到的解接近于真实最优值，当它在 X 中运动时（只是接近执行时间）。因此算法可从包含真实最优轨迹的一个同伦类中给出一个解。这种分析在文献[290,444]中被扩展到开链机械臂的一般情况。

后向和双向情况 对本节提出的概念都有完美的对称性。对时间进行后向积分就可以得到与图 14.13 中相似的一个可达性网格。这指出了行动序列和相应的初始状态，通过它们可以到达一个固定的状态。应用逆向思路时并不要求系统是对称的。假定在两个方向上都存在图，那么可以进行双向搜索。通过使用前向和后向时间受限可达性锥，初始状态和目标状态可以连接到一个公共的网格上。

1 一个技术要点：它实际上只是 $a_{\max}$、$v_{\max}$、c_0、c_1 和 $\mathcal{W}$ 中包围体宽度的伪多项式。这意味着如果这些参数的表示被当作有固定尺寸大小，那么运行时间是多项式，但它不是真实表示它所需要的比特数的多项式。

欠驱动和非完整性系统　因为有欠驱动或者其他约束，所以一些有趣的系统并不能用带有 n 个独立行动变量的 $\ddot{q} = h(q,\dot{q},u)$ 的形式来表示。例如第 13.1.2 节中的模型是欠驱动的和非完整的。在这种情况下，并不能直接将方程转换成二重积分器的向量，因为 $U(q,\dot{q})$ 的维数小于 C 的维数 n。这就不可能使用 $U(q,\dot{q})$ 的基于栅格的采样。尽管如此，在许多情况下仍然有可能用某种很巧妙的方法对系统进行离散化来得到一个网格。如果能够如此，那么就可以产生基于经典搜索算法的一个分辨率完备的方法。如果 X 是有界的（或者应用相约束以后得到一个有界区域），那么可在一个有限图上进行搜索。如果失败，那么可用普通的方式提高分辨率以最终得到分辨率完备性。就像第 14.2.2 节所述，对于一大类非完整系统来说，得到这样的网格是可能的[765]。下面介绍一种方法，它处理的可达性图不是网格。

14.4.2　结合状态空间离散化

如果可达性图不是一个网格，这种情况对欠驱动和非完整系统来说是非常典型的，那么可以用状态空间离散化来迫使其行为像一个网格。如果没有微分约束，那么很容易迫使路径沿着一个网格运动，如第 7.7.1 节所述。在微分约束情况下，不能迫使状态跟踪阶梯路径。不对 X 进行采样并迫使轨迹访问特定的点，取而代之的是将 X 分拆成小的胞腔，在这些胞腔中只有一个顶点。如果在某有界区域中搜索图有无穷多个顶点，这能够防止系统搜索算法无限进行下去。例如 Dubins 车，如果 u 固定为一个正数，那么得到一个圆上无穷多个顶点，如第 14.2.2 节所述。本节的思路主要是受到 Barraquand-Latombe 动态规划方法的启发[74]，该方法已经应用于第 13.1.2 节中的模型。但是在目前情形下，该方法将被进一步推广。这里并不要求最优性（如果需要，也可以提出这种要求）。

将 X 剖分成胞腔　开始时，将 X 剖分为胞腔的集合，而不考虑碰撞检测。假设 X 是 $\mathbb{R}^n$ 的一个 n 维矩形子集。一般情况下，如果 X 是一个平滑流形，那么可以在一个坐标邻域中定义矩形子集。如果需要，可以使用等同来遵循 X 的拓扑，但是在边界上需要坐标变化来表达速度（回忆第 8.3 节）。

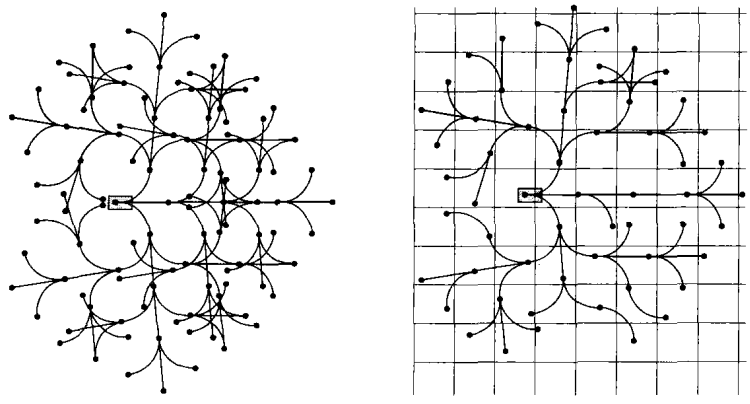
最平常的胞腔剖分是通过对每个坐标进行量化来将 X 分割为相同尺寸的 n 维方体来得到的。这称为方体分拆。一般情况下，假设 X 被分拆为 n 维胞腔的一个集合 $\mathcal{D}$。令 $D \in \mathcal{D}$ 表示一个胞腔，它是 X 的一个子集。这里假设所有胞腔的维数都是 n。在方体的情况下，这意味着方体之间公共边界上的点被声明只属于其中一个方体（因此胞腔可以是开的、闭的或者两者都不是）。

注意到 X 被分拆成胞腔，而不是第 6 章所期望的 X_{free}。这意味着当定义 $\mathcal{D}$ 时，首先忽略了碰撞检测以及 X 上的其他约束。提前定义胞腔，就像第 5.4.2 节中声明栅格一样。在方体分拆的情况下，在每个坐标轴的量化上，胞腔是已知的。

搜索　算法适合第 14.3.4 节的框架。应用任何系统搜索算法，从 x_I 递增地构造一个搜索图。假设系统已经用某种方法进行了离散化。通常采用第 14.2.2 节的离散时间模型，这产生一个固定的 Δt 和行动的一个有限集合 U_d。

在第 2.2.1 节的基本搜索算法中，重要的是跟踪哪些顶点已经被探测过。将该思路应用于胞腔，而不是应用于顶点。一个胞腔 D 将被称为已访问过的，如果目前已经构造的搜索图包含 D 中的一个顶点；否则，D 将被称为未访问的。初始情况下，只有包含 x_I 的胞腔被

标注为已访问过的。所有其他的胞腔初始化为未访问过的。这些标号在搜索过程中用来对可达性图进行裁剪，如图 14.16 所示。



(a) (b)

图 14.16 （a）Dubins 车的一个稠密可达性图的前四个阶段；（b）一个可能的搜索图，它是通过使每个胞腔最多只有一个顶点而得到的；裁剪掉一些分支；在这个简单例子中，沿着 θ 轴没有胞腔划分

基本算法如图 14.17 所示。令 Q 代表一个优先级队列，其中的元素是搜索图的顶点。如果需要对代价泛函进行优化，那么 Q 可以用沿着从 x_I 到 x 构造的路径所累积的代价来进行排序。有多种方法表示代价。它可能简单地代表时间（Δt 步数），或者它可以对行动发生改变的次数进行计数。算法在探索过程中会遇到新的候选顶点。如果它们处在被标为未访问过的胞腔中且是无违规的，那么它们将被存储在搜索图中并被放入 Q。当遇到这种胞腔时，那么就将它们标注为已访问过的。REACHED 函数将生成无违规的轨迹片段的集合。在离散时间模型下，这意味着在 Δt 上应用每个 $u \in U_d$ 并报告所到达的不违背约束的那些状态（包括避免碰撞）。

```
CELL-BASED_SEARCH (x_I,x_G)
1    Q.insert (x_I);
2    G.init (x_I);
3    while Q≠0 and x_G is unvisited
4        x_cur→Q.pop();
5        for each (ũ_I , x) ∈ REACHED (x_cur)
6            if x is unvistted
7                Q.insert (x);
8                G.add_vertex(x);
9                G.add_edge(ũ_I );
10               Mark cell that contains x as visited;
11   Return G;
```

图 14.17 用 X 的一个胞腔剖分来进行搜索

通常情况下，如果 X_G 相对于胞腔的尺寸来说非常小，那么就会出现 BVP 问题。如果 X_G 大到足以包括整个胞腔，那么就可以避免这种问题。如果 x_G 是一个单点，那么只可能近似到达 x_G。因此算法必须接受以一定的容差接近于 x_G。这可以通过定义更大的 X_G 来进行建

模，因此并不明确提及这个容差。

维持胞腔　维持胞腔的方法有许多种。需要进行的主要操作是点定位[266]：确定哪个胞腔包含一个给定的状态。文献[74]中最初的方法是预先分配一个 n 维数组。甚至可以提前进行碰撞检测。至少包含 X_{obs} 中的一个点的胞腔可以被标注为已占用（occupied）的。这就规避了那些包含碰撞位形的胞腔，而不必调用碰撞检测模块。对于固定的维数，这能够发现正确的胞腔并在固定的时间内更新标注。非常遗憾，需要的空间是维数的指数形式。

另外一种方法是使用一个表格来维持被标注为已访问过的那些胞腔的集合。如果最优性不重要并且期望在到达大多数胞腔之前就得到解，那么这可能特别有价值。不需要明确存储多维数组就可以有效地解决点定位问题。

假设不用立方剖分。一般的方法是将 $\mathcal{D}$ 定义为 X 中 m 个采样点 $(p_1,\cdots,p_m)$ 的集合 P 的 Voronoi 区域。每个胞腔的“名字”对应于 P 中唯一一个采样。用 X 上某预先确定的度量，包含某 $x \in X$ 的胞腔被定义为 P 中最邻近的采样。作为一种特殊情况，立方剖分基于一个 Sukharev 栅格来定义胞腔（回忆图 5.5（a））。如果 X 的维数不太高，那么在与 P 中点数呈现近似对数关系的时间内可以用有效的最邻近方案来确定合适的胞腔（实际上应该可用第 5.5.2 节介绍的 Kd-树）。为了维持一个立方剖分，这种方法将很麻烦，但是它可以用于任意采样集合 P。如果对于一个给定的 P 没有发现解，那么可增加采样点来改进分拆。在确保下面将要讨论的分辨率的同时，这使得可以使用任意稠密序列来引导对 X 的探索。

分辨率问题　在使用状态离散化中，一个主要的复杂性是由于需要对三个空间进行采样：时间、行动空间和状态空间。假设使用离散时间模型。如果需要得到最优解，那么应该使用非常小的胞腔（例如每轴 50～100）。这就限制它在较多维状态空间中的应用。时间区间 Δt 也应该被划分得很小，但是如果相对于胞腔尺寸来说太小，那么它就不可能离开一个胞腔。如果只要求可行性，那么可以用大的胞腔，并且 Δt 也要相应增加。U 的一个量化有可能错过解，特别是如果 Δt 很大时。当 Δt 降低时，U_d 中采样的点数就不太重要了。

为了得到分辨率完备性，当每次搜索失败时应该增加采样点。每次搜索开始时，三个空间中至少有一个空间的采样离散度应该减小。可能：（1）时间区间 Δt 减小；（2）更多行动加到 U_d；（3）更多点加到 P 中以减小胞腔的尺寸。如果三个空间的离散度接近于零，并且如果 X_G 包含 X_{free} 的一个开子集，那么就得到分辨率完备性。如果 X_G 只是一个点，那么必须容忍解存在某容差 $\varepsilon > 0$。

分辨率完备性假设 f 有有界导数或者至少满足 Lipschitz 条件见式（14.11）。实际的收敛速率主要受三个因素影响：（1）f 关于 u 和 x 的变化而变化的速率（用 Lipschitz 常数来刻画）；（2）所要求的对 X_{free} 中的狭窄区域进行遍历；（3）系统的可控性。最后一个条件将会在第 15.4 节的非完整系统中进一步学习。例如考虑 Dubins 车做半径非常大的一个 U 形调头，如图 14.18 所示。在调头时要求精确转弯，这可能需要许多阶段以前所选择的行动。Dubins 汽车模型不允许用“之”字形机动（例如平行移位停车）来对位形进行局部调整。

后向和双向版本　通常可以产生这种方法的后向和双向版本。如果 X_G 很大（或者目标容差很大）并且解决 BVP 的开销很大，那么其后向版本似乎也不会令人满意。前向进行是可取的，因为这样可以避免 BVP。

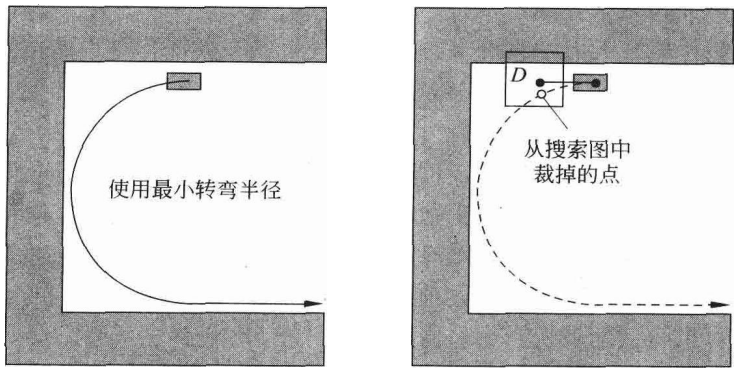


图 14.18 （a）如果 Dubins 车向左急转弯，那么它就能够调头；（b）非常遗憾，所需要的顶点将被裁剪掉，因为沿着所要求的轨迹的一个胞腔已经有一个顶点了；这就解释了丢失一个可能的行动是如何在许多阶段以后引起严重问题的

对于一个双向算法，胞腔的相同集合 $\mathcal{D}$ 可以用在两棵树上。如果两棵树到达同一个胞腔，那么认为问题已得到解决，但是必须仔细确保剩下的 BVP 能够得到解决。有可能发现一条连接基于初始状态的树的一个顶点到基于目标状态的树的一个顶点的行动轨迹片段。两棵树接近到一个容差范围内也是可以接受的。

14.4.3 RDT 方法

快速探索稠密树（RDT）的方法族包括 RRT，它不需要维持一个网格。RDT 最初是为了处理微分约束，即使在大多数情况下它被应用于钢琴搬运工问题。本节将第 5.5 节的思路从 $\mathcal{C}$ 扩展到 X 并加入微分约束。第 14.4 节介绍的方法产生近似最优解，如果用动态规划来对图进行搜索且分辨率足够高。比较而言，RDT 的目的是只返回可行解，甚至在提高分辨率的情况下。它们通常用较少的采样点就能够成功地产生解轨迹。这种性能上的提高的部分原因来自于不关注最优性问题。

令 α 表示 X 中采样点的一个无穷、稠密序列。令 $\rho: X \times X \rightarrow [0, \infty)$ 表示 X 上的一个距离函数，它可能是一个度量，也可能不是。距离函数可能不是对称的，其中 $\rho(x_1, x_2)$ 代表从 x_1 到 x_2 的有向距离。

目前的 RDT 是一个搜索图，因此可以将其阐述为某离散化模型下可达性图的一个子图。为方便起见，首先假设使用第 14.2.2 节的离散时间模型，它产生一个有限行动集合 U_d 和一个固定时间区间 Δt。运动本原的集合 $\mathcal{U}^p$ 是 $0 \sim \Delta t$ 时间内 $u \in U_d$ 保持固定的所有行动轨迹。更一般的情况将在稍后进行讨论。

并行于第 5.5.1 节，首先定义在没有障碍物情况下的 RDT。因此令 $X_{free} = X$。算法的构造如图 14.19 所示；将它与图 5.16 进行比较，在那里是为钢琴搬运工问题而在 $\mathcal{C}$ 上进行的。用 $\mathcal{G}$ 表示的 RDT 初始化为某 $x_0 \in X$ 处的一个顶点。在每次迭代中，给 $\mathcal{G}$ 加入一条新的边和一个顶点。第 3 行用 ρ 来选择 x_n，这是 $\mathcal{G}$ 的刈幅中距离 $\alpha(i)$ 最近的点。在第 5.5 节介绍的 RDT 算法中，α 的每个采样点都变成一个顶点。由于 BVP 和 $\mathcal{U}^p$ 中特定的运动本原的

缘故，很难或者不可能精确到达 $\alpha(i)$。因此第 4 行调用一个 LPM 来确定一个本原 $\tilde{u}^p \in \mathcal{U}^p$，该本原从 x_n 进行积分得到一个新的状态 x_r。结果如图 14.20 所示。对于 $\mathcal{U}^p$ 代表离散时间模型这种默认情况而言，将通过在 Δt 时间上应用所有 $u \in U$ 并选择最小化 $\rho(x_r, \alpha(i))$ 的行动。一个附加的约束是如果在以前的迭代中已经选择了 x_n，那么 $\tilde{u}^p$ 必须是以前从 x_n 没有尝试过的一个运动本原；否则，在 $\mathcal{G}$ 中将产生两条完全相同的边，或者将时间浪费在对可达性图的边中已经明确知道处在碰撞中的边还进行碰撞检测。其余的步骤从 x_n 增加新的顶点和边。如果 x_n 被包含在一条边 e 所产生的轨迹中，那么 e 将被分割，如第 5.5.1 节所述。

```
SIMPLE_RDT_WITH_DIFFERENTIAL_CONSTRAINTS (x0)
1    G.init(x0)
2    for i=1 to k do
3        xn←NEAREST (S(G), α(i));
4        (ũ^p, xr)←LOCAL_PLANNER (xn,α(i));
5        G.add_vertex (xr);
6        G.add_edge (ũ^p);
```

图 14.19　将基本 RDT 算法扩展到处理微分约束；与图 5.16 进行比较，一个 LPM 计算 x_r，它成为新的顶点，而不是 $\alpha(i)$；在某些应用中，第 4 行可能会失败，在这种情况下，第 5 行和第 6 行将被跳过

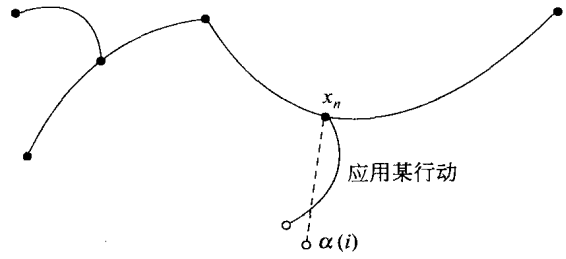


图 14.20　如果最近点 S 处在与一条边有关的状态轨迹片中，那么边将被分割为两段，新的顶点被插入 $\mathcal{G}$

有效发现最近点　对于微分约束下的 RDT，会再次出现第 5.5.2 节的问题。实际上，问题会更加复杂，因为 $\mathcal{G}$ 中的边通常会被弯曲。这就妨碍使用简单的点-片段距离检验，并且通常不知道状态轨迹的准确表示。取而代之的是用系统模拟器进行数字逼近。由于这些原因，最好是使用确定刈幅中的最近点的近似方法，它是对第 5.5.2 节中方法的直接扩展，回忆图 5.22。如果应用运动本原所产生的状态轨迹在 X_{free} 中行进很远，那么可以插入中间顶点。如果维数足够低（例如小于 20），那么可以用有效的最邻近算法（第 5.5.2 节）来弥补维持中间顶点所付出的代价。

处理障碍物　现在假设 $X_{obs} \neq 0$。在第 5.5.1 节中，延伸 RDT 直到正好在障碍物前到达一个停止位形 q_s。在微分约束下有两个新的复杂性。第一个是使用运动本原。如果 Δt 很小，那么在许多情况下在到达边界之前时间已经被耗尽。通过使用一个大的 Δt，然后只选取轨迹的无违规部分，这样就可以缓解这个问题。在这种情况下，轨迹可能被修剪以避免多次遇到 $\alpha(i)$。第二个复杂性是由于 X_{ric} 的缘故。如果动量很大，那么将树牵引到离障碍物尽可能近的地方将会增加 RDT 被陷的可能性。将会经常选择接近于障碍物的顶点，因为

它们有很大的 Voronoi 区域，但是它们不可能进一步延伸。在钢琴搬运工问题中，该问题并不明显，因为树很容易沿着边界。在多数试验工作中，最好是只行进通往边界的部分路程。

基于树的规划器 可以用第 5.5 节中的方法来构造规划算法。有前向、后向和双向版本。新的问题主要在于本节其他基于采样的方法已经遭受到的 BVP。如果对附近状态进行连接的代价非常高或者是不可能的，那么也会产生正常的复杂性。如果 X_G 包含一个相当大的开集合，那么一个适中偏离目标的前向、单树规划器能够良好地运行，同时还能够避免进行 BVP。但是如果 X_G 只是一个点，那么在声明得到解以前，必须设置一个容差来描述关于 RDT 须到达离目标多么近的地方。对于有漂移的系统，当容差减小时，搜索时间通常显著增加。

在许多情况下，双向搜索具有很大的性能优势，但是当试图连接两个树时同样存在 BVP。一种可能性是将容差设置得非常小，然后连接两个行动轨迹，如第 14.3.4 节所述。如果成功，那么规划算法就终止。然而遗憾的是，它的性能同样依赖于容差，特别是在漂移非常大的情况下。最新的研究表明：使用一个带有大的连接容差的双向 RDT，然后用有效的变分技术来使两棵树之间缩小间隙，这样能够显著地提高其性能[200,579]。非常遗憾，变分技术并不有效，因为本质上它们必须在轨迹空间中用梯度下降法来解决 BVP，参见第 14.7 节。

距离函数问题 RDT 构造算法受到距离函数 ρ 的影响。对于应用于钢琴搬运工问题的 RDT 来说也是如此，但是在有微分约束的情况下设计一个好的度量显得更加关键和更具挑战性。如例 5.3 给出的度量对于测量 Dubins 车两个位形之间的距离来说是不合适的。一个更合适的度量是使用从 q 到 q' 最短路径的长度(这种长度很容易进行计算，参见第 15.5 节)。即使对于涉及动力学和障碍物的汽车模型，这种度量也比例 5.3 中所介绍的更适于进行位形比较。

尽管用加权欧氏度量可以解决许多有挑战性的问题[614]，但是利用系统的独特性质能够显著提高整体性能。这个问题似乎与第 5.4.3 节随机势场规划器势函数的选择很相似。但是因为 RDT 接近于 $\alpha(i)$ 中许多不同的采样点，而不是只关注目标，所以性能的下降一般来说不像一个势场规划器的局部最小值问题那样严重。在 RDT 中有更多机会逃离陷阱。不能作为势函数的许多度量在基于 RDT 的规划器中却有非常好的性能。

如第 14.3 节所述，理想的距离函数是使用最优尚需代价，在这里用 ρ^* 表示。当然计算 ρ^* 至少与解决运动规划问题一样困难。因此这种思路并不实用。但是它通常在考虑 ρ^* 时非常有用，因为当 RDT 中所使用的度量 ρ 偏离 ρ^* 时，基于 RDT 的规划器的性能通常会恶化。即使忽略障碍物，对 ρ^* 进行粗略近似通常也能够导致性能的很大提高。文献[365]给出了这种情况的一个极好的例子，其中为用图 14.8 机动自动机模型所描述的自主飞行直升机，用值迭代来计算没有障碍物情况下的最优尚需代价。

确保分辨率完备性 假设使用离散时间模型。如果 α 在 X 中是稠密的，那么每个 RDT 顶点在构造以后，它将被访问可数无限多次。如果确保从同一个顶点对同一个运动本原从来不会应用两次，那么所有可用的运动本原都将被尝试到。这就确保了在一个固定的 Δt 时间上，整个可达性图都被探索过。因为可达性图不一定是有限的，所以很难得到分辨率完

备性。通过在执行过程中周期性地改变 Δt，并且在后续的迭代中使用越来越小的 Δt，这样就可以应用图 14.7 所描述的方案。若 U 是有限的，那么可以对 U_d 进行细分。这产生一个分辨率完备的 RDT。

设计良好运动本原　到目前只考虑了离散时间模型。尽管它非常简单和普通，但是通常还有更好的运动本原可以使用。对于一个特殊系统，在没有障碍物的情况下，可以离线设计一个好的轨迹族，然后将它们作为运动本原用于构造 RDT。如果可能，基于试图利用系统所具有的任何独特结构或者已经为之所设计的技术来仔细分析系统，这样做是非常重要的。对于一个车辆的运动规划，对称性使得可以应用来自不同状态的本原。例如在直升机飞行中，航向角和特定位置（除非它接近地面）可能不太重要。为一个航向角和位置所设计的轨迹族对其他角度和位置也应该可用。

使用更加复杂的运动本原可能会增加 LPM 的负担。在某些情况下，一个简单的控制律（如 PID[51]）可能就够用了。理想情况下，LPM 的行为应该是一个好的操控方法，它可以用第 15 章的方法来得到。重要的是注意到 RDT 解决问题的能力并不取决于此。如果有好的运动本原和 LPM 中好的操控方法，那么它就能够极大地提高系统性能。这其中的主要原因在于一旦这种情况发生(除了漂移所带来的不利影响)，微分约束的困难就已经得到克服。尽管好的运动本原能够在实际应用中提高性能，但是这也增加了保证分辨率完备性的困难。

14.4.4　其他方法

可以对第 5 章中的任何方法进行扩展来处理微分约束。本节将简单介绍几种方法。

随机化势场　第 5.4.3 节的随机化势场方法很容易被修改到处理微分约束，并不是向各个方向运动来减小势值，而是应用运动本原并对本原进行积分来试图减小势值。例如在离散时间模型下，可以在 Δt 上应用每个 $u \in U_d$，使得下一个状态有最低势值的那个 u 将被选择作为下降的一部分。当这种行动不存在时可以尝试随机游走，但是向任意方向运动是不可能的。那么可以采取随机行动。微分约束下的方法存在的主要问题是：（1）很难设计一个好的势函数；（2）随机行动不一定能够提供类似于随机游走所提供的运动。第 15.1.2 节将讨论 Lyapunov 函数，在存在微分约束的情况下（但是通常忽略障碍物），它是一个好的势函数。在随机游走的地方，可以用诸如 RDT 等其他规划方法来尝试逃离局部最小点。

基于树的其他规划器　基于树的其他一些规划器也可以被扩展到处理微分约束。例如为文献[469]中出现的航天器，第 5.4.4 节的可扩张空间规划器可以扩展到 kinodynamic 规划。对微分约束下的规划，最近出现了一种新的基于树的方法，它被称为有向路径细分树（path-directed subdivision tree）[575]。该方法将通过在刈幅中随机选择点、应用随机行动、使用空间分拆数据结构来对探索进行控制。

基于采样的路线图规划器　在通常情况下构造基于采样的路线图是很困难的，除非 BVP 能够得到有效解决。第 15.5 节的操控方法可以服务于该目的[934,861]。从原理上看，可以使用第 14.4 节中的任何单疑问方法；但是多次使用它们所付出的开销太大，而在路线图构造算法中要求多次使用它们。

14.5 微分约束条件下的反馈规划

14.5.1 问题的定义

表述 14.1 假设不必进行反馈。如果给定初始状态，那么解的形式是一个行动轨迹，轨迹的积分在 X_{free} 之内产生一条以时间为参数的路径。这将第 4.3.1 节的钢琴搬运工问题扩展到包括相空间和微分约束。现在假设要求反馈。理由可能是没有给定初始状态或者由于干扰或系统模型误差等使得规划的执行是不可预测的。

很容易将第 8 章的反馈运动规划框架扩展到处理微分约束。将表述 8.2 和表述 14.1 进行比较。对表述 8.2 进行调整就可以得到微分约束下的反馈运动规划：

（1）在表述 8.2 中，$X=C_{free}$，定义已经从 C 中自动删除了 C_{obs}。现在令 X 为任意 C-空间或者相空间，令 X_{obs} 如表述 8.2 所定义。这产生了表述 14.1 所定义的 X_{free}。

（2）在表述 8.2 中，状态转移方程为 $\dot{x}=u$，它直接指定了切空间 $T_x(X)$ 中的速度。现在用任意系统 $\dot{x}=f(x,u)$。在这种情况下，$U(x)$ 不再是 $T_x(X)$ 的一个子集。它仍然包括特殊终止行动 u_T。

（3）表述 14.1 中包括 x_I，但是现在将其删除，这与表述 8.2 一致。

（4）反馈规划现在被定义为一个函数 $\pi: X_{free} \rightarrow U$。对于一个给定的状态 $x \in X_{free}$，产生一个行动 $\pi(x)$。将 π 与 f 进行组合就产生了由 $\dot{x}=f(x,\pi(x))$ 给定的 $T_x(X)$ 中的速度。因此 π 定义了 X_{free} 上的一个向量场。

令 t_F 表示应用 u_T 的时刻。可行规划和最优规划都可以用代价泛函来定义：

$$L(\tilde{x}_{t_F},\tilde{u}_{t_K})=\int_0^{t_F} l(x(t),u(t))\mathrm{d}t+l_F(x(t_F)) \tag{14.26}$$

它与第 8.4.1 节中的一样。现在详细说明微分约束下反馈运动规划问题。

与第 8 章最重要的不同之处在于 $\dot{x}=u$ 被 $\dot{x}=f(x,u)$ 代替，这样允许用第 13 章的复杂微分模型。产生于 π 的向量场必须满足 $\dot{x}=f(x,u)$ 所施加的微分约束。在第 8.4.4 节中，对可允许的向量场施加了简单约束，如速度界或者平滑性，但是这些约束不如第 13 章中的模型那么严格。例如 Dubins 车不允许反向运动，而第 8.4.4 节中的约束则允许向任意方向运动。

14.5.2 带插值的动态规划

就像在第 14.4 节中所观察到的，微分约束下的运动规划非常具有挑战性。而反馈又使得问题进一步复杂化。如果 $X_{obs}=\phi$，那么可以用控制论中的许多方法来设计反馈规划，参见第 15.2.2 节和文献[194,526,848]。在许多情况下，设计反馈规划并不比计算开环轨迹更困难。但是如果 $X_{obs}\neq\phi$，反馈通常使得问题更难解决。

幸运的是，动态规划再次扮演了救星的角色。在第 2.3 节，值迭代为离散状态空间和状态转移方程产生反馈规划。值得注意的是这种思路可以被推广到 U 和 X 是连续的和连续时间这种情况。目前多数用到动态规划的工具已经在第 8.5.2 节介绍过。那里主要的思路是

用插值来表示最优尚需代价 G^*，并用到对运动规划问题的离散时间近似。

在目前情形下可以用离散时间模型来得到 $x_{k+1} = f_d(x_k, u_k)$ 形式的离散阶段状态转移方程。就像第 8.5.2 节中所介绍的那样，代价泛函用式（8.65）来近似。用系统模拟器的结果进行数值积分，产生的每阶段的代价为 $l_d(x_k, u_k)$。用后向值迭代，递推动态规划为：

$$G_k^*(x_k) = \min_{u_k \in U_d} \left\{ l_d(x_k, u_k) + G_{k+1}^*(x_{k+1}) \right\} \tag{14.27}$$

它与式（2.11）和式（8.56）相似。如果 U 并不已经是有限的，那么使用行动采样的有限集合 U_d。用系统模拟器来确定沿着轨迹的某些点是否在 X_{obs} 中。在这种情况下，$l_d(x_k, u_k) = \infty$，以防选择违背约束的行动。

就像第 8.5.2 节中所介绍的那样，采样点的一个集合 $P \subset X$ 用来在 X 上近似 G^*。$X \setminus P$ 中的点上所需要的值可用插值来得到。例如可以应用图 8.20 的重心重分，在单形上在 $O(n \lg n)$ 时间内进行插值，其中 n 是 X 的维数。

通常情况下，后向值迭代从某最终阶段 F 开始并向后进行经过所有阶段。当所有尚需代价值稳定时发生终止。阶段 F 初始化产生 $G_F^*(x) = 0$，$x \in X_G \cap P$；否则 $G_F^*(x) = \infty$。在每个 $x \in P$ 上计算式（14.27）并使用插值得到 $G_{k+1}^*(x_{k+1})$。

所产生的平稳尚需代价函数 G^* 可以用作 X_{free} 上的一个导航函数，如第 8.5.2 节所述。回忆第 8 章，导航函数通过一个局部算子而转化为一个反馈规划。目前情形下的局部算子为：

$$\pi(x) = \min_{u \in U_d} \left\{ l_d(x, u) + G^*(x) \right\} \tag{14.28}$$

它为 X_{free} 中落入 P 中某些采样点的一个插值邻域内的任意状态产生一个行动。

非常遗憾，这里提出的方法只在几维空间中有用。如果 $X = C$，那么它可以应用于第 13.1.2 节中的系统。如果考虑动力学，那么在许多环境下维数太大，因为 X 的维数通常是 C 的维数的两倍。例如，如果 $\mathcal{A}$ 是平面中的一个刚体，那么 X 的维数是六，这通常已经是实际使用的上限。

可以将这里采用的动态规划与第 14.4.1 节和第 14.4.2 节中介绍的进行比较，在那里是构造一个搜索图。如果使用 Dijkstra 算法（或者在时间最优性情况下的广度优先搜索），那么用动态规划原理，所产生的解是近似最优的。为了确保收敛，给出的分辨率完备性自变量是基于 f 上的 Lipschita 条件的。当搜索没有找到解时允许提高分辨率。值迭代是基于计算尚需代价函数，而不是基于计算一个搜索图。与基于树的那些方法都有前向和后向版本一样，这里可以采用前向和后向值迭代。只不过分辨率完备性更加复杂，因为 x_I 并不固定。因此不知道某个分辨率对于特定应用来说是否足够好。如果已知 x_I，那么通过系统模拟器，使用 G^* 可以生成一条从 x_I 的轨迹。如果轨迹不能到达 X_G，那么可通过增加对 P 和 U_d 的采样点或者减小 Δt 来提高分辨率。在 f 上的 Lipschita 条件下，方法收敛到真实最优尚需代价[94,170,568]。因此可以认为值迭代关于一个给定的 x_I 是分辨率完备的。收敛甚至可以扩展到计算带有大自然行动的最优反馈规划。这就扩展了第 10.6 节的值迭代方法。

通过构造值迭代的类似 Dijkstra 版本，可以使基于搜索图的方法和基于值迭代的方法之间的关系更加密切，如第 8.5.2 节所述。这就扩展了 Dijkstra 算法，在第 2.3.3 节的有限情况下 Dijkstra 算法被视为是对值迭代的一种改进提高。对值迭代有所提高是因为认识到

在式（14.27）的计算中，尚需代价值并不发生改变。这是由两个因素引起的：（1）在某些状态下，没有发现导致 X_G 的轨迹，因此尚需代价一直为无穷大；（2）在某状态下，最优尚需代价可能已经被计算过，将来的迭代并不能改进代价。

可以构造类似 Dijkstra 算法的前向或者后向版本。考虑后向情况。在第 8.5.2 节用后向投影的概念来刻画能够一步到达另外一个状态集的状态的集合。这在式（8.68）中用来在类似 Dijkstra 算法执行过程中定义边缘（frontier）。目前情形下所处理的系统是 $\dot{x}=f(x,u)$，从本质上没有什么不同。一旦进行了离散时间近似，后向投影的定义本质上与式（8.66）相同。使用第 14.2.2 节的离散时间模型，状态 $x\in X_{free}$ 的后向投影为：

$$B(x)=\{x'\in X_{free}\mid \exists u\in U_d, x=f_x(x',u)\} \tag{14.29}$$

后向投影与第 14.2.1 节的后向时间受限可达集有密切的关系。后向投影可以被认为是后向时间受限可达集的离散一步版本，它表示在 Δt 时间内应用一个常数行动 $u\in U_d$ 所能够到达 x 的那些状态。如第 8.5.2 节所述，计算对边缘集合的一个近似可能在实际中更可取。这可以通过对后向投影的近似来获得，一般情况下，微分约束下的后向投影比第 8.5.2 节所考虑的情况更加复杂。一种简化是在定义 $B(x)$ 中忽略与障碍物的碰撞。也可以用真实后向投影的一个简单包围体。这种折中与第 5.3.2 节所提到的碰撞检测中的情况类似。有时系统的独特结构对确定后向投影有极大的帮助。例如对一个二重积分器而言，可以得到一个非常好的波前传播算法，这在第 14.6.3 节将得到利用。更多关于值迭代和类似 Dijkstra 版本的内容参见文献[610]。

14.6 解耦规划法

14.6.1 解耦大问题的不同方法

基于采样的算法随着计算机技术的发展而不断得到提高，在实际应用中它直接解决微分约束下的复杂的规划问题变得越来越可行。但是在许多情况下由于数值积分、碰撞检测、高维状态空间中复杂的障碍物等缘故，进行这种计算的开销仍然是非常大的。解耦方法就显得非常重要，因为它们将大的问题分解为更容易求解的模块。对于钢琴搬运工问题，这种方法已经在第 7 章中讨论过。第 7.1.3 节介绍了用速度调节方法来处理时变障碍物，第 7.2.2 节提出用解耦方法来协调多机器人的运动。

理想情况下，在有障碍物和微分约束的任何状态空间上，我们希望得到反馈规划。这假设在执行过程中能够对状态进行可靠测量。第 14.5 节为解决该问题提供了最好的通用技术，但遗憾的是，它们只能解决维数不高的问题。如果有大量的感测不确定性，那么反馈规划必须被定义在第 11 章介绍的 I-空间上。回顾第 1.4 节，图 1.19 显示了一个将大规划问题解耦为一系列改进问题的公共模型。一个典型的解耦方法涉及以下四个模块：

（1）用一个运动规划算法来发现一条无碰撞路径 $\tau:[0,1]\rightarrow C_{free}$。

（2）将 τ 变换为一个新的路径 τ' 以满足 C 上的速度约束（如果有）。例如这能确保 Dubins 车能够跟踪路径。在多数环境中至少还需要某种路径平滑。

（3）为 τ' 计算一个定时函数 $\sigma:[0,t_F]\rightarrow[0,1]$ 以使 $\tau'\circ\sigma$ 是 C_{free} 中的一条以时间为参数的路径，并带有如下要求。状态轨迹 $\tilde{x}$ 在所有时间内都必须满足 $\dot{x}=f(x(t),u(t))$ 且 $u(t)\in U(x(t))$ 直到在 t_F 时间应用 u_T。

（4）设计一个跟踪 $\tilde{x}$ 的反馈规划（或者反馈控制律）$\pi:X\rightarrow U$。规划应该尝试使得期望状态和执行过程中测量的状态之间的误差最小化。

在给定最新技术和计算能力的情况下，这种方法的重要性可能会有所降低，但是它仍然是分解和解决问题的一种重要手段。但需要注意的是这种分解具有随意性。如果每个模块都得到解决，那么足以产生问题的解，但这种分解不是必要的。由于先前模块选得很差，那么在沿着这种方法的任何一步中都有可能失去完备性。对于模块来说，很难考虑到后来可能出现的问题。

有各种各样的合并模块的方法。第 14.4 节介绍的方法能够解决如下两个情况之一：（1）前两个模块同时被解决，如果在 C_{free} 中计算了满足 $\dot{q}=f(q,u)$ 的路径；（2）前三个模块同时被解决，如果在 X_{free} 中计算了满足 $\dot{x}=f(x,u)$ 的路径。第 14.5 节所介绍的方法可同时解决所有四个模块，但只限于低维空间。

现在考虑将模块保持分离。第 II 部分中的规划方法可以用来解决第一个模块。第 14.6.2 节将介绍实现第二个模块的方法。第 14.6.3 节将介绍解决第三个模块的方法，可能同时也解决了第二个模块。第四个模块是一个控制问题，许多教科书对它都有介绍[526,848,858]。

14.6.2　规划与变换

对于本节的解耦方法，假设 $X=C$，这意味着只有速度约束，如第 13.1 节所述。系统可能被指定为 $\dot{q}=f(q,u)$ 或者隐含地被指定为 $g_i(q,\dot{q})=0$ 形式的约束集合。本节的思路很容易被扩展到相空间。这里给出的方法最初由 Laumond 提出（见文献[599]），并应用于第 13.1.2 节中的简易汽车，在文献[599]中还介绍了该方法在其他方面的应用。

图 14.21 描述了规划-变换方法。在第 1 步中，我们忽略微分约束来计算无碰撞路径 $\tau:[0,1]\rightarrow C_{free}$。然后迭代修改路径直到它满足约束条件。在每次迭代中，通过指定某 $s_1,s_2\in[0,1]$ 使得 $s_1<s_2$ 来选择一个子区间 $[s_1,s_2]\subseteq[0,1]$。这些点可能用随机序列来选择或者用确定的方法来选择。该方法可能会用二叉重分来细分区间，并在迭代中逐步提高 $[0,1]$ 上的分辨率。

规划-变换方法

1. 用第 II 部分中所介绍的一个运动规划算法计算一条路径 $\tau:[0,1]\rightarrow C_{free}$。
2. 选择某 $s_1,s_2\in[0,1]$ 使得 $s_1<s_2$，使用一个 LPM 试图用满足微分约束的一条路径 γ 来代替 τ 的从 $\tau(s_1)$ 到 $\tau(s_2)$ 的部分。
3. 如果现在 τ 在[0,1]上满足微分约束，那么算法结束；否则，返回到第 2 步。

图 14.21　规划-变换方法的轮廓

对于所选择的每个区间 $[s_1,s_2]$，用一个 LPM 来计算满足 $\gamma(0)=\tau(s_1)$ 和 $\gamma(1)=\tau(s_2)$ 条件的路径片段 $\gamma:[0,1]\rightarrow C_{free}$。如果 LPM 不能连接两个位形或者可能发生碰撞，可能它会失败。在这种情况下，选择另外一个子区间，然后重复过程。当 LPM 成功时，τ 更新为 τ'：

$$\tau'(s)=\begin{cases}\tau(s) & (s<s_1)\\ \gamma\big((s-s_1)/(s_2-s_1)\big) & (s\in[s_1,s_2])\\ \tau(s) & (s>s_2)\end{cases} \tag{14.30}$$

γ 的自变量从 s_1 到 s_2，而不是从 0 到 1。

例 14.5（Dubins 车的规划-变换） 假设任务是为 Dubins 车规划路径。图 14.22 显示了一条路径 τ，它可能是在忽略微分约束的情况下用运动规划算法计算而来。有两个锐角汽车不能经过。假设随机选择 s_1 和 s_2，它们出现在图 14.22 所示的位置。$\tau(s_1)$ 和 $\tau(s_2)$ 之间的 τ 需要用 Dubins 车能够执行的路径来代替。注意到匹配 $\tau(s_1)$ 和 $\tau(s_2)$ 的方向是很重要的，因为它们是位形的一部分。

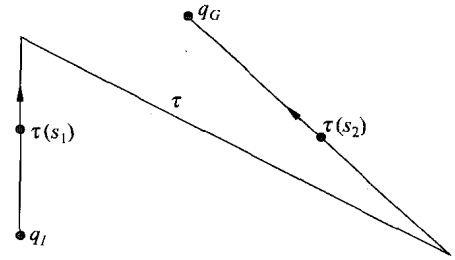


图 14.22 忽略微分约束的一条初始路径

图 14.23 显示了一个替代路径 γ。这是通过实现如下 LPM 得到的。对于 Dubins 车，任何位形之间的路径都可以通过在起止位形处画圆来发现，如图 14.23 所示。每个圆对应于可能的最急右转或者左转。很容易发现一条与一个位形的圆相切、与汽车的方向相匹配的直线。用 γ，路径 τ 更新为 τ'，如图 14.24 所示，它满足 Dubins 车的微分约束。这个问题很简单，在实际中，代替路径片段可能需要数十次迭代。并且如果采用随机选择，那么形式为 $[0,s]$ 和 $[s,1]$ 的区间一定不能被忽略。 ■

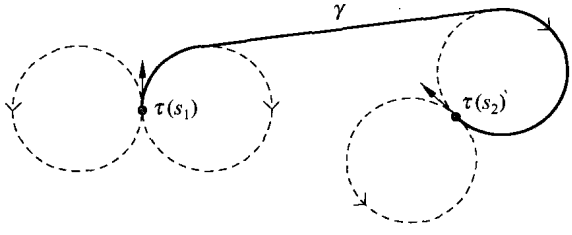


图 14.23 通过用一个双切线连接最小转弯半径所生成的两个圆，总能够发现 Dubins 车的一条路径；路径不一定是最优的，最优路径参见第 15.3.1 节

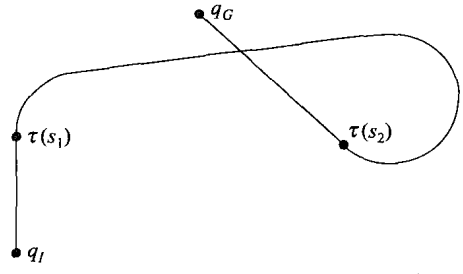


图 14.24 通过替代置换，Dubins 车能够跟踪所产生的路径 τ'

例 14.5 似乎很容易，因为存在一个简单的局部规划器，并且没有障碍物。假设 τ 行进通过一个狭窄的“之”字形走廊。在这种情况下，由于 Dubins 车不能转急弯，因此解可能不存在。如果有一个障碍物恰好与图 14.24 中的圆环相交，那么替代将会失败。一般情况下，并不能保证用于替代的片段是无碰撞的。对于 LPM 来说构造尽可能接近原始路径的路径片段是非常重要的。对于 Dubins 车，这在许多情况下是不可能的。例如 Dubins 车向后运动一小段距离都会要求沿着图 14.23 所示的圆运动。即使减小两个位形之间的距离，Dubins 车需要行驶的距离也不会接近于零。即使 Dubins 车用可能的最短路径，也是如此。

LPM 应该具有什么特性才能保证规划-变换方法的分辨率完备性？文献[599]给出了一个充分条件。令 ρ 表示 X 上的一个度量。一个 LPM 被认为是满足拓扑性质（topological

property)，当且仅当如下论述成立：对于任意 $\varepsilon>0$，存在某 $\delta>0$ 使得对于任意 $q,q'\in C_{free}$，$\rho(q,q')<\delta$ 意味着对于所有 $s\in[0,1]$，有 $\rho(\tau(s),q)<\varepsilon$。如果一个 LPM 满足拓扑性质，那么 C_{free} 中任何无碰撞路径都可以变换成满足微分约束的路径。假设一条路径 τ 在 C_{free} 中有至少 ε 的空隙。将 τ 的定义域划分为许多区间使得在每个区间上 q 的变化不超过 δ，那么 LPM 将产生用于替代的无碰撞路径片段。

对于 Reed-Sheep 车来说可以设计这种的 LPM，因为它是短时局部可控的，这种性质将在第 15.1.3 节和第 15.4 节进行介绍。一般情况下，对于分析和设计有效的 LPM 来说，第 15 章的许多技术是非常有用的。

对于涉及 k 个 $g_i(q,\dot{q})=0$ 形式的隐含约束的问题，可以对规划-变换方法进行适应性修改。文献[862]所介绍的多级方法如图 14.25 所示（文献[335]也介绍了一个相似的方法）。思路是将 k 个约束排成一个序列，每次只用一个。初始情况下，可规划一条忽略约束的路径。首先用图 14.21 的规划-变换方法对该路径进行变换以满足 $g_1(q,\dot{q})=0$ 且避免碰撞。如果成功，那么所产生的路径然后变换为无碰撞且满足 $g_1(q,\dot{q})=0$ 和 $g_2(q,\dot{q})=0$。通过每次增加一个约束，该过程重复进行直到方法失败或者所有 k 个约束都得到满足。

多级方法

1. 用标准的运动规划算法计算一条路径 $\tau:[0,1]\rightarrow C_{free}$。
2. 将 τ 变换为一条满足 $g_j(q,\dot{q})=0$ 的无碰撞路径，j 从 1～i。
3. 如果第 2 步的变换失败，那么过程结束并报告失败。
4. 如果 $i<k$，k 是隐含速度约束的数量，那么递增 i 并返回到第 2 步。否则，结束并报告 τ 就是一条满足所有约束的路径。

图 14.25　多级方法每次只考虑一个隐含约束

14.6.3　具有路径约束的轨迹规划

本节假设给定一条路径 $\tau:[0,1]\rightarrow C_{free}$。它可以用第 II 部分介绍的运动规划算法来计算，也可以人为给定。那么剩下的任务是用满足相空间 X 上微分约束的方式确定沿着路径的速度。假设每个状态 $x\in X$ 表示一个位形和它的时间导数，那么 $x=(q,\dot{q})$。令 n 表示 C 的维数，因此 X 的维数是 $2n$。一旦给定一条路径，那么在 X 中只剩两个自由度：（1）沿着 τ 的定义域的位置 $s\in[0,1]$；（2）每个 s 处的速度 $\dot{s}=\mathrm{d}s/\mathrm{d}t$。整个状态 x 都可以从这两个参数来获得。当状态发生变化时，它必须满足给定的系统 $\dot{x}=f(x,u)$。可以看到将会出现一个二维规划问题，它可用许多技术来解决。对第 7.1 节中的时变运动规划的解耦版本也有相似的概念。目前情形下的表示是受到几十年前发展起来的关于机器人操作臂时标路径的启发[459,878,881]。那时的计算机处理速度非常慢，从而促进了解耦方法的发展。

14.6.3.1　用 $s,\dot{s},\ddot{s}$ 表达系统

假设给定一个如下形式的系统：

$$\ddot{q}=h(q,\dot{q},u) \tag{14.31}$$

其中有 n 个行动变量 $u=(u_1,\cdots,u_n)$。在例 14.6 中，我们通过二重积分器 $\ddot{q}=u$ 这种简单情况来阐述一些即将出现的概念。C 中的加速度由状态 $x=(q,\dot{q})$ 和行动 u 来确定。假设 $u\in U$，

其中 U 是 $\mathbb{R}^n$ 的一个 n 维子集。如果 h 在 x 处是非奇异的，那么从 $u \in U$ 的选择中会出现可能加速度的一个 n 维集合。这意味着它是全驱动的。如果行动变量少于 n，那么通常会没有足够的自由度来跟踪一条特定路径。因此 U 必须是 n 维的。但是选择什么行动 u 才能约束运动来跟踪给定路径 τ？为此，需要分别将变量 q、$\dot{q}$ 和 $\ddot{q}$ 与路径定义域 s 及其一阶导数 $\dot{s}$、二阶导数 $\ddot{s}$ 联系起来。这就产生了对应于跟踪路径的行动 U 的一个子集。

假设给定 s、$\dot{s}$、$\ddot{s}$ 和一条路径 τ。位形 $q \in C_{free}$ 为：

$$q = \tau(s) \tag{14.32}$$

假设 τ 的所有一阶和二阶导数存在。用链式法则，速度 $\dot{q}$ 可为：

$$\dot{q} = \frac{\mathrm{d}\tau}{\mathrm{d}s}\frac{\mathrm{d}s}{\mathrm{d}t} = \frac{\mathrm{d}\tau}{\mathrm{d}s}\dot{s} \tag{14.33}$$

其中导数 $\mathrm{d}\tau/\mathrm{d}s$ 在 s 处计算。再次求导就得到加速度：

$$\begin{aligned}\ddot{q} &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}\tau}{\mathrm{d}s}\dot{s}\right)\\ &= \frac{\mathrm{d}^2\tau}{\mathrm{d}s^2}\frac{\mathrm{d}s}{\mathrm{d}t}\dot{s} + \frac{\mathrm{d}\tau}{\mathrm{d}s}\ddot{s}\\ &= \frac{\mathrm{d}^2\tau}{\mathrm{d}s^2}\dot{s}^2 + \frac{\mathrm{d}\tau}{\mathrm{d}s}\ddot{s}\end{aligned} \tag{14.34}$$

所有状态 $x = (q, \dot{q})$ 都可以用式（14.32）和式（14.33）从 $(s, \dot{s})$ 得到。

下一步是得到一个看起来像式（14.31）的方程，只不过是用 s、$\dot{s}$ 和 $\ddot{s}$ 表达的。用 $\tau(s)$ 代替 q、用式（14.33）的右边代替 $\dot{q}$ 就可以从 $h(q, \dot{q}, u)$ 得到一个函数 $h'(s, \dot{s}, u)$：

$$h'(s, \dot{s}, u) = h\left(\tau(s), \frac{\mathrm{d}\tau}{\mathrm{d}s}\dot{s}, u\right) \tag{14.35}$$

这产生了：

$$\ddot{q} = h'(s, \dot{s}, u) \tag{14.36}$$

对于一个给定的状态 x（它可以从 s 和 $\dot{s}$ 来得到），在式（14.36）中选择不同的行动所得到的加速度的集合与式（14.31）中原始系统的加速度相同。仅有的不同之处在于 x 现在被约束在 X 的一个二维子集上，它们是通过选择 s 和 $\dot{s}$ 值所能够到达的那些状态。

将式（14.34）用在式（14.36）的左边，这样就约束加速度产生跟踪 τ 的运动。这产生：

$$\frac{\mathrm{d}^2\tau}{\mathrm{d}s^2}\dot{s}^2 + \frac{\mathrm{d}\tau}{\mathrm{d}s}\ddot{s} = h'(s, \dot{s}, u) \tag{14.37}$$

将式（14.34）的第一项移到右边，它也可以被表达为：

$$\frac{\mathrm{d}\tau}{\mathrm{d}s}\ddot{s} = h'(s, \dot{s}, u) - \frac{\mathrm{d}^2\tau}{\mathrm{d}s^2}\dot{s}^2 \tag{14.38}$$

实际上式（14.38）表示的是 n 个方程。对于 $\mathrm{d}\tau_i/\mathrm{d}s \neq 0$ 的每个 i，求解 $\ddot{s}$ 可以得到形式为：

$$\ddot{s} = \frac{1}{\mathrm{d}\tau_i/\mathrm{d}s}h_i'(s, \dot{s}, u_i) - \frac{\mathrm{d}^2\tau_i}{\mathrm{d}s^2}\dot{s}^2 \tag{14.39}$$

的一个约束。

14.6.3.2　确定允许的加速度

现在描述 U 中能够引起跟踪 τ 的行动。一个行动 $u \in U$ 跟踪 τ，当且仅当满足形式为式（14.39）的每个方程。如果对于从 1 到 n 的所有 i，$\mathrm{d}\tau_i / \mathrm{d}s \neq 0$，那么存在 n 个这种方程。假设选择 u_1，并且对第一个方程求解 $\ddot{s}$。将所确定的 $\ddot{s}$ 的值代入其余 $n-1$ 个方程就可以得到其他行动变量 $u_2, \cdots, u_n$ 所要求的值。这意味着跟踪 τ 的行动最多是 U 的一个一维子集。

如果对于某 i，$\mathrm{d}\tau_i / \mathrm{d}s = 0$，那么跟踪路径要求 $\dot{q}_i = 0$。约束条件是 $h_i(q, \dot{q}, u) = 0$，而不是式（14.39）。例 14.6 提供对此的一个简单阐述。如果对于所有 i，$\mathrm{d}\tau_i / \mathrm{d}s = 0$，那么不允许改变位形。这发生在 τ 是一个常数函数这种退化情况下。

在许多情况下，不存在满足所有约束方程的 u 的值。这意味着在那种特殊状态下无法跟踪路径。如果可能，应该通过在 X 上定义相约束的方法来删除这种状态。如果路径 τ 没有选择好，那么违背其中一个相约束是在所难免的。可能存在某 s，使得没有 $u \in U$ 能够跟踪 τ，无论 $\dot{s}$ 怎样。

即使一个状态轨迹从某种意义上是最优的，它的性能仍然依赖于给定的路径 $\tau : [0,1] \rightarrow C_{free}$。考虑如图 14.26 所示的路径。在 $\tau(1/3)$ 时，到达一个“角”。这违背了可微性假设，保持在 τ 上要求无穷大加速度。对于某些模型，有可能停在 $\tau(1/3)$ 上，然后再次启动。例如，假设平面上存在一个质点。它减速停在 $\tau(1/3)$ 上，然后开始向一个新的方向转动来准确地跟踪曲线。这假设质点是全驱动的。如果在 C 上有非完整约束，就像 Dubins 车，那么在考虑加速度之前，给定的路径必须至少满足这些约束。这种情况下的解依赖于解耦向量场的存在性[159,226]。

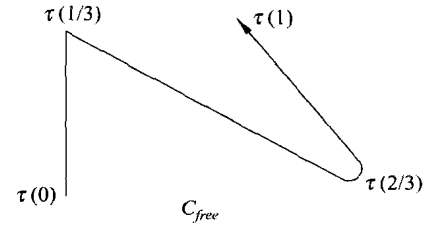


图 14.26　有路径约束的轨迹规划产生的一条不好的路径

在构造 τ 中通常希望对运动规划算法已经产生的任何角进行圆化。这是有益的，但是还不能完全解决问题。$\tau(2/3)$ 附近的路径并不令人满意，因为它过于弯曲。在固定速度下，通常需要更大的加速度来跟踪急转弯。可能不得不降低速度，因为 τ 在 C 上有急转弯。假设自主开发一辆双层旅游公共汽车。很明显，在高速情况下跟踪 $\tau(2/3)$ 附近的路径可能会引起公共汽车侧翻。公共汽车不得不降低速度，因为它受制于 τ。

14.6.3.3　路径约束下的相空间

回忆第 14.4.1 节中的方法，对于某些合适的 $U'(q, \dot{q}) \subseteq U$，方法使得形式为 $\ddot{q} = h(q, \dot{q}, u)$ 的系统可以表示为 $\ddot{q} = u'$（在图 14.15 中阐述过）。这可以将一些系统假设成多个独立的、有行动空间上相约束的二重积分器。这里可以用相同的思路来得到单个积分器。

令 S 表示一个有路径约束的二维相空间，其中每个元素的形式是 $(s, \dot{s})$，它们表示沿着 τ 的位置和速度。这就对原始相空间 X 的一个二维子集进行了参数化。每个原始状态向量为 $x = (q, \dot{q}) = (\tau(s), \mathrm{d}\tau / \mathrm{d}s\ \dot{s})$。在 S 中的点上可能的加速度是什么？在每个 $(s, \dot{s})$，可以确定 U 的满足（14.39）方程的一个子集。每个有效行动产生一个加速度 $\ddot{s}$。令 $U'(s, \dot{s}) \subseteq \mathbb{R}$ 表示可以由一个行动 $u \in U$ 得到的 $\ddot{s}$ 的所有值的集合，对于每个 i（除了使 $\mathrm{d}\tau_i / \mathrm{d}s = 0$ 的那些 i 以外），行动满足（14.39）方程。现在系统可以表示为 $\ddot{s} = u'$，其中 $u' \in U'(s, \dot{s})$。完成所有

这些工作以后，我们有了二重积分器。但问题在于对某些系统来说$U'(s,\dot{s})$很难确定。它可能由一个区间组成。也可能由没有交集的多个区间组成，甚至也有可能是空集。假设已经描述了$U'(s,\dot{s})$，那么很容易用本章的方法解决规划问题的其余部分。一个二重积分器并不难以处理，因此存在有效的基于采样的算法。

现在考虑一个障碍区域$S_{obs} \subset S$。这包括属于X_{free}的任何状态。给定s和$\dot{s}$，可以计算状态x来确定是否违背X上的约束条件。通常情况下，构造τ避免与障碍物发生碰撞，但是一些相约束可能已经存在。障碍区域S_{obs}也包括使$U'(s,\dot{s})$为空集的那些点。令S_{free}表示$S \setminus S_{obs}$。

在考虑计算方法之前，先给出一些例子。

例 14.6（有路径约束的二重积分器） 考虑两个二重积分器。从物理上，它对应于$\mathbb{R}^2$中的一个运动质点。因此$\mathcal{C}=\mathcal{W}=\mathbb{R}^2$。令$U=[-1,1]^2$和对于$u \in U$，有$\ddot{q}=u$。选择路径$\tau$迫使质点沿着一条直线运动。对于直线路径，$\mathrm{d}\tau/\mathrm{d}s$是常数且$\mathrm{d}^2\tau/\mathrm{d}s^2=0$。基于这些认识和$h'(s,\dot{s},u)=u$这样一个事实，对于$i=1,2$，（14.39）方程被简化为：

$$\ddot{s}=\frac{u_i}{\mathrm{d}\tau_i/\mathrm{d}s} \tag{14.40}$$

假设$\tau(s)=(s,s)$，即质点必须沿着一条经过$\mathcal{C}$的原点的对角线运动。这可以进一步将式（14.40）简化为$\ddot{s}=u_1$和$\ddot{s}=u_2$。因此可以任意选择$u_1 \in [-1,1]$，但是必须选择u_2使得$u_2=u_1$。受约束的系统可以记为一个二重积分器$\ddot{s}=u'$，其中$u' \in [-1,1]$。从u'产生u_1、u_2为$u_1=u_2=u'$。注意到U'在S不发生变化，这是因为一条直线路径是退化的。

现在考虑将运动约束到一般直线：

$$\tau(s)=(a_1 s+b_1, a_2 s+b_2) \tag{14.41}$$

其中a_1和a_2是非零的。在这种情况下，式（14.40）产生$\ddot{s}=u_1/a_1$和$\ddot{s}=u_2/a_2$。因为每个$u_i \in [-1,1]$，那么每个方程表明$\ddot{s} \in [-1/a_i, 1/a_i]$。加速度必须在这两个区间的交集中。如果$|a_1| \geqslant |a_2|$，那么$\ddot{s} \in [-1/a_1, 1/a_1]$。可以指定$u'=u_1$并令$u_2=u'a_2/a_1$。如果$|a_1|>|a_2|$，那么$\ddot{s} \in [-1/a_2, 1/a_2]$，$u'=u_2$和$u_1=u'a_1/a_2$。

假设$a_1=0$和$a_2 \neq 0$。路径是：

$$\tau(s)=(q_1, a_2 s+b_2) \tag{14.42}$$

其中q_1是固定的，质点被约束沿着$\mathcal{C}=\mathbb{R}^2$中的一条垂线运动。在这种情况下，从式（14.40）只得到一个约束$\ddot{s}=u_2$。u_1单独约束到$u_1=0$，因为水平方向运动被禁止。

若n个独立的二重积分器被约束到一条直线，则可以得到一个相似的结果。有n个式（14.40）形式的方程。对应于$|a_i|$最大的那个$i \in \{1,\cdots,n\}$决定加速度的范围为$\ddot{s} \in [-1/a_i, 1/a_i]$。行动$u'$定义为$u'=u_i$，并从其余$n-1$个方程得到$u_j$，$j \neq i$。

现在假设τ是非线性的，在这种情况下，对于$\mathrm{d}\tau_i/\mathrm{d}s \neq 0$的每个$i$，（14.39）方程变成：

$$\ddot{s}=\frac{u_i}{\mathrm{d}\tau_i/\mathrm{d}s}-\frac{\mathrm{d}^2\tau_i}{\mathrm{d}s^2}\dot{s}^2 \tag{14.43}$$

现在集合$U'(s,\dot{s})$在S上变化。当速度$\dot{s}$增加时，$U'(s,\dot{s})$为非空的可能性越来越小。即（14.43）形式的所有方程存在解的可能性越来越小。在一个物理系统中，这意味着在路径

上需要急转弯。在高速情况下，这要求$[-1,1]^n$范围以外的加速度。 ■

对更加复杂的系统也可以应用相同的思路。这并不令人感到惊奇，因为在第 14.4.1 节中$\ddot{q}=h(q,\dot{q})$形式的系统被解释为多个独立的、$\ddot{q}=u'$形式的二重积分器，其中$u'\in U'(q,\dot{q})$提供可能的加速度。在这种意义下并根据例 14.6，将一个一般系统的运动约束到一条路径τ只是对$U'(q,\dot{q})$做进一步限制。所产生的可允许加速度可能最多是一维的。

如下例子是（14.39）方程对机械臂而言的具体化。

例 14.7（有路径约束的操作臂） 假设系统被描述成第 13.4.2 节的式（13.142）。多年来机械臂控制都采用这种形式。用s、$\dot{s}$和$\ddot{s}$来表达q、$\dot{q}$和$\ddot{q}$就可以产生（14.39）形式的约束。这要求采用式（14.32）、式（14.33）和式（14.34）。直接代入式（13.142）就产生：

$$M(\tau(s))\left(\frac{\mathrm{d}^2\tau}{\mathrm{d}s^2}\dot{s}^2+\frac{\mathrm{d}\tau}{\mathrm{d}s}\ddot{s}\right)+C\left(\tau(s),\frac{\mathrm{d}\tau}{\mathrm{d}s}\dot{s}\right)\frac{\mathrm{d}\tau}{\mathrm{d}s}\dot{s}+g(\tau(s))=u \tag{14.44}$$

这可以简化为n个如下形式的方程：

$$\alpha_i(s)\ddot{s}+\beta_i(s)\dot{s}^2+\gamma_i(s)\dot{s}=u_i \tag{14.45}$$

每个方程求解$\ddot{s}$就产生（14.39）方程的一个特殊情况。如同例 14.6，每个方程确定了$\ddot{s}$的一个包围区间。所有n个方程的区间的交集产生$\ddot{s}$的可允许区间。行动u'指示区间中的加速度，并且从式（14.45）中可以得到原始行动变量u_i。如果$\mathrm{d}\tau_i/\mathrm{d}s=0$，那么$\alpha_i(s)=0$，这对应于没有约束的情况。取而代之的是必须选择向量u使得$\dot{q}_i=0$。 ■

14.6.3.4　用动态规划计算最优解

第 14.5 节介绍的带插值的动态规划一旦用有路径约束的相空间$S\subset\mathbb{R}^2$来表达，那么就可以用它来解决问题。τ的定义域提供约束$0\leqslant s\leqslant 1$。假设只需要沿着路径向前进，应该不需要向相反方向运动。这意味着$\dot{s}>0$。为了使得S有界，通常假设一个上界$\dot{s}_{\max}$，超过该上界以后往往由于速度过高而不能跟踪路径。

这产生了如图 14.27（a）所示的规划问题。系统表达为$\ddot{s}=u'$，其中$u'\in U'(s,\dot{s})$。初始为$(0,\dot{s}_i)$，目标为$(1,\dot{s}_g)$。典型情况下，$\dot{s}_i=\dot{s}_g=0$。如图 14.27 所示的区域被限制在相空间的第一象限，因为只允许s和$\dot{s}$为正值（在图 14.13 中，q和$\dot{q}$可正可负）。这意味着所有运动都是向右的。行动确定将发生的是加速还是减速。

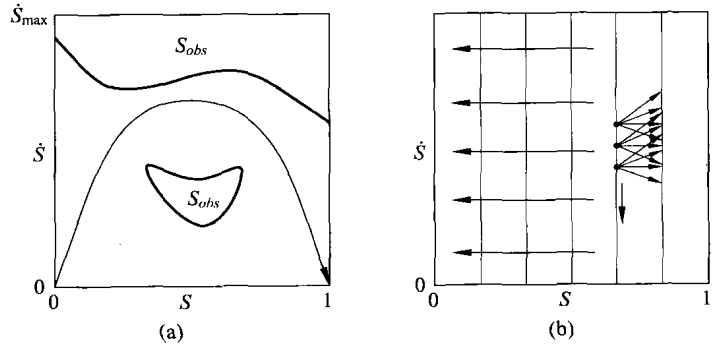


图 14.27　（a）有路径约束的相空间中的规划；（b）由于假设向前进，值迭代可以简化成穿过S中的等间隔垂线的一个快速波前传播

通过离散化 S 和 $U'(s,\dot{s})$，就可以很方便地应用带插值的后向值迭代。由于约束 $\dot{s}>0$，很容易产生算法的一个类似 Dijkstra 的版本。可以进行一个简单的波前传播，过程从 $s=1$ 开始，并在垂直波中向后进行直到到达 $s=0$，如图 14.27（b）所示。后向投影（14.29）方程可以得到显著简化。假设对于某固定的 $\Delta s>0$，s 轴在每个 Δs 上离散化为 $m+1$ 个等分值 $s_0,\cdots,s_m$。因此 $s_k=(k\Delta s)/m$。下标 k 可以解释为阶段。从 $k=m$ 开始，如果相应的初始相代表目标，那么最后的尚需代价 $G_m^*(s_m,\dot{s}_m)$ 定义为 0，否则代价为 ∞。在每个 s_k，$\dot{s}$ 的值被采样，并且尚需代价函数用沿着纵轴的一维线性插值来表示。在每个阶段，在每个 $\dot{s}$ 的采样点上进行动态规划计算：

$$G_k^*(s_k,\dot{s}_k)=\min_{u'\in U'(s_k,\dot{s}_k)}\left\{l_d'(s_k,\dot{s}_k,u')+G_{k+1}^*(s_{k+1},\dot{s}_{k+1})\right\} \tag{14.46}$$

这是式（14.27）的一种特殊形式。在离散化的 $\dot{s}$ 的值上用线性插值来计算 $G_{k+1}^*(s_{k+1},\dot{s}_{k+1})$。用 s 和 $\dot{s}$ 计算原始状态 $x\in X$ 就可以从 l_d 得到代价项 l_d'，但是如果轨迹片段进入 S_{obs}，那么它的代价为无穷大。计算过程进行直到 $k=1$，此时得到最优尚需代价 $G_1^*(s_1,\dot{s}_1)$。将每个阶段的尚需代价函数当作一个导航函数就可以得到最优轨迹。

动态规划方法具有一般性，它可以扩展到存在更高阶约束情况下的有路径约束的轨迹规划中[882]。例如，如果一个系统为 $q^{(3)}=h(q,\dot{q},\ddot{q},u)$，那么会产生一个三维有路径约束的相空间，其中每个元素表示为 $(s,\dot{s},\ddot{s})$。该空间中的行动是加加速度，对于 $u'\in U'(s,\dot{s},\ddot{s})$，它产生 $s^{(3)}=u'$。

14.6.3.5 时间最优的 bang-bang 法

动态规划方法很有效率，因为它的搜索只限于二维空间。尽管如此，如果 S_{obs} 具有某种特殊结构，那么计算时间最优轨迹可能会更有效率。该思路是寻找一个在两个运动本原之间交替进行的序列：一个是最大加速度，另一个是最大减速度。这种在两个相反极限之间进行的切换通常称为 bang-bang 控制，它通常出现在获得时间最优控制律中（见例 15.4）。这里所介绍的方法是文献[123,881]提出的。获得时间最优轨迹的不足之处在于如果出现误差，该轨迹就无法跟踪（第 14.6.1 节中的第 4 个模块），因为问题的解运动在可达集合的边界上。

方法是为机械臂而设计的，如例 14.7 所述。假设 S_{obs} 是上部以 $\dot{s}_{\max}$ 为界、侧面以 $s=0$ 和 $s=1$ 为界的一个单连通分支。假设 S 的出现只是因为 $\ddot{s}$ 的可允许值的区间趋于零（在这种情况下，$U'(s,\dot{s})$ 成为空集）。还假设 S_{obs} 的下边界可以表示为一个称为极限曲线的可微函数 $\phi:[0,1]\to S$，它为每个 $s\in[0,1]$ 产生最大速度 $\dot{s}=\phi(s)$。文献[881]对该方法进行扩展以处理多障碍物，但是这里不考虑这种情况。还假设对于每个 i，$\mathrm{d}\tau_i/\mathrm{d}s\neq 0$。用该方法也可以处理 $\mathrm{d}\tau_i/\mathrm{d}s=0$ 的情况[880]。

令 $u'_{\min}(s,\dot{s})$ 和 $u'_{\max}(s,\dot{s})$ 分别表示 $(s,\dot{s})\in S$ 的可能的最小和最大加速度。如果 $(s,\dot{s})\notin S_{obs}$，那么 $u'_{\min}(s,\dot{s})<u'_{\max}(s,\dot{s})$。在极限曲线上，$u'_{\min}(s,\phi(s))=u'_{\max}(s,\phi(s))$。只用这种情况下的可行行动产生与极限曲线相切的速度。这称为 ϕ 的一个切点 $(s_{\tan},\dot{s}_{\tan})$。在 S_{obs} 内，不可能有加速度。

图 14.28 描述了 bang-bang 方法，图 14.29 是对其进行的图形说明。假设初始和目标状态分别为(0,0)和(1,0)。步骤 1 通过构造一条终止于(1,0)的最大减速曲线来扩大目标。通过

切换到最大减速上，连接该曲线的轨迹能够最优到达(1,0)。步骤 3 和步骤 4 构造一条最大加速曲线和一条最大减速曲线。因为运行加速曲线会穿过极限曲线。必须避免发生这种违背约束的情况。因此还必须确定一条减速曲线，它使得运动更早离开加速曲线，而又刚好不进入 S_{obs} 的内部。这种曲线必须与极限曲线相切；因此沿着极限曲线对下一个可能切点进行搜索。从那里，步骤 4 采用逆时间积分来生成一条与加速曲线相接触的减速曲线。现在得到部分解，其中加速后面紧跟着一个到达 ϕ 的一个切点的减速。有可能不会用到步骤 4，因为接触到连接目标的曲线。从切点开始，步骤 3 和步骤 4 重复进行直到接触到目标曲线。

bang-bang 方法

1. 从最后状态 (1,0)，对 $\ddot{s} = u'_{\min}(s,\dot{s})$ 进行逆时间积分。用数值方法连续构造曲线直到进入 S_{obs} 的内部或者 $\dot{s} = 0$。在后一种情况下，算法以失败而终止。
2. 令 $(s_{cur}, \dot{s}_{cur}) = (0,0)$。
3. 从 $(s_{cur}, \dot{s}_{cur})$ 对 $\ddot{s} = u'_{\max}(s,\dot{s})$ 进行前向积分直到进入 S_{obs} 内部或者第 1 步中生成的曲线被交叉。在后一种情况下，问题就得到解决。
4. 从第 3 步的轨迹与极限曲线交叉的点开始，发现沿着极限曲线的下一个切点 $(s_{\tan}, \dot{s}_{\tan})$。从 $(s_{\tan}, \dot{s}_{\tan})$，进行逆积分 $\ddot{s} = u'_{\min}(s,\dot{s})$ 直到遇到第 3 步的曲线。令 $(s_{cur}, \dot{s}_{cur}) = (s_{\tan}, \dot{s}_{\tan})$ 并返回到第 3 步。

图 14.28　bang-bang 方法用比动态规划方法更少的搜索来寻找一条时间最优、有路径约束的轨迹

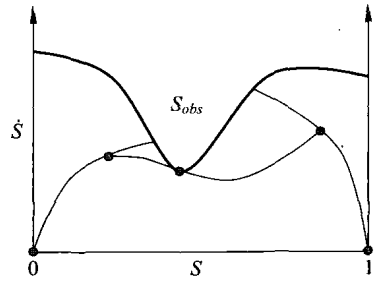


图 14.29　阐述用 bang-bang 方法来计算时间最优轨迹；解轨迹是通过将点连接起来而得到的

14.7　基于梯度的轨迹优化

本节将对运动规划的补充问题进行简要介绍。假设本章的算法返回一个可行行动轨迹。如何改进问题的解？轨迹优化问题指的是在满足所有约束的情况下对轨迹进行扰动以使轨迹的性能得到提高。例如想要缩短 RRT 计算的轨迹，或者想删除由于随机化而引起的行动变化的随意性等。轨迹优化被认为是对运动规划问题的补充，因为它通常要求一个初始猜测，而规划算法可以提供该初始猜测。轨迹优化也可以认为是一种 BVP，只不过它改进提高了初始猜测，而不是从零开始确定轨迹。

对于钢琴搬运工问题，由基于采样的算法计算出来的路径也存在优化问题。但是由于没有微分约束，那么很容易缩短路径。可以采用第 14.6.2 节的规划和变换方法，LPM 只是沿着 C 中最短路径来连接位形对。在有微分约束的情况下，必须面对 BVP。

在更一般的情况下，改进轨迹是非常困难的。关于优化的文献中介绍了大量的方法，参见文献[100,153,667]。本节的目的是通过简要介绍已有的各种方法来促进对它们的进一

步学习，所有这里没有对它们进行详细论述。这些方法属于非线性规划（NLP）或者非线性优化，而不是第 9.3 节中用来发现随机化安全策略的线性规划。优化通常发生在可能轨迹的一个空间中，每个轨迹都是一个时间的函数。因此第 13.4.1 节用到的变分法就与极值有关。那种情形下的泛函 Φ 就是目前情形下的代价泛函 L。系统 $\dot{x}=f(x,u)$ 形成了一个必须满足的附加约束集合，但是 u 可以在优化中进行选择。

为了进行数值计算，可根据一个参数空间来指定一族轨迹。那么优化可以被看作是在满足所有约束的同时，在参数空间中进行的递增搜索。每一步的运动方向是通过计算代价泛函关于参数的梯度来确定的，同时方向被限制在必须与约束条件相切。因此一些非线性规划可以认为是牛顿法或者梯度下降法的一种应用。如同标准优化，代价泛函的二阶导数可以用来指示什么时候应该终止搜索。这些方法的数值计算问题很复杂，有几种 NLP 软件包可供使用，如 NAG Fortran 库或者 Matlab 中的工具包等。

非线性最优控制理论可以看作是 NLP 的一种变形。动态规划递推在连续时间情况下就变成微分方程，哈密顿方程（13.198）是对庞特里亚金最小值原理的推广。这些内容将在第 15.2 节介绍。出现在最小值原理中的额外变量可以认为是一个受约束优化的拉格朗日乘子，其中 $\dot{x}=f(x,u)$ 是约束条件。动态规划或者最小值原理中出现的微分方程很难得到解析解，因此在大多数情况下要用到数值方法。数值动态规划已在第 14.5 节介绍过。

打靶法（shooting methods）构成了一族最简单的轨迹优化方法。作为一个简单例子，假设已经计算了一个形式为：

$$u(t)=w_1+w_2t \tag{14.47}$$

的行动轨迹 $\tilde{u}:[0,t_F]\to\mathbb{R}$，其中 w_1 和 w_2 是某固定参数。考虑对 w_1 和 w_2 进行小量扰动并应用式（14.1）中的积分。如果 f 满足 Lipschitz 条件，那么一个小的扰动对 $\tilde{x}$ 应该产生一个小的变化。用代价泛函计算所产生的新的轨迹以确定轨迹是否得到改进。例如它可能有更低的最大曲率。不采用随机选取扰动的方法，而是计算代价泛函关于参数的梯度。沿着负梯度方向的一个小的参数变化应该减小代价。但是很可能发生的是对 w_1 和 w_2 的扰动将会移动最终状态 $x(t_F)$。通常施加诸如 $x(t_F)=x_G$ 这样的一个终止条件来作为优化中的一个约束条件。这减少了优化的自由度，因此通常需要更多的轨迹参数。

更一般情况下，假设一个运动规划算法基于离散时间模型来计算行动序列。序列中的每个行动在 Δt 时间内保持不变。每个行动的持续时间可以定义为一个受到扰动的参数。每个区间上的每个行动变量 u_i 可以用初始条件为 $w_1=u_i$ 和 $w_2=0$ 的式（14.47）来进行扰动。搜索的维数已经增加，但是也有了更多的自由度。在某些表述中，参数可能是以隐含约束的形式出现。在这种情况下，每次迭代必须解决一个 BVP。最小值原理通常应用在这种情况下[100]。更多关于用打靶法来表述和解决轨迹优化问题的详细内容出现在文献[153]中。

当将打靶法用于在障碍物中间进行轨迹优化时会遇到几个困难。每个扰动都要求进行积分和碰撞检测。对于涉及车辆的问题来说，利用对称性有时可以避免进行积分[199]。例如通过在短时间内改变转向角来扰动 Dubins 车的一条路径，轨迹的其他部分可以用 $SE(2)$ 的一个矩阵来进行变换。一个关键问题是沿负梯度方向缩短路径可能会引起碰撞。将轨迹分成多个片段可以使问题得到缓解，如规划-变换方法，但是这产生了更多的优化问题。另外一种可能的解决方法是对障碍物引入一个罚函数，但是由于局部最小值问题以及缺乏对

X_{obs} 边界的精确表示，所以很难引入罚函数。

打靶法的另外一个困难是在开始时刻附近的一个小的行动变化可能导致后来状态的巨大变化。缓解该问题的一种方法是采用多次打靶法（与目前已经介绍过的单次打靶比较而言）。在这种情况下，初始时将轨迹分成多个片段。这些对应于运动本原序列所施加的时间边界。在这种情况下，假设对每个运动本原分别进行扰动。这种情况需要额外的约束来指示所有轨迹片段必须保持连接。多次打靶法可以推广到被称为 transcription 或 collocation 的方法族。这些方法将轨迹分割成片段，但是每个连接约束沿着轨迹联系更多的点，而不只是片段的端点。Transcription 的一个版本使用隐含约束，它要求使用另外一个 BVP 解算，而另外一个版本使用参数约束，这增加了搜索的维数。后一种情况在实际工作中仍然有用，因为它利用的是快速的、稀疏矩阵计算方法。

轨迹优化方法的主要困难之一是它们在轨迹空间中可能陷入局部最小值点。这意味着它们的行为在很大程度上依赖于初始猜测。对它们来说发现一条与初始轨迹不同伦的轨迹通常是不可能的。它们不能从很糟糕的同伦类中的一个初始猜测中恢复回来。如果 X_{obs} 很复杂，那么这种问题也就变得越发重要。在许多情况下，变分技术不能在单个同伦类中发现一个最优解。如果 X_{free} 的闭包包含正曲率，那么可能存在多个局部最小值点。如果不是这样，那么空间被称为非正弯曲的（NPC）或者 CAT(0)，这个特性可以直接从 X 上的度量来产生[141]。对于这些空间，关于该度量的局部最优轨迹在它的同伦类中总是最好的。

进一步阅读

可达集的特征和计算日益受到广泛的关注[102,104,709,710,917,955]。研究可达性是为了验证能够确保一个控制系统即使在所有可能的干扰下也会产生期望的行为。这实际上可以建模为一个与大自然之间的对策，其中大自然试图将系统带入一个不期望的状态（例如坠落飞机）。关于刻画 X_{ric} 方面的最新进展，参见文献[357]。为完备性而三角化的观点出现在文献[294]中一个相似的情形下。为微分约束下分辨率完备的基于采样的运动规划方法，用离散度和 Lipschitz 条件来表达的精确收敛速率是在文献[198]中予以介绍的。对于控制问题的计算复杂性见文献[116,769]。更多关于规划情形下运动本原的资料见文献[362,364,365,395,790,797,850]。更多关于动态仿真和数值积分的资料见文献[333,443,865]。

第 14.4.1 节是基于文献[290,292,444]。文献[205,239,291,358,362,614,783,999]中有关于更多 kinodynamic 规划的工作。第 14.4.2 节是受文献[74]的启发。第 14.4.3 节来自于文献[613]。更多关于微分约束下 RRT 的工作见文献[140,201,226,326,362,395,512,949]。其他更多关于非完整规划的工作见综述文献[599]和文献[68,279,336,337,356,359,485,582,636,675]。非完整性规划的组合方法出现在文献[13,131,349]中。

第 14.4.5 节是通过将值迭代适应性修改到运动规划问题而产生的。对于带插值的值迭代的一般收敛定理，见文献[170,294,402,568,570]。在文献[170]中考虑的是相空间上的全局约束。文献[610]介绍了这些技术的应用以及类 Dijkstra 算法的发展。在人工智能[725]和控制论[946]中也有相关工作。

第 14.6 节介绍的规划的解耦方法在有关机器人的文献中非常普遍。与规划-变换方法

有关的资料见文献[335,599,861]。更多解耦轨迹规划和时间标定，见文献[355,459,460,846,878,879,882,883]。对于特别强调时间最优轨迹，见文献[106,122,123,788,881,895,880]。

更多关于基于梯度的技术，见文献[100]和其中的参考文献。关于这个方面的经典书籍见文献[153,667]。非完整性规划情形下，文献[199,345,578]用基于梯度的方法解决路径变形问题。

本章介绍的技术在机器人以外的其他领域也非常有用。对于运动规划的航空应用，见文献[88,204,439,440,789]。运动规划问题和技术已在计算机图形学中赢得广泛关注，特别是生成虚拟人（数字演员）动画，这些工作包括文献[35,88,395,502,547,557,560,594,620,652,715,805,980]。在许多工作中，运动捕获（motion capture）是生成运动记录数据库的一种常见的方法，该数据库用作规划方法中的一个运动本原的集合。

习　题

1. 对于 $\mathcal{W}=\mathbb{R}^2$ 中的一个质点，它的每个坐标都建模为一个二重积分器，刻画 X_{ric} 的特性。假设 $u_1=1$ 和 u_2 可以取 $[-1,1]$ 中的任意值。确定 X_{ric}，为了：

（a）$\mathcal{W}$ 中(0,0)点上的一个点障碍物。

（b）$\mathcal{W}$ 中从(0,−1)到(0,1)的一个线段。

根据相变量 $q_1(0)$、$q_2(0)$、$\dot{q}_1(0)$ 和 $\dot{q}_2(0)$ 来刻画问题的解。

2. 扩展二重积分器：

（a）开发三重积分器 $q^{(3)}=u$ 的一个网格，它是从二重积分器网格很自然地扩展而来。

（b）描述如何为高阶积分器 $q^{(n)}$（$n>3$）开发一个网格。

3. 为 Reeds-Shepp 车的三个阶段绘制一个与图 14.6(b)相似的图。

4. 确定如图 14.14 所示的时间受限可达集的上界和下界的表达式。将它们表示成抛物线，其中 $\dot{q}$ 是 q 的函数。

5. 一个可达性的图可以通过在平面上“滚动”一个多面体来产生。例如，假设在一个二维平面上放一个实心正四面体。假设摩擦力很大，推其上部，四面体可以向四个方向的任何一个方向翻倒。为这个问题构造三阶段可达性的图。

6. 为式（13.17）所建模的一个差动机构机器人构造类似于图 14.6(b)中所显示的一个四阶段可达性图。采用三个行动(1,0)、(0,1)和(1,1)。在平面上绘制图并指出每个顶点的位形坐标。

7. 第 14.2.2 节解释了在微分约束下的规划中分辨率完备算法是如何存在的。假设除了有连续状态变量以外，还有第 7.3 节介绍的离散模式来形成一个混合系统。解释如何开发这种情况下的分辨率完备规划算法。扩展如图 14.7 所示的自变量。

实现

8. 在为单积分器、二重积分器和三重积分器而生成的轨迹上，比较欧拉积分和四阶龙格-库塔法的性能和精度。对精度而言，将结果与解析解进行比较。对于在不同条件使用哪

一个提出建议。

9. 产生一些实际轨迹，其中大多数是抛物线，用这些轨迹来改进图 14.13。

10. 在图 14.13 中，状态轨迹片段会随着 $|\dot{x}|$ 的增加而变得更长。设计一个网格，当 $|\dot{x}|$ 增加时通过减小 Δt 来试图使所有的片段尽可能保持相同长度。用不同的方案来实现和试验，并且报告结果。

11. 设计实现近似计算二维多边形世界中的一个质点的时间最优状态轨迹。机器人动力学可以建模为两个独立的二重积分器。通过在 $X = \mathbb{R}^4$ 中搜索二重积分器的网格来求解问题。动画所计算出来的解。

12. 将 RDT 方法应用于一个可以建模为一个带推进器的三维刚体的航天器。编写软件代码计算其无碰撞轨迹。仔细研究与选择 X 上的度量有关的问题。

13. 使用带插值的值迭代方法，求解将 Dubins 车最优地带到一个多边形世界中的目标区域这种问题。

14. 差动机构机器人在多边形环境中推动一个盒子。选择和实现计算推动轨迹的一个规划算法。这在第 13.1.3 节中是作为一个非完整系统的例子而给出的。为了使用 U 上的相关约束，可参见文献[674]。

15. 为用式（13.19）建模的停泊一辆带拖车的车辆，选择和实现计算其轨迹的一个规划算法。在 $\mathcal{W}$ 中设置一个障碍区域，它对应于一个狭窄的停泊空间，改变空隙的大小，并用驱动车辆通过一个障碍物来进行试验。

16. 为式（13.20）中的飞机模型生成可达性图的一个三维透视图。基于飞行向左、向右或者向前直飞和增加、降低或者保持高度等的组合，假设在每个阶段有九个可能的行动。

17. 为例 13.13 给出的二连杆操作臂模型而实现如图 14.27 所示的动态规划算法。

18. 为例 13.13 给出的二连杆操作臂模型而实现如图 14.28 所示的 bang-bang 算法。

19. 对于 Dubins 车（或者另外系统），基于图 14.7 通过在不同步长之间交替切换来生成一个搜索图。在平面中绘制与图的边有关的顶点和状态轨迹。用不同方案在一个矩形区域中生成一个分辨率完备的搜索图，试验并比较结果。

20. 用带插值的值迭代方法来计算 Reeds-Shepp 车的最优尚需代价。绘制尚需代价的水平集，它指示时间受限可达集。将结果与图 14.4 进行比较。

第 15 章　系统理论与分析技术

本章是对第 14 章的补充，将提供用来开发更好局部规划方法（LPM）的工具和概念。很多内容是在控制领域中发展而来的，因为控制论主要关注于刻画特定系统的行为，并在没有障碍物的情况下对这些系统进行控制。用本章的思路，在第 14 章中令人为难的两点边值问题（BVP）将会得到更好的理解和解决。需要注意的是本章不涉及障碍物。尽管这种情况下的规划在第 II 部分中是很平凡的，但是微分约束的出现带来了巨大挑战。

本章将简要介绍相关的概念和技术，以期激发读者对其他相关书籍和文献的进一步学习。现代控制理论的内容非常丰富，本章仅能窥其一斑。第 15.1 节将介绍稳定性和可控性概念，它们都用来刻画到达目标状态的可能性。稳定性刻画向量场的积分曲线在目标点周围的行为，可控性指示是否存在能够到达特定目标的行动轨迹。

第 15.2 节重温动态规划。在这里动态规划变成了一个用最优尚需代价函数表示的偏微分方程。在某些情况下，它有一个封闭形式的解，这与用在计算机科学中的情况不同，在那里是为了得到算法约束。本节还将介绍从动态规划产生的庞特里亚金最小值原理。

本章的其余部分将介绍非完整系统，它在欠驱动机械系统中经常出现。对于 Dubins 车、Reeds-Shepp 车和差动机构，第 15.3 节表达的是点与点之间的最短路径。路径是 BVP 的很好的解，而作为 LPM 时路径特别有价值，例如一些已经用在第 14.6.2 节的规划-变换方法中。第 15.4 节将讨论非完整系统的某些基本性质。最重要的问题是确定非完整约束是否是可积的（它去掉所有 $\dot{x}_i$ 变量）以及刻画由于非完整约束而引起的可达集的特征。第 15.5 节与第 15.3 节的目的相同，只是针对的是更有挑战性的非完整系统。在这些情况下，BVP 问题的解决可能不是最优的，某些方法可能还没有精确地到达目标点。但是在适当的时候，在基于采样的运动规划算法中它们可以用来构造强有力的 LPM。

15.1　系统的基本特性

本节简要介绍控制论中的两个基本概念：稳定性和可控性。它们都可以刻画目标状态是如何到达的。稳定性通常涉及反馈，当时间接近无穷时，它可能仅仅收敛到目标。可控性用来评估是否存在一个导向特定目标状态的行动轨迹。在这两种情况下，在 X 中没有障碍区域。

15.1.1　稳定性

稳定性讨论的是关于一个给定点的向量场的性质。令 X 表示一个平滑流形，向量场就定义在该流形上，X 可能是一个 C-空间或者一个相空间。给定点用 x_G 表示，在运动规划中它可以阐述为目标状态。稳定性刻画的是，通过向量场积分，X 中的其他状态是如何逼近

$\dot{x}_G$ 的。

给定向量场 f 可以认为是一个速度场，表示为：

$$\dot{x} = f(x) \tag{15.1}$$

这看起来像没有行动的状态转移方程。如果给定形如 $\dot{x} = f(x,u)$ 的一个系统，那么通过设计一个反馈规划 $\pi : X \to U$ 就可以确定 u。那么 $\dot{x} = f(x,\pi(x))$，这是 X 上的一个向量，它不再进一步依赖于行动。第 14.5 节中的动态规划方法可以计算这种解。设计一个稳定的反馈规划的过程在控制论中被称为反馈镇定。

平衡点和李亚普诺夫稳定性　如果到达 x_G，至少状态应该保持在 x_G。一个点 $x_G \in X$ 被称为是向量场 f 的一个平衡点（或者不动点），当且仅当 $f(x_G) = 0$。但是这并没有刻画轨迹在 x_G 附近的行为。令 $x_I \in X$ 表示某初始状态，$x(t)$ 指的是从 $x_I = x(0)$ 开始对向量场 f 进行积分后所得到的 t 时刻的状态。

如图 15.1 所示，一个平衡点 $x_G \in X$ 被称为是李亚普诺夫稳定的，如果对于 x_G 的任意开邻域[1] O_1，存在 x_G 的另外一个任意开邻域 O_2，使得 $x_I \in O_2$ 意味着对于所有 $t > 0$ 有 $x(t) \in O_1$。如果 $X = \mathbb{R}^n$，那么通过使用一个用欧氏度量来表达的等价定义就可以得到某种直觉认识。一个平衡点 $x_G \in \mathbb{R}^n$ 被称为李亚普诺夫稳定的，如果对于任意 $t > 0$，存在某 $\delta > 0$ 使得当 $\|x_I - x_G\| < \delta$ 时，有 $\|x(t) - x_G\| < \varepsilon$。这意味着可以选择一个半径任意小且包围 x_G 的球，所有将来的状态都将落在这个球内，只要它们从一个更小半径 δ 的球内开始。如果 δ 的选择不依赖于 ε 和 x，那么平衡点称为一致李亚普诺夫稳定的。

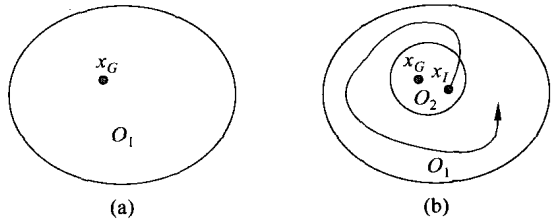


图 15.1　李亚普诺夫稳定性：（a）选择包含 x_G 的任意开集合 O_1；（b）存在某开集合 O_2，从该集合开始的轨迹将不离开 O_1；这里并不要求收敛到 x_G

渐近稳定性　李亚普诺夫稳定是弱稳定的，因为当 t 趋近无穷时，它并不一定收敛于 x_G。仅要求状态在 x_G 附近徘徊。收敛则需要一个更强的概念——渐近稳定性。一个点 x_G 是 f 的一个渐近稳定平衡点，如果：

（1）它是 f 的一个李亚普诺夫平衡点。

（2）存在 x_G 的某开邻域 O，使得对于任意 $x_I \in O$，当 t 趋近无穷时 $x(t)$ 收敛[2]于 x_G。

对于 $X = \mathbb{R}^n$，第二个条件可以表示为：存在某 $\delta > 0$ 使得对于任意 $x_I \in X$ 且 $\|x_I - x_G\| < \delta$，当 t 趋近无穷时状态 $x(t)$ 收敛于 x_G。渐近稳定性需要两个条件。第一个条件限制积分曲线摆动范围的大小，而范围的大小并未被第二个条件所确定。

渐近稳定性是一个合理的要求，但是它并没有规定用多长时间收敛。如果 x_G 是渐近稳定的，并且存在某 $m > 0$ 和 $\alpha > 0$ 使得：

1　一个点 x 的一个开邻域是指包含 x 的一个开集。

2　这种收敛可以用 X 上的度量 ρ 来进行评估计算。

$$\|x(t)-x_G\| \leq me^{-\alpha t}\|x_I-x_G\| \tag{15.2}$$

那么称之为指数稳定。这为表达收敛速度提供了一个简便方法。

对于运动规划来说，甚至指数收敛也可能不够强。这种问题在第 8.4.1 节讨论过。例如实际上通常期望在有限时间内到达 x_G，而不是在极限情况下“到达” x_G。有两个普通的解决方法。其一是用渐近稳定性，如果状态到达预先确定的包围 x_G 的一个小球时就宣布到达目标。在这种情况下，如果 x_G 是渐近稳定的，那么总能够在有限时间内到达扩大后的目标。其二是要求一种更强的稳定性，其中必须在有限时间内精确到达 x_G。但是这必须使用诸如图 8.5(b)的向内流动这样的不连续的向量场。大多数控制论专家对此感到烦恼，因为执行这种轨迹通常要求无穷大能量。但在另一方面，在某些应用中不连续向量场可能是一种比较适宜的表示方法，如第 8 章所述。注意到如果没有反馈，这种问题似乎并不重要。期望第 14 章所设计的状态轨迹在有限时间内到达目标。如果没有反馈，那么就不要期望周围的向量场能够保持连续性或者平滑性。第 15.1.3 节将介绍可控性，这是基于在有限时间内到达目标，但是它也基于一个给定系统 $\dot{x}=f(x,u)$ 的轨迹的存在性，而不是一个向量场 $x=f(x)$ 的轨迹族。

时变向量场 这里所表达的稳定性概念通常在时变情形 $\dot{x}=f(x,t)$ 中进行介绍。因为本书中的绝大多数规划问题都是时不变的，所以表示都仅限于时不变向量场。但是在寻找一个能够镇定系统的反馈规划中存在一个特性。Brockett 条件指的是对于存在连续、时变反馈规划的某时不变系统，却不存在一个连续、时不变反馈规划[145,158,996]。这包括无漂移控制系统，如简易汽车和独轮车。这意味着为了保持向量场的连续性，必须引入时间来使得当接近 x_G 时，向量场能够发生改变。如果向量场的连续性并不重要，那么就不存在这个问题。

吸引域 目前的稳定性的定义通常称为是局部的，因为它们都是用 x_G 的一个邻域来表达的。将邻域扩展到所有 X 就可以定义全局稳定性。一个平衡点是全局渐近稳定的，如果它是李亚普诺夫稳定的，并且当时间趋近无穷时从任意 $x_0 \in X$ 开始的积分曲线都收敛于 x_G。事实是仅有 X 的某适当子集中的点才能够收敛于 x_G。X 的收敛于 x_G 的所有点的集合通常称为 x_G 的吸引域。第 8.5.1 节的漏斗是基于吸引域。与此相关的还有第 14.2.1 节的后向可达集。在那里考虑的是在有限时间内通向 x_G 的行动轨迹。对于吸引域来说，只假设渐近收敛于 x_G，并且给定向量场（不选择行动）。

极限环 对于某些向量场来说，状态可能被吸引到一个极限环内。并不是稳定到一个点，状态轨迹收敛到 X 中的一条环形路径。例如它们可能收敛到如下一个圆。在各种可能发生振荡的机械系统中都会发生这种情况。文献[44]中介绍了某些基本问题以及 $X=\mathbb{R}^2$ 情况下的几个例子。

15.1.2 李亚普诺夫函数

假设给定一个速度场 $\dot{x}=f(x)$ 和一个平衡点 x_G。是否能够很容易地确定各种形式的稳定性？证明稳定性的最有力的一种方法是构造一个李亚普诺夫函数。稍后将介绍该方法，这里首先简要介绍一些其他方法。

如果 $f(x)$ 是线性的，这意味着对于某 $n\times n$ 常数矩阵 A 和 $X=\mathbb{R}^n$，有 $f(x)=Ax$，那么关于原点 $x_G=0$ 的稳定性问题可以用 A 的特征值来解答[194]。状态 $x=0$ 是渐近稳定的，当且仅当 A 的所有特征值有负实部。考虑标量情况 $\dot{x}=ax$，它的 $X=\mathbb{R}$ 且 a 为常数。该微分方程的解为 $x(t)=x(0)e^{at}$，只有 $a<0$ 才能使解收敛于 0。这很容易扩展到 $X=\mathbb{R}^n$ 且 A 是一个 $n\times n$ 的对角矩阵的情况，对角矩阵的每个对角元素（或者特征值）是负的。对于更一般的矩阵，实数或者复数特征值就决定了稳定性（复数特征值引起振荡）。也存在李亚普诺夫稳定性条件。$\dot{x}=Ax$ 的每个平衡点状态都是李亚普诺夫稳定的，如果 A 的所有特征值都有非正实部，并且零实部特征值是 A 的特征多项式的不同的根。

如果 $f(x)$ 是非线性的。那么稳定性有时可以用 $f(x)$ 关于 x_G 线性化并进行线性稳定性分析来推断。但是在许多情况下该步骤并不能确定稳定性（见文献[158]的第 6 章）。证明一个向量场的稳定性对于大多数非线性系统来说是很困难的。一个方法是基于 LaSalle 不变性原理[39,158,187]，它对于判断收敛于任意多个目标状态非常有用（见文献[848]的第 5.4 节）。其他主要方法是构造一个李亚普诺夫函数，它被当作一个中间工具来间接建立稳定性。如果这种方法失败，那么仍然可能用其他方法来判断稳定性。因此，它是稳定性的一个充分条件，而不是必要条件。

确定稳定性　假设给定一个速度场 $\dot{x}=f(x)$ 和一个平衡点 x_G。令 ϕ 表示一个候选李亚普诺夫函数，它将用作建立 f 稳定性的一个辅助工具。必须为特定向量场 f 确定一个适当的 ϕ。这里并不详细讨论它。从某种意义上，步骤可以用“猜测和验证”来刻画，这是微分方程的许多求解技术所描述的方法。如果 ϕ 成功建立稳定性，那么就称之为 f 的李亚普诺夫函数。

重要的是刻画 ϕ 在由 f 所引起的流动方向上是如何变化的。用李导数来计算：

$$\dot{\phi}(x)=\sum_{i=1}^{n}\frac{\partial\phi}{\partial x_i}f_i(x) \tag{15.3}$$

这产生了一个新的函数 $\dot{\phi}(x)$，它指示 ϕ 沿着 $\dot{x}=f(x)$ 方向的变化。

需要几个概念来确定稳定性。一个函数 $h:[0,\infty)\to[0,\infty)$ 将被称为一个斜坡，如果它是连续的、严格递增的且 $h(0)=0$。可以认为它是一个一维导航函数，只在目标 0 处有一个最小值点。一个函数 $\phi:X\to[0,\infty)$ 将被称为局部正定的，如果存在某开集 $O\subseteq X$ 和一个斜坡函数 h 使得对于所有 $x\in O$，有 $\phi(x_G)=0$ 和 $\phi(x)\geq h(\|x\|)$。如果选择 O 为 $O=X$，并且如果 X 是有界的，那么 ϕ 被称为是全局正定的或者正定的。由于 X 的拓扑，在某些空间中这是不可能的，当构造第 8.4.4 节中的导航函数时会出现这种问题。如果 X 是无界的，那么当 $\|x\|$ 趋近无穷时 h 还必须趋近无穷以产生一个正定 ϕ[848]。对于 $X=\mathbb{R}^n$，一个有正定矩阵 M 的二次型 x^TMx 是一个全局正定函数。这启发了在李亚普诺夫稳定性分析中采用二次型形式。

现在可以论述李亚普诺夫定理[158,848]。假设 ϕ 在 x_G 是局部正定的。如果存在一个开集 O 使得 $x_G\in O$，并且在所有 $x\in O$ 上 $\dot{\phi}(x)\leq 0$，那么 f 是李亚普诺夫稳定的。如果 $-\dot{\phi}(x)$ 在 O 上也是局部正定的，那么 f 则是渐近稳定的。如果 ϕ 和 $\dot{\phi}$ 都是全局正定的，那么 f 则是全局渐近稳定的。

例 15.1（通过李亚普诺夫函数建立稳定性） 令 $X=\mathbb{R}$。令 $\dot{x}=f(x)=-x^5$，选择李亚普诺夫函数 $\phi(x)=\frac{1}{2}x^2$。李导数（15.3）产生 $\dot{\phi}(x)=-x^6$。显然 ϕ 和 $\dot{\phi}$ 都是全局正定的，因此 0 是 f 的一个全局渐近稳定点。 ■

规划中的李亚普诺夫函数 李亚普诺夫函数与规划中的导航函数和最优尚需代价函数的关系非常密切。在第 2.3 节和第 8.2 节的最优离散规划问题中，尚需付出的代价值可以被认为是一个离散李亚普诺夫函数。采取所计算的行动，在搜索图上可以假设存在一种离散向量场。采取的每个最优行动都会降低最优尚需代价值，直到在目标处成为 0。最优尚需代价和李亚普诺夫函数都能够确保轨迹不陷入局部最小值。李亚普诺夫函数比尚需代价函数更加一般，因为李亚普诺夫函数不要求最优性。它们更像第 8 章中的导航函数。对第 8.2.2 节离散导航函数的要求就如同这里所给出的正定条件。假设图 8.3 所示的导航函数是对 $\mathbb{R}^2$ 上的一个李亚普诺夫函数的一个离散近似。一般情况下，一个李亚普诺夫函数指示与 x_G 的某种距离，尽管它可能不是最优的。尽管如此，它仍然是基于向 x_G 的单调接近过程。因此在第 14 章的许多基于采样的规划算法中它都可以用作一个距离函数。因为它服从系统的微分约束，所以它提供一种比忽略了这些考虑的欧几里得度量更好的、关于在规划中如何取得进步的方法。李亚普诺夫函数在第 14.4.3 节的 RDT 方法中特别有价值，因为 RDT 非常依赖于 X 上的距离函数。

15.1.3 可控性

现在假设在一个平滑流形 X 上给定一个系统 $\dot{x}=f(x,u)$。系统可以认为是向量场的一个参数化的族，其中的参数是 u。对于稳定性来说，假设该参数由一个反馈规划来确定以得到某 $\dot{x}=f(x)$。本节讨论可控性，它表示从其他状态应用一个行动轨迹 $\tilde{u}$ 是否能够到达某一状态。回顾第 14.2.1 节的可达集的定义会对此有所帮助。

经典可控性 令 $\mathcal{U}$ 表示系统的可允许行动轨迹。在默认情况下它可以是使式（14.1）能够积分的任何 $\tilde{u}$。一个系统 $\dot{x}=f(x,u)$ 是可控的，如果对于所有 $x_I, x_G \in X$，存在一个时间 $t>0$ 和行动轨迹 $\tilde{u} \in \mathcal{U}$ 使得从 $x(0)=x_I$ 进行积分得到 $x(t)=x_G$。可控性也可以用第 14.2.1 节的可达集来表达。如果对于所有 $x_I, x_G \in X$，有 $x_G \in R(x_I, \mathcal{U})$，那么系统是可控的。

如果没有障碍物的任何运动规划问题的解存在，则一个系统是可控的。也就是说，存在两点边值问题（BVP）的解。

例 15.2（经典可控性） 第 13.1.2 节中的所有车辆模型都是可控的。例如在一个无穷大的平面内，Dubins 车可以在任意两个位形之间运动。但需要注意的是，如果平面被障碍物所限制，那么 Dubins 车不一定是可控的。考虑一个不可控的系统的例子，令 $X=\mathbb{R}$、$\dot{x}=u$ 和 $U=[0,1]$。在这种情况下，状态不能减小。例如不存在驱动状态从 $x_I=1$ 到 $x_G=0$ 的行动轨迹。 ■

控制论的书籍中介绍了许多确定系统可控性的方法。如果系统是线性的，式（13.37）给出的维数为 m 和 n 的系统是可控的，当且仅当 $n \times nm$ 可控性矩阵：

$$M=\left[B \vdots AB \vdots A^2B \vdots \cdots \vdots A^{n-1}B\right] \tag{15.4}$$

是满秩的[194]。这被称为卡尔曼秩条件[504]。如果系统是非线性的，那么可以计算系统线

性化后的可控性矩阵。满秩条件是可控性的充分条件（见文献[848]中的命题 11.2）。如果不满秩，系统仍然可能是可控的。某些非线性系统具有这样的特性，即它们可能在看起来不被允许的方向上产生运动。例如第 13.1.2 节中的简易汽车不能侧滑，但是可以通过“之”字形机动方式来使车辆侧向移动。第 15.4 节将介绍确定这种系统可控性的方法。

对于 $\ddot{q}=h(q,\dot{q},u)$ 形式的全驱动系统，可以通过将其转化为第 14.4.1 节的二重积分器的形式来确定其可控性。令系统表示为 $\ddot{q}=u'$，其中 $u'\in U'(q,\dot{q})$。如果 $U'(q,\dot{q})$ 是包含 $\mathbb{R}^n$ 原点的一个开邻域，并且对于任意 $x\in X$ 都可以使用相同的邻域，那么系统是可控的。如果一个非线性系统是欠驱动的，如简易汽车，那么可控性问题就变得相当复杂。下面的概念适合这种系统。

STLC：处理障碍物的可控性　目前所讨论的可控性概念并不关心到达 x_G 之前轨迹在 X 中运动了多远。但是对于欠驱动系统和在障碍物中间进行的规划来说，这个问题就非常重要了。该问题很自然地产生这样一个疑问：是否存在适合障碍物的一种可控性？应该能够断言：在没有微分约束的情况下如果一个状态是可达的，那么对给定系统 $\dot{x}=f(x,u)$ 来说它也是可达的。这可以用时间受限可达集来表达。令 $R(x,\mathcal{U},t)$ 表示在小于等于 t 的时间内从 x 所能够到达的所有状态的集合。系统 $\dot{x}=f(x,u)$ 被称为是从 x_I 短时局部可控的（small-time locally controllable，STLC），如果存在某 $t>0$ 使得对于所有 $t'\in(0,t]$，有 $x_I\in\text{int}(R(x_I,\mathcal{U},t'))$（这里的 int 表示一个集合的内部，如第 4.1.1 节所定义）。如果系统 $\dot{x}=f(x,u)$ 从每个 $x_I\in X$ 都是 STLC 的，那么整个系统是 STLC 的。

考虑用该定义来解答上面的问题。因为 $\text{int}(R(x_I,\mathcal{U},t'))$ 是一个开集，那么必然存在某小量 $\epsilon>0$，使得开球 $B(x_I,\epsilon)$ 是 $\text{int}(R(x_I,\mathcal{U},t'))$ 的一个真子集。如图 15.2 所示，可以到达 $B(x_I,\epsilon)$ 的边界上的任意点，这意味着在任意方向上都可以进行一次步长为 ϵ 的过程，即使存在微分约束。但是如果有障碍物，那么我们必须仔细工作，以使从 x_I 到 $B(x_I,\epsilon)$ 的表面的轨迹不绕得太远。

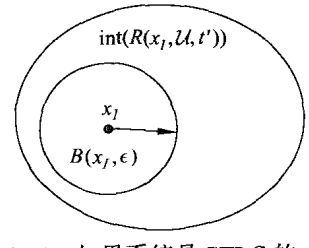


图 15.2　如果系统是 STLC 的，那么运动可以在任意短的时间内向任意方向进行

假设有个障碍区域 X_{obs}，以及给定一个在 t_F 时刻终止于 x_G 内的无违规状态轨迹 $\tilde{x}$，它不一定满足给定系统。如果系统是 STLC 的，那么基于 $\tilde{x}$，总可能发现另外一个满足微分约束的轨迹。应用第 14.6.2 节的规划-变换方法。假设用二叉递归重分来为可能的替换选择区间。也假设存在计算任意两个状态之间最短轨迹的 LPM。这个轨迹忽略了障碍物，但是服从微分约束。初始情况下，$[0,t_F]$ 被 LPM 产生的一个轨迹所代替，并且如果它不是无违规的，那么 $[0,t_F]$ 细分为 $[0,t_F/2]$ 和 $[t_F/2,t_F]$，在更小的区间上再进行替换。递归地使用这种思路直到最后区间足够小到一定是无违规的为止。

最后这一点是 STLC 的特性所致。不管区间多么小，都一定存在一个替换轨迹。如果一个区间很大，那么就有足够的时间偏离原始轨迹很远。但是当时间区间减小时，就没有太多时间偏离原始轨迹。这种假设 f 有比较合适的条件使得它是 Lipschitz 的。假设轨迹被一个半径为 ε 的无碰撞管道所保护。因此沿着轨迹的所有点距离 X_{free} 的边界至少是 ε。选择时间区间足够小以确保轨迹偏离原始轨迹不超过 ε。因此对在有障碍物情况下的规划来

说 STLC 是系统所拥有的一个非常重要的性质。为确定一个非线性系统是否是 STLC 的，第 15.4 节将介绍一些数学工具。

与可控性密切相关的一个概念是可达性，它仅与可达集的维数有关。令 n 为 X 的维数。如果存在某 $t>0$ 使得 $R(x_I,\mathcal{U},t)$ 的维数为 n，那么称系统是从 x_I 可达的。另外，这也可以表达为要求 $\text{int}(R(x_I,\mathcal{U},t))\neq\phi$。

例 15.3（可达性） 回顾第 13.1.3 节的系统，其中状态在一个圆上。在这种情况下，$X=\mathbb{R}^2$，并且状态转移方程为 $\dot{x}=yu$ 和 $\dot{y}=-xu$。该系统不是可达的，因为可达集的维数为 1。 ■

通过要求存在某 t 使得对于所有 $t'\in(0,t]$，有 $\text{int}(R(x_I,\mathcal{U},t))\neq\phi$，这样就可以定义短时可达性。可达性对有漂移的系统来说特别重要。

15.2 连续时间动态规划

在本书中动态规划是一个不断重复出现的主题。目前它总是采用在某阶段序列上计算最优尚需代价（或者已付代价）函数的形式。已经形成值迭代和类 Dijkstra 算法。在计算机科学中，在研究计算问题最优解的算法的过程中动态规划是基础，但是动态规划的初衷是用来解决最优控制问题的[86]。在这种情形下，离散阶段集合被连续时间代替。动态规划递归是一个偏微分方程，被称为 Hamilton-Jacobi-Bellman（HJB）方程。HJB 方程可以用数值算法来解决，但是在某些情况下，它可以产生解析解[1]。第 15.2.2 节简要介绍了线性系统情况下的解析解。第 15.2.3 节将介绍庞特里亚金最小值原理，它可以从动态规划原理产生，并将哈密顿力学中进行的优化进行了推广。

15.2.1 Hamilton-Jacobi-Bellman 方程

Hamilton-Jacobi-Bellman（HJB）方程是最优控制理论中重要的结果。一些其他原理和设计技术都从 HJB 方程而来，HJB 方程仅仅是动态规划原理在连续时间情况下的一种叙述。HJB 方程的推导过程都超出了本书的范围。在这里即将给出的是与规划有关的一种时不变表述，并且给出一种非正式的推导，这基于文献[97]。

15.2.1.1 离散情形

在进入连续范畴之前，首先描述离散规划的概念。回顾第 2.3 节，如果 X、U 和阶段是离散的，那么最优规划在搜索图中可以用值迭代或者 Dijkstra 算法来解决。平稳最优尚需代价函数 G^* 可以用作一个导航函数，它对最优反馈规划进行编码。这是第 8.2.2 节所描述的，一个例子如图 8.3 所示。

假设在表述 8.1（或者表述 2.3）下已经计算了 G^*。令状态转移方程表示为：

$$x'=f_d(x,u) \tag{15.5}$$

那么 G^* 的动态规划递归是：

1 令计算机科学家感到惊讶的是，这种情况下的动态规划不产生算法，而是产生了问题的一个封闭形式的解。

$$G^*(x)=\min_{u\in U(x)}\{l(x,u)+G^*(x')\} \tag{15.6}$$

它被认为是一种离散形式的 HJB 方程。但是为了获得对即将介绍的概念的一些认识，还要进行进一步的处理。

令 u^* 表示式（15.6）求最小值中所应用的最优行动。假定 u^* 是最优行动，但是还需要在式（15.6）中加以检验。如果它真是最优的，那么：

$$G^*(x)=l(x,u^*)+G^*(f_d(x,u^*)) \tag{15.7}$$

这已经被认为是庞特里亚金最小值原理的一种离散形式，它将在第 15.2.3 节出现。对式（15.7）进行移项，得到：

$$G^*(f_d(x,u^*))-G^*(x)=-l(x,u^*) \tag{15.8}$$

在单个阶段，当 G^* 用作一个导航函数时（式（15.8）乘以–1），最优尚需代价降低 $l(x,u^*)$。当沿着最优路径向目标前进一步时，可以精确得出最优单阶段代价。在最好方向上移动的同时，尚需代价函数中的这种增量变化形成了 HJB 方程和最小值原理的基础。

15.2.1.2　连续情形

现在考虑连续情形。假设 X 和 U 都是连续的，但是阶段保持离散，并且式（15.5）～式（15.8）仍然成立。它们目前的形式可以被用离散时间近似的任何系统所使用。假设用第 14.2.2 节的离散时间模型来近似状态空间 X 上的一个系统 $\dot{x}=f(x,u)$，其中 X 是一个平滑流形。那么 U 离散化为 U_d，但是这里仍然用其原来的形式表示。令 Δt 代表离散时间。

用离散时间模型近似式（15.6），并且令 Δt 趋近于零，这样就会得到 HJB 方程。这里的变量很不正式，详见文献[97,573,913]。用离散时间近似，动态规划递归是：

$$G^*(x)=\min_{u\in U(x)}\{l_d(x,u)+G^*(x')\} \tag{15.9}$$

其中 l_d 是对第 k 阶段所积累代价的一个离散时间近似，它是：

$$l_d(x,u)\approx l(x,u)\Delta t \tag{15.10}$$

假设当 Δt 趋近于零时，总的离散化的代价收敛于其连续时间下的积分代价。

采用在 x 处泰勒级数展开的线性部分，$G^*(x')$ 项可以近似为：

$$G^*(x')\approx G^*(x)+\sum_{i=1}^{n}\frac{\partial G^*}{\partial x_i}f_i(x,u)\Delta t \tag{15.11}$$

这是用 x 处的切平面来近似 $G^*(x')$。将式（15.10）和式（15.11）代入式（15.9），那么：

$$G^*(x)\approx\min_{u\in U(x)}\left\{l(x,u)\Delta t+G^*(x)+\sum_{i=1}^{n}\frac{\partial G^*}{\partial x_i}f_i(x,u)\Delta t\right\} \tag{15.12}$$

式（15.12）的两边都减去 $G^*(x)$ 就得到：

$$\min_{u\in U(x)}\left\{l(x,u)\Delta t+\sum_{i=1}^{n}\frac{\partial G^*}{\partial x_i}f_i(x,u)\Delta t\right\}\approx 0 \tag{15.13}$$

当 Δt 趋近于零时取极限，然后除以 Δt，这样就产生了 HJB 方程：

$$\min_{u\in U(x)}\left\{l(x,u)+\sum_{i=1}^{n}\frac{\partial G^*}{\partial x_i}f_i(x,u)\right\}=0 \tag{15.14}$$

将 HJB 方程与式（15.6）的离散时间情形相比较。它们都表示在最好的方向上运动时

代价是如何变化的。将最优行动u^*代入式（15.14）就产生了：

$$\sum_{i=1}^{n}\frac{\partial G^*}{\partial x_i}f_i(x,u^*)=-l(x,u^*) \tag{15.15}$$

这只是式（15.8）的连续时间版本。在目前情形下，该式的左边表示沿着从 x 应用最优行动所得到的方向尚需代价函数的导数。

HJB 方程和指定最后阶段代价的一个边界条件就足以刻画规划问题最优解的特征。因为它是在整个状态空间上表示的，所以 HJB 方程的解就产生最优反馈规划。非常遗憾，HJB 方程在大多数情况不能得到解析解。因此需要用到第 14.5 节的值迭代方法等数值求解方法。但是有一类重要问题，它们可以直接用 HJB 方程来解决，参见第 15.2.2 节。

15.2.1.3 HJB 方程的变分

存在几个版本的 HJB。其中式（15.14）比较适合第 14 章所表达的规划问题。如果尚需代价函数与时间有关，那么 HJB 方程是：

$$\min_{u\in U(x)}\left\{l(x,u,t)+\frac{\partial G^*}{\partial t}+\sum_{i=1}^{n}\frac{\partial G^*}{\partial x_i}f_i(x,u,t)\right\}=0 \tag{15.16}$$

并且G^*是 x 和 t 的函数。仍然可以用泰勒级数展开来产生，但是变量为 x 和 t。最优控制理论的大多数书籍提到的都是这种形式的 HJB 方程，或者只是将$\partial G^*/\partial t$移到 min 的外面后再移到方程的右边：

$$\min_{u\in U(x)}\left\{l(x,u,t)+\sum_{i=1}^{n}\frac{\partial G^*}{\partial x_i}f_i(x,u,t)\right\}=-\frac{\partial G^*}{\partial t} \tag{15.17}$$

在微分对策论中，HJB 方程被推广到 Hamilton-Jacobi-Isaacs（HJI）方程[60,480]。假设给定系统如式（13.203）并且用$l(x,u,v,t)$形式的代价项来定义一个零和对策。HJI 方程就刻画了鞍点的特征，它表示为：

$$\min_{u\in U(x)}\max_{v\in V(x)}\left\{l(x,u,v,t)+\frac{\partial G^*}{\partial t}+\sum_{i=1}^{n}\frac{\partial G^*}{\partial x_i}f_i(x,u,v,t)\right\}=0 \tag{15.18}$$

$$\max_{v\in V(x)}\min_{u\in U(x)}\left\{l(x,u,v,t)+\frac{\partial G^*}{\partial t}+\sum_{i=1}^{n}\frac{\partial G^*}{\partial x_i}f_i(x,u,v,t)\right\}=0 \tag{15.19}$$

这些方程与式（15.16）非常相似，并且，将 min 和 max 算子进行交换类似于第 9.3 节中的鞍点的定义。

15.2.2 线性二次型问题

本节描述的问题能够直接求解HJB方程产生封闭形式的解，而不是计算数值解的算法。假设一个线性系统由式（13.37）给出，它要求指定矩阵 A 和 B。任务是设计一个反馈规划，从任意初始状态它都能够渐近镇定系统。这是一个无限范围问题，不用终止行动。

要求得到关于二次型代价泛函的最优解。令 Q 为一个非负定[1]的$n\times n$矩阵，令 R 为一个正定的$n\times n$矩阵。二次型代价泛函定义为：

1 非负定意味着对于所有 $x\in\mathbb{R}^n$，有 $x^TQx\geq 0$。正定意味着对于所有 $x\in\mathbb{R}^n$，有 $x^TRx>0$。

$$L(\tilde{x},\tilde{u})=\frac{1}{2}\int_0^{\infty}\left(x(t)^T Qx(t)+u(t)^T Ru(t)\right)\mathrm{d}t \tag{15.20}$$

为了保证产生有限代价的解的存在性，必须对矩阵做几个假设。(A,B) 必须是可稳定的，(A,Q) 必须是可检测的，特定条件和解的推导过程见文献[28]。

HJB 可以用来产生代数 Riccati 方程：

$$SA+A^TS-SBR^{-1}B^TS+Q=0 \tag{15.21}$$

其中除 S 以外的所有矩阵都已经给出。有求解 S 的方法，在非负定 $n\times n$ 矩阵空间中它是唯一的。

线性向量场：

$$\dot{x}=\left(A-BR^{-1}B^TS\right)x \tag{15.22}$$

是渐近稳定的（矩阵的所有特征值都具有负实部）。得到这个向量场，如果用如下定义的一个反馈规划 π 来选择 u：

$$\pi(x)=-R^{-1}B^TSx \tag{15.23}$$

反馈规划 π 实际上是最优的，最优尚需代价为：

$$G^*(x)=\frac{1}{2}x^TSx \tag{15.24}$$

因此对于二次型代价的线性系统来说，得到的是解析解，而不需要借助数值近似。非常遗憾，这种求解技术并不能推广到非线性系统或有障碍物的系统。因此第 14 章的规划方法是有意义的。

尽管这里给出的解也确实存在一些变化和扩展，但是都是仅对于二次型代价的线性系统而言的。在某种情况下，应用 HJB 方程可以得到 Riccati 方程的某些变化形式。在文献[28]中产生了时变系统的解。如果在预测中存在高斯不确定性，那么得到线性二次型高斯（LQG）问题[567]。甚至对于式（13.204）形式的微分对策，也存在线性二次型问题及其解[60]。

15.2.3　庞特里亚金最小值原理

庞特里亚金最小值原理[1]与 HJB 方程有非常密切的关系，它提出了最优轨迹必须满足的条件。但是需要记住的是最小值原理提出的是关于最优性的必要条件，而不是充分条件。而 HJB 方程提出的是充分条件。单独使用最小值原理通常不能确定轨迹是最优的。但是在某些情况下，它对于发现候选最优轨迹是十分有用的。不能满足最小值原理的任何轨迹都不是最优的。

为了理解最小值原理，首先回到离散规划的情况。最小值原理本质上是由式（15.7）给出的。这可以认为是 HJB 方程在应用最优行动 u^* 这种特殊情况下的具体化。这使得 min 消失，而且 HJB 方程的全局特性也随之消失。最小值原理表达的是沿着最优轨迹的条件，而不是整个状态空间上的尚需代价函数。因此在最好情况下它只能保证在可能轨迹空间中的局部最优性。

连续情况下的最小值原理本质上是由式（15.15）给出的，它类似于式（15.7）。但是它

1　这通常被称为庞特里亚金最大值原理，因为庞特里亚金最初将它定义为最大值[804]。在大多数关于控制的文献中的哈密顿函数是庞特里亚金的哈密顿函数的取负，因此它是最小化。两个名称都可以使用。

通常采用第 13.4.4 节的伴随变量和一个哈密顿函数来表达。

令 λ 表示一个 n 维向量，它被定义为：

$$\lambda_i = \frac{\partial G^*}{\partial x_i} \tag{15.25}$$

哈密顿函数被定义为：

$$H(x,u,\lambda) = l(x,u) + \sum_{i=1}^{n} \lambda_i f_i(x,u) \tag{15.26}$$

这是应用式（15.25）所定义的伴随变量以后 HJB 方程（15.14）的 min 内部的精确表达式。这可以与第 13.4.4 节中式（13.192）所给出的哈密顿函数相比较（那里的 ρ 变成这里的 λ）。两者并不十分相同，但是它们都受到相同基本原理的启发。

在执行最优行动轨迹 $\tilde{u}^*$ 时，HJB 方程意味着对于所有 $t \geqslant 0$：

$$H(x(t),u^*(t),\lambda(t)) = 0 \tag{15.27}$$

这只是表达式（15.15）的另外一种方法。H 保持固定常数这个事实很像一个守恒定理，这是第 13.4.4 节中哈密顿力学的基础。在最小值原理中使用哈密顿函数更具有一般性。

使用 HJB 方程（15.14），最优行动为：

$$u^*(t) = \arg\min_{u \in U(x)} \{H(x(t),u(t),\lambda(t))\} \tag{15.28}$$

也就是说，哈密顿函数在 $u(t) = u^*(t)$ 处最小。

到目前还没有讨论的是 λ 如何随时间而演化。如下形式的系统：

$$\dot{\lambda} = g\left(x,\lambda,u^*\right) \tag{15.29}$$

可以由对哈密顿函数（或者 HJB 方程）关于 x 求导来得到。这产生了两个耦合的系统 $\dot{x} = f(x,u^*)$ 和式（15.29）。它们实际上可以解释为 $2n$ 维相空间中的单个系统，其中每个相向量是 (x,λ)。这类似于第 13.4.4 节对哈密顿力学的解释，在那里产生了式（13.198）。

式（15.25）定义 λ 只是为了跟踪 G^* 的变化。用式（15.29）进一步明确表示它。假设一个反馈规划选择 u^* 产生 $u^* = \pi^*(x)$。那么哈密顿函数可以认为只是 x 和 λ 的一个函数。在这种假设下，哈密顿函数（15.26）对 x_i 求导就产生了：

$$\frac{\partial l\left(x,\pi^*(x)\right)}{\partial x_i} + \sum_{j=1}^{n} \frac{\partial \lambda_j}{\partial x_i} f_j\left(x,\pi^*(x)\right) + \sum_{j=1}^{n} \lambda_j \frac{\partial f_j\left(x,\pi^*(x)\right)}{\partial x_i} \tag{15.30}$$

求导的有效性要求一个引理来断言可以忽略 $\pi(x)$ 的导数（见文献[97]的引理 3.3.1），并且假设 U 在随后的自变量中是凸的，即使能够证明最小值原理不要求如此。

式（15.30）中的第二项实际上是 $\dot{\lambda}_i$。λ_i 关于状态的全微分是：

$$\mathrm{d}\lambda_i = \sum_{j=1}^{n} \frac{\partial \lambda_i}{\partial x_j} \mathrm{d}x_j \tag{15.31}$$

两边除以 $\mathrm{d}t$ 后得到：

$$\frac{\mathrm{d}\lambda_i}{\mathrm{d}t} = \sum_{j=1}^{n} \frac{\partial \lambda_i}{\partial x_j} \frac{\mathrm{d}x_j}{\mathrm{d}t} = \sum_{j=1}^{n} \frac{\partial \lambda_i}{\partial x_j} \dot{x}_j \tag{15.32}$$

每个 $\dot{x}_j$ 由状态转移方程给出：$\dot{x}_j = f_j(x,\pi^*(x))$。因此：

$$\dot{\lambda}_i = \frac{\mathrm{d}\lambda_i}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial G^*}{\partial x_i} = \sum_{j=1}^{n}\frac{\partial \lambda_i}{\partial x_j} f_j(x, \pi^*(x)) \tag{15.33}$$

将式（15.33）代入式（15.30），并将方程置为零（因为哈密顿函数沿着最优轨迹的值为零）就产生了：

$$\frac{\partial l\left(x,\pi^*(x)\right)}{\partial x_i} + \dot{\lambda}_i + \sum_{j=1}^{n}\lambda_j \frac{\partial f_j\left(x,\pi^*(x)\right)}{\partial x_i} = 0 \tag{15.34}$$

求解 $\dot{\lambda}_i$ 得到：

$$\dot{\lambda}_i = -\frac{\partial l\left(x,\pi^*(x)\right)}{\partial x_i} - \sum_{j=1}^{n}\lambda_j \frac{\partial f_j\left(x,\pi^*(x)\right)}{\partial x_i} \tag{15.35}$$

这与：

$$\dot{\lambda}_i = -\frac{\partial H}{\partial x_i} \tag{15.36}$$

相同，正如所料，这产生了伴随方程。

$\dot{x} = f(x,u)$ 和式（15.36）确定了最小值原理给出的系统的演化过程。这类似于第 13.4.4 节给出的哈密顿方程（13.198）。那里的广义动量变成这里的伴随变量。

当应用最小值原理时，通常要求最优行动在所有时间都必须满足式（15.28）。通常情况下，这可等价地被表达为：

$$H(x(t),u^*(t),\lambda(t)) \leqslant H(x(t),u(t),\lambda(t)) \tag{15.37}$$

即当偏离最优行动时，哈密顿函数值会增加或者保持不变（哈密顿函数值不能减小）。

例 15.4（二重积分器的最优规划） 回顾例 13.3 的二重积分器系统。令 $\ddot{q} = u$、$\mathcal{C} = \mathbb{R}$ 和 $U = [-1,1] \cup \{u_T\}$。假设存在一个在 $\mathbb{R}$ 中运动的质点。行动就是一个方向上的力，其值最大为 1。状态转移方程为 $\dot{x}_1 = x_2$、$\dot{x}_2 = u$ 和 $X = \mathbb{R}^2$。任务是在任意两个状态 $x_I, x_G \in X$ 之间进行最优运动规划。从给定初始状态 x_I 开始，必须在最短时间内到达目标状态 x_G。这种情况下的代价泛函定义为 $l(x,u) = 1$，对于所有 $x \in X$ 和 $u \in U$ 且 $u \neq u_T$。

使用式（15.26），哈密顿函数被定义为：

$$H(x,u,\lambda) = 1 + \lambda_1 x_2 + \lambda_2 u \tag{15.38}$$

由式（15.28）可以得到最优行动轨迹：

$$u^*(t) = \operatorname*{arg\,min}_{u\in[-1,1]}\{1 + \lambda_1(t)x_2(t) + \lambda_2(t)u(t)\} \tag{15.39}$$

如果 $\lambda_2(t) < 0$，那么 $u^*(t) = 1$；如果 $\lambda_2(t) > 0$，那么 $u^*(t) = -1$。因此如果 $\lambda_2(t) \neq 0$，那么行动可以指定为 $u^*(t) = -\mathrm{sgn}(\lambda_2(t))$。这两种情况是第 14.6.3 节的 bang-bang 控制，它们也是第 14.4.1 节中规划算法所用的极值行动。在 $\lambda_2(t) = 0$ 这种边界情况下，可以选择 $[-1,1]$ 中的任意行动。

剩下的任务是确定伴随变量的值。$\dot{\lambda}_1 = 0$ 和 $\dot{\lambda}_2 = -\lambda_1$ 时的式（15.36）就成为伴随方程。解是 $\lambda_1(t) = c_1$ 和 $\lambda_2(t) = c_2 - c_1 t$，其中 c_1 和 c_2 是常数，它们由 $t = 0$ 时的式（15.38）和式（15.39）来确定。最优行动仅取决于 $\lambda_2(t)$ 的符号。因为它的解是一条直线的方程，所以它最多只能改变一次符号。因此根据 x_I 和 x_G 的不同，可能有四种类型的解：

（1）一直进行加速 $u^*(t) = 1$。

（2）一直进行减速 $u^*(t)=-1$。

（3）加速到某个时刻 t'，然后进行减速直到最后时刻。

（4）减速到某个时刻 t'，然后进行加速直到最后时刻。

对于最后两种情况，t' 通常被称为切换（开关）时刻，在该时刻 $\tilde{u}^*$ 发生突变。它们都是第 14.6.3 节描述的 bang-bang 控制的解。■

这可能是一个最简单的例子，很容易发现最优解，因为伴随变量是时间的线性函数。第 15.3 节将介绍 Dubins 车、Reeds-Sheep 车、差动机构的最优解，它们都是用最小值原理以及某些几何自变量来建立的。当系统变得复杂时，这种分析就变得很困难。在这些情况下，必须使用第 14 章的基于采样的方法来确定最优轨迹。

一个公共的难点在于沿着解轨迹存在奇异弧（singular arc）。它们是在某时间段上 H 关于 u 的一种退化。例如当 H 与 u 无关时就会发生这种情况。在例 15.4 中，当 $\lambda_2(t)=0$ 时 H 就与 u 无关；但是没有奇异弧，因为这仅发生在一个瞬间时刻。如果持续时间更长些，那么将会有一个无法确定最优行动的时间区间。一般情况下，如果 H 关于 u 的 Hessian 矩阵是正定的，那么就没有奇异弧（这通常被称为 Legendre-Clebsch 条件）。这种情况下的最小值原理为在可能状态轨迹空间中的局部最优性提供了一个充分条件。如果对于某区间 $[t_1,t_2]$ 且 $t_1<t_2$，Hessian 不是正定的，那么需要更多信息来确定从 $x^*(t_1)$ 到 $x^*(t_2)$ 这段奇异弧上的最优轨迹。

注意到所有这些分析都没有考虑障碍物。没有采用措施来防止解进入一个障碍区域。可以调整行动集合 $U(x)$ 和代价 $l(x,u)$ 来考虑障碍物；但是此时不可能用最小值原理确定一个最优解，除非在某种特殊情况下。

还有其他方法来产生最小值原理。回顾第 13.4.4 节哈密顿方程可以从欧拉-拉格朗日方程产生，那么最小值原理可以用变分法来产生也就不足为奇了[97,792]。最小值原理可以认为是一种形式的约束优化。这就将 λ 作为拉格朗日乘子来解释。为进一步学习最小值原理，可参考庞特里亚金的原始文献[804]。

时间最优 对最优化到达目标的时间[427,904]这种情况，最小值原理有非常有趣的解释。在这种情况下，式（15.26）中 $l(x,u)=1$，并且忽略代价项。对于其余部分，令 λ 定义为：

$$\lambda_i=-\frac{\partial G^*}{\partial x_i} \tag{15.40}$$

而不是用式（15.25）。那么哈密顿函数可以表达为：

$$H(x,u,\lambda)=\sum_{i=1}^{n}\lambda_i f_i(x,u)=\left\langle -\frac{\partial G^*}{\partial x}, f(x,u)\right\rangle \tag{15.41}$$

它是 $f(x,u)$ 和 G^* 负梯度的内积。用式（15.40），哈密顿函数将被最大化，而不是被最小化（这等价于庞特里亚金的原始表述[804]）。当两个向量的方向接近于平行时，它们的内积会增加。优化式（15.41）也就是选择 u 使得 $\dot{x}$ 尽可能接近于 G^* 的最速下降方向。通过分析可达集 $R(x_0,\mathcal{U},t)$ 的边界在 X 是如何传播的，这就能得到很好的说明。由定义，$\partial R(x_0,\mathcal{U},t)$ 上的点必须对应于最优时间轨迹，并且 $\partial R(x_0,\mathcal{U},t)$ 可以被认为是一个垂直于 $-\partial G^*/\partial x$ 的传播波前。最小值原理指出：应该选择 u 使得 $\dot{x}$ 向内指向传播边界，并尽可能接近正交[427]。

15.3　一些轮式车辆的最优路径

对于第 13.1.2 节的某些轮式车辆模型来说，任意两个位形之间的最短路径已经被完全描绘。在这种情况下，$X = C = \mathbb{R}^2 \times \mathbb{S}^1$，这对应于平面内一个刚体的 C-空间。对于每个模型，必须仔细定义 C 中的路径长度以在车辆所运动的世界 $\mathcal{W} = \mathbb{R}^2$ 中保持某种物理意义。例如对于简易汽车，可优化后轮轴中心在 $\mathcal{W}$ 中运动的距离。如果恰当安排坐标系，那么这就是在不考虑方向的情况下优化 C 的 $\mathbb{R}^2$ 子空间中的路径长度。需要注意的是本节给出的解依赖于所要优化的代价泛函。

第 15.3.1 节～第 15.3.3 节分别介绍 Dubins 车、Reeds-Sheep 车、差动机构模型的最短路径。在每种情况下，路径都可以描述为多个运动本原的组合。由于对称性，描述从一个固定初始位形 $q_I = (0,0,0)$ 到任意目标位形 $q_G \in C$ 的最优路径就足够了。如果想要得到从不同 $q_I \in C$ 开始的最优路径，那么需要对 q_I 和 q_G 进行刚体变换（很容易对整个路径进行平移和旋转，而不影响其最优性，只要 q_G 相对 q_I 没有发生运动）。也可以固定 q_G，考虑从所有可能的 q_I 开始的最优路径。

一旦 q_I（或者 q_G）固定，那么可以将 C 分拆为对应于 q_G（或者 q_I）位置集合的胞腔。在每个胞腔内，最优曲线用参数化的运动本原的一个固定序列来描述。例如 Dubins 车的一个胞腔指示“左转”、“直行”，然后“右转”。曲线适合用作基于采样的规划算法中的一个 LPM。

本节主要着力于提出问题的解。这涉及证明它们的正确性，并且证明过程还需基于第 15.2.3 节的庞特里亚金最小值原理。其他重要组成部分有 Filipov 存在性定理（见文献[904]）和 Boltyanskii 关于最优性的充分条件（它也证明了动态规划的合理性）[132]。关于第 15.3.1 节和第 15.3.2 节中提出的曲线的更多详细内容和说明出现在文献[904,905,924]中。与第 15.3.3 节的曲线有关的详细内容将出现在文献[65]中。

15.3.1　Dubins 曲线

回顾第 13.1.2 节给出的 Dubins 简易汽车。系统由式（13.15）给出。这里假设汽车以恒定速度 $u_s = 1$ 向前运动。另外一个重要的约束是最大转向角 $\phi_{\max}$，它产生最小转弯半径 $\rho_{\min}$。当车行驶时，考虑绑在后轮轴中心处的一支铅笔在 $\mathcal{W} = \mathbb{R}^2$ 中所绘出的曲线的长度。这是图 13.1 中载体坐标系原点的位置。任务是当汽车在任意 q_I 和 q_G 之间行驶时，最小化曲线的长度。由于 $\rho_{\min}$ 的缘故，这可以认为是一个有界曲率最短路径问题。如果 $\rho_{\min} = 0$，那么没有曲率界限，最短路径是 $\mathbb{R}^2$ 中的一条直线。根据（8.39）形式的代价泛函，优化指标为：

$$L(\tilde{q},\tilde{u}) = \int_0^{t_F} \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}\,\mathrm{d}t \tag{15.42}$$

其中 t_F 是到达 q_G 的时间，一个位形表示为 $q = (x,y,\theta)$。如果不能到达 q_G，那么假设 $L(\tilde{q},\tilde{u}) = \infty$。

因为速度是固定的，那么系统简化为：

$$\begin{aligned}\dot{x}&=\cos\theta\\ \dot{y}&=\sin\theta\\ \dot{\theta}&=u\end{aligned}\tag{15.43}$$

其中 u 在区间 $U=[-\tan\phi_{\max},\tan\phi_{\max}]$ 中进行选择。这意味着式（15.42）简化为优化到达 q_G 的时间 t_F，因为被积函数简化为 1。为方便起见，假设 $\tan\phi=1$。如下结果对于任意 $\phi_{\max}\in(0,\pi/2)$ 也是成立的。

文献[296]指出：在任意两个位形之间，Dubins 车的最短路径总可以表示为不超过三个运动本原的一个组合。每个运动本原在一个时间区间上应用一个常数行动，并且按照最短路径行进时仅需要的行动是 $u\in\{-1,0,1\}$。本原和它们的符号如图 15.3 所示。S 本原驱动汽车向前直线行驶。L 和 R 本原分别向左和右转尽可能急的弯。用这些符号，每条最短路径都可以指定为一个有三个符号的序列，它们对应于应用本原的顺序。令这种序列称为一个字。前后两个连续本原不必是相同类型的，因为如果相同，那么它们可以合并为一个。基于这种认识，那么长度为 3 的字可能有 10 个。Dubins 指出其中六个可能是最优的：

符号	u
S	0
L	1
R	−1

图 15.3 三个运动本原，它们可以构造 Dubins 车的所有最优曲线

$$\{LRL,RLR,LSL,LSR,RSL,RSR\}\tag{15.44}$$

任意两个位形之间的最短路径总可以用其中一个字来刻画。它们被称为 Dubins 曲线。

为更加准确，每个本原的持续时间也应该确定。对于 L 或者 R，令下标表示在本原应用期间所累积的旋转的总数。对于 S，令下标表示运动总距离。用这些下标，Dubins 曲线可以更加准确地表示为：

$$\{L_\alpha R_\beta L_\gamma,R_\alpha L_\beta R_\gamma,L_\alpha S_d L_\gamma,L_\alpha S_d R_\gamma,R_\alpha S_d L_\gamma,R_\alpha S_d R_\gamma\}\tag{15.45}$$

其中 $\alpha,\gamma\in[0,2\pi)$、$\beta\in(\pi,2\pi)$、$d\geqslant 0$。图 15.4 说明了两种情况。注意到 β 必须大于 π（如果小于 π，那么某其他字将变成最优的）。

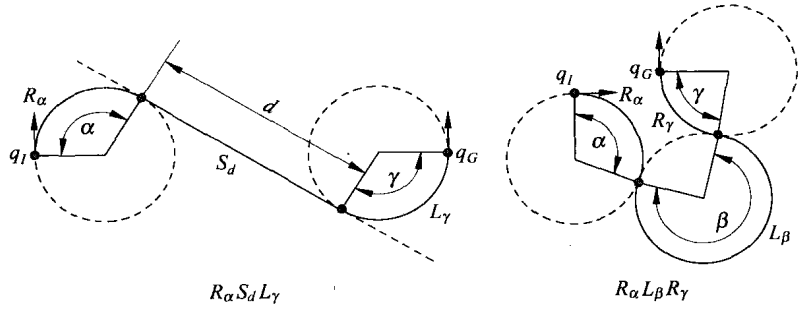


图 15.4 在 $\mathcal{W}=\mathbb{R}^2$ 中两个字的轨迹

构造压缩形式的字来将性质上相似的路径聚集在一起，这样做是很方便的。在第 15.3.2 节介绍 Reeds-Shepp 曲线时特别有用，因为它们有 46 条，而不是像 Dubins 曲线那样只有 6 条。令 C 表示一个“转弯”符号，代表 R 或者 L。用 C，式（15.44）中的六个字可以被压缩为两个基字：

$$\{CCC, CSC\} \tag{15.46}$$

在这种压缩形式下，需要记住的是连续两个 Cs 必须是不同的转弯（不允许 RR 和 LL）。这样，基字可以更准确地表示为：

$$\{C_\alpha C_\beta C_\gamma, C_\alpha S_d C_\gamma\} \tag{15.47}$$

其中 $\alpha, \gamma \in [0, 2\pi)$、$\beta \in (\pi, 2\pi)$、$d \geq 0$。

到目前已经为刻画最短路径而提供了大量的信息，但是对于给定的 q_I 和 q_G 来说，还有两个问题：

（1）式（15.45）中的六个字中哪个能够产生 q_I 和 q_G 之间的最短路径？

（2）对于特定字来说，下标 α, β, γ 和 d 的值是多少？

为了将 Dubins 曲线用作一个 LPM，应该有效地解答这些问题。一个简单方法是尝试所有六个字并选择最短的一个。每个字的参数可以通过描绘从 q_I 到 q_G 的最小半径圆来确定，如图 14.23 所示。另外一种方法是用区域的特点，在该区域上的一个特定字是最优的。假设 q_G 固定在 $(0,0,0)$。基于 q_I 的可能位置，C-空间可以分拆为胞腔，对于这些胞腔来说同一个字是最优的。文献[904]准确给出了胞腔和它们的边界。例如对于 $\theta = \pi$ 的胞腔剖分如图 15.5 所示。

除了可以用作一个 LPM 以外，最短路径所产生的代价可能是一些基于采样的规划算法中的一个非常有用的距离函数。这有时被称为 Dubins 度量（但是它不是一个真正意义上的度量，因为它违背了对称公理）。这可以被认为是最优尚需代价 G^*。它已经用第 14.5 节中的动态规划方法近似计算出来，但是通过仔细分析，已经知道精确值。Dubins 度量的一个有趣性质是它不连续，如图 15.6 所示。将采用 R 本原行进 $\pi/2$ 距离所付出的代价与行进到要求转弯半径比用 R 本原的转弯半径更小的一个附近点所付出的代价进行比较。在 U 中不存在所要求的行动，所以不得不用更长的本原序列才能到达目标点。Dubins 车不具备第 15.1.3 节的 STLC 性质，所以产生了 G^* 中的不连续性。对于 STLC 系统，G^* 是连续的。

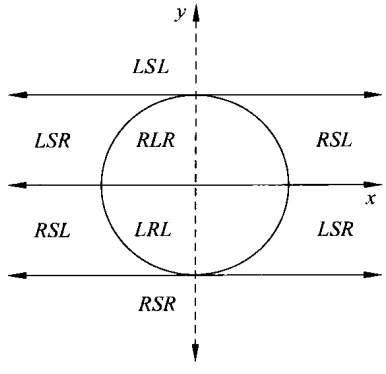


图 15.5　分拆为不变字胞腔在 $\theta = \pi$ 处的一个以原点为中心的圆片

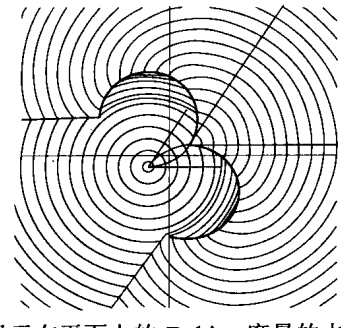


图 15.6　显示在平面上的 Dubins 度量的水平集。沿着两个圆弧，度量是不连续的（由 Philippe Souères 提供）

15.3.2 Reeds-Shepp 曲线

现在考虑 Reeds-Shepp 车的最短路径，它与 Dubins 车之间仅有的不同之处在于它能够向后行驶。需要优化相同的指标（见式（15.42）），它是后轮轴中心所行驶的距离。最短路径等价于最短时间内行驶的路径，如同 Dubins 车。提高式（15.43）中的简化系统可以得到：

$$\begin{aligned}\dot{x} &= u_1 \cos\theta \\ \dot{y} &= u_1 \sin\theta \\ \dot{\theta} &= u_1 u_2\end{aligned} \tag{15.48}$$

其中 $u_1 \in \{-1,1\}$ 和 $u_2 \in [-\tan\phi_{\max}, \tan\phi_{\max}]$。第一个行动变量 u_1 选择齿轮向前（$u_1 = 1$）或者向后（$u_1 = -1$）。为简单起见，仍然假设 $u_2 \in [-1,1]$。这里叙述的结果可以应用于任意 $\phi_{\max} \in (0, \pi/2)$。

文献[817]指出：描述 Reeds-Shepp 车最短路径的字不超过 48 个。第 15.3.1 节的基字符号可以扩展来表达最短路径。字中用一个新的符号“|”来表明“齿轮”从向前转换到向后或者从向后转换到向前。Reeds-Shepp 车最短路径总可以用如下基字中的一个来表达：

$$\{C|C|C, CC|C, C|CC, CSC, CC_\beta|C_\beta C, C|C_\beta C_\beta|C, C|C_{\pi/2}SC, CSC_{\pi/2}|C, C|C_{\pi/2}SC_{\pi/2}|C\} \tag{15.49}$$

执行最短路径需要五个本原。在某些情况下给出下标 $\pi/2$，因为必须跟踪半径为 $\pi/2$ 的曲线。对其他一些情况，给出下标 β 表示它必须匹配另外一个本原的参数。式（15.49）中给出的形式类似于 Dubins 车的式（15.46）。也可以指定参数范围来产生一种类似于式（15.47）的形式。结果如图 15.7 所示。这两种情况下例子的曲线如图 15.8 所示。

Base	α	β	γ	d
$C_\alpha \vert C_\beta \vert C_\gamma$	$[0,\pi]$	$[0,\pi]$	$[0,\pi]$	—
$C_\alpha \vert C_\beta C_\gamma$	$[0,\beta]$	$[0,\pi/2]$	$[0,\beta]$	—
$C_\alpha C_\beta \vert C_\gamma$	$[0,\beta]$	$[0,\pi/2]$	$[0,\beta]$	—
$C_\alpha S_d C_\gamma$	$[0,\pi/2]$	—	$[0,\pi/2]$	$(0,\infty)$
$C_\alpha C_\beta \vert C_\beta C_\gamma$	$[0,\beta]$	$[0,\pi/2]$	$[0,\beta]$	—
$C_\alpha \vert C_\beta C_\beta \vert C_\gamma$	$[0,\beta]$	$[0,\pi/2]$	$[0,\beta]$	—
$C_\alpha \vert C_{\pi/2} S_d C_{\pi/2} \vert C_\gamma$	$[0,\pi/2]$	—	$[0,\pi/2]$	$(0,\infty)$
$C_\alpha \vert C_{\pi/2} S_d C_\gamma$	$[0,\pi/2]$	—	$[0,\pi/2]$	$(0,\infty)$
$C_\alpha S_d C_{\pi/2} \vert C_\gamma$	$[0,\pi/2]$	—	$[0,\pi/2]$	$(0,\infty)$

图 15.7 Reeds-Shepp 最优曲线的每个运动本原参数的区间范围

现在用指定特定运动本原来使得基字更加准确。假设为 Dubins 车构造一个类似于式（15.44）的字的列表。这六个本原如图 15.9 所示。还使用符号 S、L 和 R。为了指示向前或者向后的齿轮，用如图 15.9 所示的上标＋和－[1]。

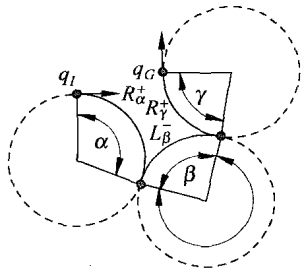


图 15.8　$R_\alpha^+ L_\beta^- R_\gamma^+$ 曲线的一个例子；这通过向后行驶来生成一条比图 15.4（b）中 Dubins 车的曲线短的曲线

Symbol	Gear：u_1	Steering：u_2
S^+	1	0
S^-	−1	0
L^+	1	1
L^-	−1	1
R^+	1	−1
R^-	−1	−1

图 15.9　六个运动本原，用它们可以构造 Reeds-Shepp 车的所有最优曲线

图 15.10 显示了 48 个不同的字，它们产生于用式（15.49）中的 C、S 和“|”表达的未压缩的基字。每条最短路径是一个长度最多为 5 的字。这比 Dubins 车的字多。用运动本原，（15.49）中的每个基字都可以扩展到四个或者八个字。为了不压缩每个基字，R 和 L 仍不能连续使用。这为前六个压缩字产生了四种可能性。其余三个涉及一个中间 S 本原，该本原允许 Rs 和 L 的八个可能的序列。48 个字的其中两个在文献[924]中被排除。其余 46 个字对于一条最短路径来说都可能发生，它们被称为 Reeds-Shepp 曲线。

基　字	运动本原的序列
$C\|C\|C$	$(L^+R^-L^+)(L^-R^+L^-)(R^+L^-R^+)(R^-L^+R^-)$
$CC\|C$	$(L^+R^+L^-)(L^-R^-L^+)(R^+L^+R^-)(R^-L^-R^+)$
$C\|CC$	$(L^+R^-L^-)(L^-R^+L^+)(R^+L^-R^-)(R^-L^+R^+)$
CSC	$(L^+S^+L^+)(L^-S^-L^-)(R^+S^+R^+)(R^-S^-R^-)$ $(L^+S^+R^+)(L^-S^-R^-)(R^+S^+L^+)(R^-S^-L^-)$
$CC_\beta\|C_\beta C$	$(L^+R_\beta^+L_\beta R^-)(L^-R_\beta^-L_\beta^+R^+)(R^+L_\beta^+R_\beta^-L^-)(R^-L_\beta^-R_\beta^+L^+)$
$C\|C_\beta C_\beta\|C$	$(L^+R_\beta^-L_\beta^-R^+)(L^-R_\beta^+L_\beta^+R^-)(R^+L_\beta^-R_\beta^-L^+)(R^-L_\beta^+R_\beta^+L^-)$
$C\|C_{\pi/2}SC$	$(L^+R_{\pi/2}^-S^-R^-)(L^-R_{\pi/2}^+S^+R^+)(R^+L_{\pi/2}^-S^-L^-)(R^-L_{\pi/2}^+S^+L^+)$ $(L^+R_{\pi/2}^-S^-L^-)(L^-R_{\pi/2}^+S^+L^+)(R^+L_{\pi/2}^-S^-R^-)(R^-L_{\pi/2}^+S^+R^+)$
$CSC_{\pi/2}\|C$	$(L^+S^+L_{\pi/2}^+R^-)(L^-S^-L_{\pi/2}^-R^+)(R^+S^+R_{\pi/2}^+L^-)(R^-S^-R_{\pi/2}^-L^+)$ $(R^+S^+L_{\pi/2}^+R^-)(R^-S^-L_{\pi/2}^-R^+)(L^+S^+R_{\pi/2}^+L^-)(L^-S^-R_{\pi/2}^-L^+)$
$C\|C_{\pi/2}SC_{\pi/2}\|C$	$(L^+R_{\pi/2}^-S^-L_{\pi/2}^-R^+)(L^-R_{\pi/2}^+S^+L_{\pi/2}^+R^-)$ $(R^+L_{\pi/2}^-S^-R_{\pi/2}^-L^+)(R^-L_{\pi/2}^+S^+R_{\pi/2}^+L^-)$

图 15.10　Reeds 和 Shepp 的 48 条曲线；Suaamann 和 Tang[924]显示第一行中的 $(L^-R^+L^-)$ 和 $(R^-L^+R^-)$ 可以去除；因此只需要 46 个字来描述最短路径

1　这不同于文献[904]中所用到的符号。因此 r^- 对应于这里的 L^-。这里的 L 意味着转向轮是为左转的，但是汽车处在向后行驶状态。这有助于实现 R 和 L 不能在一个字中连续出现。

当用作一个 LPM 时仍然会出现为 q_I 和 q_G 确定特定字和参数的问题。这对于 Dubins 车来说并不困难，但是现在有 46 种可能性。对每个字进行测试并选择最短一个这种朴素策略所付出的代价可能过于昂贵。文献[904]给出 C 中准确的胞腔边界，每个字就在这个 C 上。胞腔边界十分复杂，这使得点定位算法很难实现。裁剪掉一些字的一种简单方法是用 θ 的有效性区间。对于某些 θ 值，某些压缩字不可能是最短路径。文献[904]给出在 θ 的范围内起作用的那些字的一个表格。最短路径的长度可以用作采样规划算法中的一个距离函数。所产生的 Reeds-Shepp 度量是连续的，因为 Reeds-Shepp 车是 STLC 的，它将在第 15.4 节被介绍。

15.3.3 Balkcom-Mason 曲线

近些年来，另外两个最优曲线族得到确定[65,213]。回顾第 13.1.2 节中的差动机构系统，它出现在许多可移动机器人系统中。在许多方面，差动机构是一种特殊形式的简易汽车。式（13.17）所给出的系统的表达式可以构造得与式（15.48）中的 Reeds-Shepp 车相同。例如令 $r=1$ 和 $l=1$，通过 $u_\omega = u_1$ 和 $u_\psi = u_1 u_2$ 可以使它们等价。考虑用式（15.42）来表示差动机构轴的中心所行进的距离。很容易最小化任意 q_I 和 q_G 之间的距离，如图 13.4 所示。可以使中心点沿 $\mathcal{W}=\mathbb{R}^2$ 中的一条直线运动。这对于 Reeds-Shepp 车来说是可能的，如果 $\rho_{\min}=0$，也就是说 $\phi_{\max}=\pi/2$。因此多年来对于差动机构来说不存在特别有趣的曲线。

但是测量轴的中心所行驶的距离所存在的问题是纯旋转不花代价。当车轮以相同的速度、相反的角速度旋转时就会发生这种情况。中心不发生移动，这里忽略了持续时间、能量消耗和车轮的旋转。将它们中的一个或者多个融入代价泛函，就会产生一个非常具有挑战性的优化问题。Balkcom 和 Mason 限制了差动机构的速度以及最小化了从 q_I 到 q_G 行进的总时间。用式（13.16），行动集合被定义为 $U=[-1,1]\times[-1,1]$，其中每个车轮最大转速为 1（不失一般性，也可以选择其他值作为界限）。优化指标为：

$$L(\tilde{q},\tilde{u}) = \int_0^{t_f} \left(\sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} + \left| \dot{\theta}(t) \right| \right) \mathrm{d}t \qquad (15.50)$$

这里考虑了式（15.42）所忽略的 θ。该指标等价于最小化到达 q_G 的时间。所产生的模型称为 Balkcom-Mason 机构。如果指标采用车轮总的旋转数，那么这就产生了另外一个最优曲线族[213]。

文献[65]指出，只需要图 15.11 所示的四个运动本原来表达差动机构机器人的最优时间路径。每个本原对应于在一个时间区间内将一个行动变量固定在它的界限值。用图 15.11 中的符号（文献[65]也使用它），可以形成描述最优路径的字。字长不超过 5。因此任意最

Symbol	Left wheel:u_l	Right wheel:u_r
⇑	1	1
⇓	−1	−1
↶	−1	1
↷	1	−1

图 15.11 四个运动本原，用它们可以构造差动机构的所有最优曲线

短路径都可以表示为一个分段固定的行动轨迹，其中段数不超过 5。每段对应于图 15.11 中的一个本原。

在 Balkcom-Mason 机构的情况下基字和本原使用相同的符号，这是很方便的。对每个基字应用对称变换来产生八个字的族，它们精确地确定了运动本原序列。九个基字描述了最短路径：

$$\{\curvearrowright,\Downarrow,\Downarrow\curvearrowright,\curvearrowright\Downarrow\curvearrowright,\Uparrow\curvearrowright_{\pi}\Downarrow,\curvearrowleft\Downarrow\curvearrowright,\Downarrow\curvearrowright\curvearrowright,\curvearrowleft\Downarrow\curvearrowright\Uparrow,\Uparrow\curvearrowleft\Downarrow\curvearrowright\Uparrow\} \tag{15.51}$$

这类似于式（15.46）和式（15.49）给出的压缩形式。运动如图 15.12 所示。

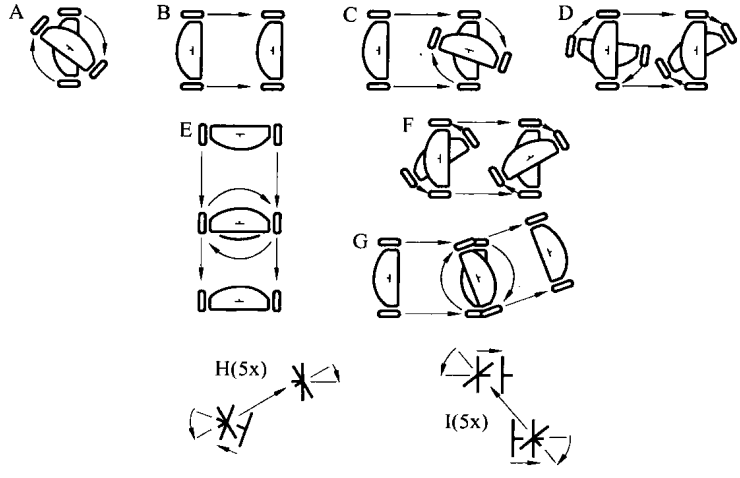


图 15.12　刻画九个基字[65]：对于小的运动来说只有最后两个字有效，它们被放大五倍，没有绘出机器人的形状

图 15.13 显示了 40 个不同的 Balkcom-Mason 曲线，它们是将对称变换应用于式（15.51）的基字产生而来。图 15.13 中有 72 项，但是有许多是相同的。T_1 变换指示⇑和⇓的方向从基字发生翻转。T_2 变换将颠倒运动本原的顺序。T_3 变换将翻转$\curvearrowleft$和$\curvearrowright$的方向。有七种可能的变换组合，它们构成了图 15.13 中的一行。

	Base	T_1	T_2	T_3	T_2oT_1	T_3oT_1	T_3oT_2	$T_3oT_2oT_1$
A.	$\curvearrowright$	$\curvearrowright$	$\curvearrowright$	$\curvearrowleft$	$\curvearrowright$	$\curvearrowleft$	$\curvearrowleft$	$\curvearrowleft$
B.	$\Downarrow$	$\Uparrow$	$\Downarrow$	$\Downarrow$	$\Uparrow$	$\Uparrow$	$\Downarrow$	$\Uparrow$
C.	$\Downarrow\curvearrowright$	$\Uparrow\curvearrowright$	$\curvearrowright\Downarrow$	$\Downarrow\curvearrowleft$	$\curvearrowright\Uparrow$	$\Uparrow\curvearrowleft$	$\curvearrowleft\Downarrow$	$\curvearrowleft\Uparrow$
D.	$\curvearrowright\Downarrow\curvearrowright$	$\curvearrowright\Uparrow\curvearrowright$	$\curvearrowright\Downarrow\curvearrowright$	$\curvearrowleft\Downarrow\curvearrowleft$	$\curvearrowright\Uparrow\curvearrowright$	$\curvearrowleft\Uparrow\curvearrowleft$	$\curvearrowleft\Downarrow\curvearrowleft$	$\curvearrowleft\Uparrow\curvearrowleft$
E.	$\Uparrow\curvearrowleft_{\pi}\Downarrow$	$\Downarrow\curvearrowleft_{\pi}\Uparrow$	$\Downarrow\curvearrowleft_{\pi}\Uparrow$	$\Uparrow\curvearrowright_{\pi}\Downarrow$	$\Uparrow\curvearrowleft_{\pi}\Downarrow$	$\Downarrow\curvearrowright_{\pi}\Uparrow$	$\Downarrow\curvearrowright_{\pi}\Uparrow$	$\Uparrow\curvearrowright_{\pi}\Downarrow$
F.	$\curvearrowleft\Downarrow\curvearrowright$	$\curvearrowleft\Uparrow\curvearrowright$	$\curvearrowright\Downarrow\curvearrowleft$	$\curvearrowright\Downarrow\curvearrowleft$	$\curvearrowright\Uparrow\curvearrowleft$	$\curvearrowright\Uparrow\curvearrowleft$	$\curvearrowleft\Downarrow\curvearrowright$	$\curvearrowleft\Uparrow\curvearrowright$
G.	$\Downarrow\curvearrowright\Uparrow$	$\Uparrow\curvearrowright\Downarrow$	$\Uparrow\curvearrowright\Downarrow$	$\Downarrow\curvearrowleft\Uparrow$	$\Downarrow\curvearrowright\Uparrow$	$\Uparrow\curvearrowleft\Downarrow$	$\Uparrow\curvearrowleft\Downarrow$	$\Downarrow\curvearrowleft\Uparrow$
H.	$\curvearrowleft\Downarrow\curvearrowright\Uparrow$	$\curvearrowleft\Uparrow\curvearrowright\Downarrow$	$\Uparrow\curvearrowright\Downarrow\curvearrowleft$	$\curvearrowright\Downarrow\curvearrowleft\Uparrow$	$\Downarrow\curvearrowright\Uparrow\curvearrowleft$	$\curvearrowright\Uparrow\curvearrowleft\Downarrow$	$\Uparrow\curvearrowleft\Downarrow\curvearrowright$	$\Downarrow\curvearrowleft\Uparrow\curvearrowright$
I.	$\Uparrow\curvearrowleft\Downarrow\curvearrowright\Uparrow$	$\Downarrow\curvearrowleft\Uparrow\curvearrowright\Downarrow$	$\Uparrow\curvearrowright\Downarrow\curvearrowleft\Uparrow$	$\Uparrow\curvearrowright\Downarrow\curvearrowleft\Uparrow$	$\Downarrow\curvearrowright\Uparrow\curvearrowleft\Downarrow$	$\Downarrow\curvearrowright\Uparrow\curvearrowleft\Downarrow$	$\Uparrow\curvearrowleft\Downarrow\curvearrowright\Uparrow$	$\Downarrow\curvearrowleft\Uparrow\curvearrowright\Downarrow$

图 15.13　用对称类进行排序的差动机构的 40 种最优曲线类型[65]

为了构造一个 LPM 或者距离函数，也会出现与 Reeds-Shepp 车、Dubins 车相同的问题。不能通过测试所有 40 个字来发现最短路径，而是要定义简单测试，在这些测试中每次只有

一个特定字起作用[65]。胞腔剖分的一个片段和产生的最优尚需代价（可以被称为Balkcom-Mason 度量）如图 15.14 所示。

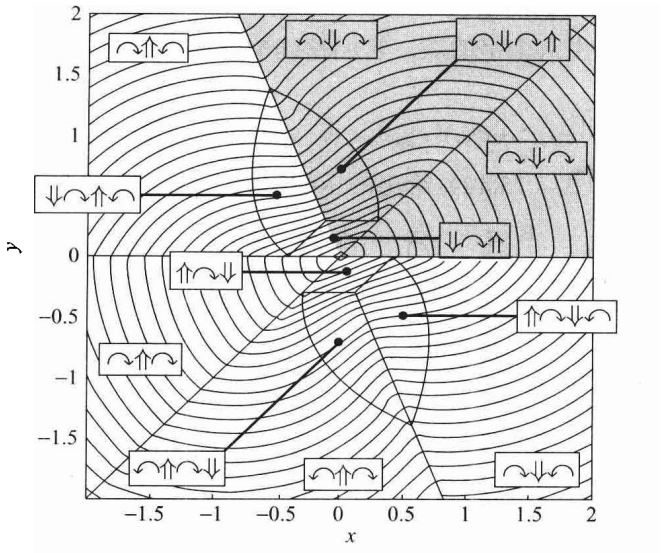


图 15.14 对于 $q_I=\left(x,y,\frac{\pi}{4}\right)$ 和 $q_G=(0,0,0)$ 这种情况的最优曲线的一个片段[65]；显示了最优尚需代价 G^* 的水平集；坐标对应于式（13.16）中 $r=L=1$ 的一个差动机构

15.4 非完整性系统理论

本节准确描述曾在第 13 章和第 14 章使用过的非完整性问题，并且还将讨论第 15.1.3 节定义的短时可控性（STLC）。因为非完整性涉及来自于微分几何、代数、控制论、力学等许多学科中很深的数学原理，所以这里所提到的内容对于非完整性这门学科来说还只是很肤浅的一小部分。目的是促进读者对这些学科进行进一步的学习和研究。

15.4.1 仿射控制系统

非完整性系统理论被限制在一类非常特殊的非线性系统。第 15.4 节的内容利用了线性代数的思路。主要概念将用平滑流形 X 上的向量场的线性组合来表述。因此只讨论第 13.2.3 节简要介绍过的仿射控制系统。对于这些系统，$\dot{x}=f(x,u)$ 的形式是：

$$\dot{x}=h_0(x)+\sum_{i=1}^{m}h_i(x)u_i \tag{15.52}$$

其中每个 h_i 都是 X 上的一个向量场。

向量场用 X 的一个坐标邻域来表达。通常 $m<n$，其中 n 是 X 的维数。除非特别指出，否则都假设 $U=\mathbb{R}^m$。在某些情况下，对 U 可能有所限制。

每个行动变量 $u_i\in\mathbb{R}$ 都可以被想象成一个系数，它确定有多少 $h_i(x)$ 能够融合到 $\dot{x}$。漂移项 $h_0(x)$ 总是存在的，但是往往只关注无漂移的情况。这意味着对于所有 $x\in X$，$h_0(x)=0$，

所以：

$$\dot{x}=\sum_{i=1}^{m}h_i(x)u_i \tag{15.53}$$

本节的大多数地方都将使用无漂移的情况。集合 $h_1,\cdots,h_m$ 称为系统向量场。从本质上讲，U 至少包含一个含 $\mathbb{R}^m$ 的原点的开集。如果原点不包含在 U 中，那么系统不再是无漂移的[1]。

仿射控制系统出现在许多机械系统中。C-空间上的速度约束通常都是 Pfaffian 形式（13.5）。在第 13.1.1 节中，在这种约束下，可以产生一个位形转移方程（13.6），如果 q 是固定的，那么方程是线性的。这是式（15.53）采用 $X=C$ 时的无漂移形式。第 13.1.2 节中的大多数方法都可以用这种形式来表达。对位形的这种 Pfaffian 约束通常被称为运动学约束，因为它们是由于相互接触的物体的运动而产生的，如一个车轮的滚动。式（15.52）在相空间 X 中表达的更一般的情况来自于动力学约束，这些约束从力学表述的欧拉-拉格朗日方程（13.118）和哈密顿方程（13.198）中产生。它们反映守恒定理，并且漂移项的出现通常是由于动量的缘故。

例 15.5（差动机构和汽车的一个简化模型） 将第 13.1.2 节的简易汽车和差动机构模型进行简化后可以用式(15.53)的形式来表达。通过对 U 进行不同的限制，简化模型式(15.48)经过改写后就可以很方便地表达它们。式（15.48）的第三个方程可以被简化为 $\dot{\theta}=u_2$ 而不影响速度的集合。为了与式（15.53）保持一致，方程可以表示为线性代数的形式：

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix}=\begin{pmatrix}\cos\theta\\ \sin\theta\\ 0\end{pmatrix}u_1+\begin{pmatrix}0\\ 0\\ 1\end{pmatrix}u_2 \tag{15.54}$$

显然，这里有两个系统向量场，选择标量 u_1 和 u_2 可以将它们组合在一起。一个向量场进行平移，而另外一个进行旋转。如果对 U 没有限制，该系统的行为就像一个差动机构，因为简易汽车不能进行单纯的旋转。用式（15.54）对简易汽车进行模拟要求对 U 有所限制（如第 15.3.2 节所要求的 u_1 为 1 或者 −1）。这等价于图 13.5 和式（13.18）的独轮车。

通过将向量场写入矩阵，式（15.54）可以被等价地表示为：

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix}=\begin{pmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}u_1\\ u_2\end{pmatrix} \tag{15.55}$$

■

在式（15.54）中，将向量场写成用行动变量线性组合起来的列向量。那么仿射控制系统也可以用式（15.55）中的矩阵乘法来表达。一般情况下，向量场可以记为一个 $n\times m$ 的矩阵：

$$H(x)=(h_1(x)\ h_2(x)\ \cdots\ h_m(x)) \tag{15.56}$$

在无漂移情况下，这产生的：

$$\dot{x}=H(x)u \tag{15.57}$$

是一个等价的表达式（15.53）的方法。

1　实际上，如果 U 的凸壳含有包含原点的一个开集，那么无漂移系统可以用快速切换来进行模拟。

有时可以用 Pfaffian 约束：

$$g_1(x)\dot{x}_1 + g_2(x)\dot{x}_2 + \cdots + g_n(x)\dot{x}_n = 0 \tag{15.58}$$

来代替状态转移方程。就像第 13.1.1 节所指出的，k 个独立的 Pfaffian 约束的集合可以转换为一个有 $m=(n-k)$ 个行动变量的状态转移方程。所产生的状态转移方程是无漂移的仿射控制系统。因此 Pfaffian 约束提供了确定无漂移的仿射控制系统的一个对偶方法。k 个 Pfaffian 约束可以用矩阵的形式来表达：

$$G(x)\dot{x} = 0 \tag{15.59}$$

它是式（15.57）的对偶形式，并且 $G(x)$ 是每个 Pfaffian 约束的系数 g_i 所形成的一个 $k \times n$ 矩阵。有漂移的系统可以用类似 Pfaffian 的形式来表达：

$$g_0(x) + g_1(x)\dot{x}_1 + g_2(x)\dot{x}_2 + \cdots + g_n(x)\dot{x}_n = 0 \tag{15.60}$$

15.4.2 确定一个系统是否是非完整的

本节将简要介绍类似于平常线性代数但却是用于向量场的线性组合的一些概念，提出描述系统性质的概念和工具。那么这就能够评估一个系统是否是非完整的和是否是 STLC 的。许多构造方法都用 19 世纪挪威数学家 Sophus Lie 的名字命名，他对恰好与非完整性系统的研究有关的代数和几何有重大贡献。

15.4.2.1 完全可积的或非完整的

每个仿射控制系统必须是如下情形之一（而不能两者都是）。

（1）完全可积的：这意味着 Pfaffian 式（15.59）可以通过 k 个 $f_i(x)=0$ 形式的方程对时间求导来得到。这种情况被解释为限制在第 13.1.3 节中的一个曲面上。式（13.22）给出了一个限制在 $\mathbb{R}^2$ 中一个圆上的例子。

（2）非完整的：这意味着系统不是完全可积的。在这种情况下，它甚至可能到达整个 X，即使行动变量的数量 m 远小于 X 的维数 n。

在这里，“完整的”和“完全可积的”是同义的，而“不可积的”与“非完整的”是同义的。“非完整的”这个术语有时应用于非 Pfaffian 约束[591]，但是为了与力学文献[114]相一致，这里避免了这种做法。

这里所用的可积性的概念与式（14.1）的要求不同。在那里，需要对状态转移方程进行积分来得到从任意初始状态开始的积分曲线。本书所考虑的系统都要求如此。比较而言，这里的完全可积性则意味着不用导数也可以表达系统。也就是说，所有积分曲线最终都可以用不涉及导数的约束来刻画。

为了有助于完全可积性的理解，积分曲线的概念将从一维推广到 m 维。一个流形 $M \subseteq X$ 被称为是一个 Pfaffian 约束集合的一个积分流形，如果在每个 $x \in M$，切空间 $T_x(M)$ 中的所有向量满足约束。对于完全可积 Pfaffian 约束的一个集合，通过定义每个 $x \in X$ 的最大积分流形就可以得到 X 的一个积分流形分拆。所产生的分拆称为一个叶层结构，其中最大积分流形称为叶子[874]。

例 15.6（圆叶的叶层结构） 作为一个例子，假设 $X=\mathbb{R}^n$ 并考虑 Pfaffian 约束：

$$x_1\dot{x}_1 + x_2\dot{x}_2 + \cdots + x_n\dot{x}_n = 0 \tag{15.61}$$

这是完全可积的，因为它可以通过如下一个球的方程对时间求导来得到：

$$x_1^2 + x_2^2 + \cdots + x_n^2 - r^2 = 0 \tag{15.62}$$

其中 r 是一个常数。通过积分得到的特定球体依赖于一个初始状态。叶层结构是所有中心在原点的同心球的集合。例如，如果 $X = \mathbb{R}^3$，那么每个点 $(0,0,r)$ 都有一个最大积分流形。在这种情况下，它是一个半径为 r 的球。选择每个 $r \in [0,\infty)$ 就会生成叶层结构。 ■

本节的任务是确定一个系统是否是完全可积的。假设某人在与你一起打游戏。给你一个仿射控制系统，要求你确定它是否是完全可积的。与你打游戏的人可以从方程 $h_i(x) = 0$ 开始并求导得到 Pfaffian 约束。然后这些都可以转换到参数形式来得到状态转移方程（15.53）。很容易构造问题；但是它们往往很难解决。本节的概念可以用来确定是否可能在游戏中取胜。主要的工具是 Frobenius 定理，它判断一个系统是否是完全可积的。非常遗憾，得到的结论没有产生积分约束 $h_i(x) = 0$。因此重要的是需要记住“可积性”并不意味着能够对它积分得到解析形式。这是一个很有挑战性的逆工程问题。另一方面，也很容易朝另外一个方向进行，对约束进行微分来产生一个很有挑战性的游戏。

15.4.2.2　分布

分布[1]表达了一个平滑流形上的向量场的一个集合。假设给定一个无漂移仿射控制系统（见式（15.53））。回顾第 8.3.1 节或者线性代数中的向量空间的定义。同时也可回顾如果行动是固定的，那么状态转移方程可以阐述为一个向量场；如果状态是固定的，也可以阐述为一个向量空间。对于 $U = \mathbb{R}^m$ 和一个固定的 $x \in X$，状态转移方程定义了一个向量空间，该空间中在 x 处计算的每个 h_i 都是一个基向量，每个 u_i 是一个系数。例如在式（15.54）中，在 $q = (0,0,0)$ 处计算的向量场 h_1 和 h_2 分别变成 $[1\ 0\ 0]^T$ 和 $[0\ 0\ 1]^T$。它们作为基向量。通过选择 $u \in \mathbb{R}^2$ 的值，产生一个二维向量空间。通过设置 $u_1 = a$ 和 $u_2 = b$ 来表示 $[a\ 0\ b]^T$ 形式的任何向量。更加一般情况下，令 $\Delta(x)$ 表示用这种方法得到的任意 $x \in X$ 的向量空间。

向量空间的维数是基向量的个数。因此 $\Delta(x)$ 的维数是式（15.56）中 $H(x)$ 在特定的 $x \in X$ 处的秩。现在考虑为每个 $x \in X$ 定义 $\Delta(x)$。这产生了向量空间的一个参数表示的族，其中一个空间对应于一个 $x \in X$。结果也可以阐述为向量场的一个参数表示的族。例如 i 从 1 到 m，对于所有 $i \neq j$，考虑行动 $u_i = 1$ 和 $u_j = 0$。如果行动在所有 $x \in X$ 上保持固定常数，那么它从 m 个向量场的族中选择一个向量场 h_i：

$$\dot{x} = h_i(x) \tag{15.63}$$

使用常数行动可以定义一个 m 维向量空间，其中每个向量场 h_i 是一个基向量（假设 h_i 是线性无关的），且 $u_i \in \mathbb{R}$ 是系数：

$$u_1 h_1(x) + u_2 h_2(x) + \cdots + u_m h_m(x) \tag{15.64}$$

这个思路可以被推广到允许 u_i 在 X 上变化。因此不将 u 固定为常数，而是将它当作一个反馈规划 $\pi : X \to U$，其中 x 处的行动由 $u = \pi(x)$ 给定。所有如下向量场：

$$\pi_1(x)h_1(x) + \pi_2(x)h_2(x) + \cdots + \pi_m(x)h_m(x) \tag{15.65}$$

的集合被称为向量场集合 $\{h_1, \cdots, h_m\}$ 的分布，用 Δ 表示。如果 Δ 从一个仿射控制系统中得到，那么 Δ 称为系统分布。所产生的向量场的集合并不完全是一个向量空间，因为非零系

1　这个分布与概率论中分布没有关系。它只是术语的一个巧合。

数 π_i 不一定有倒数。而对于一个向量场的系数来说这是必需的，这个条件在常数行动情况下使用 $\mathbb{R}$ 可以得到满足。一个分布在代数意义上被认为是一个模[472]。在大多数环境下，将它想象成一个向量空间是有益的（只是不设法颠倒系数）。因为一个分布几乎是一个向量空间，所以经常用线性代数中张成的空间来定义它：

$$\Delta = \text{span}\{h_1, h_2, \cdots, h_m\} \tag{15.66}$$

并且，它实际上就是一个关于常数行动 $u \in \mathbb{R}^m$ 的向量空间。注意到对于每个固定的 $x \in X$，可以得到向量空间 $\Delta(x)$，像前面所定义的那样。向量场 f 被认为属于一个分布 Δ，如果它可以用式（15.65）来表达。如果对于所有 $x \in X$，$\Delta(x)$ 的维数为 m，那么称 Δ 为一个非奇异分布（或者正则分布）。否则称之为奇异分布，使得 $\Delta(x)$ 的维数小于 m 的点 $x \in X$ 称为奇异点。如果 $\Delta(x)$ 的维数是所有 $x \in X$ 上的一个常数 c，那么 c 被称为分布的维数，用 $\dim(\Delta)$ 表示。如果向量场是平滑的，并且 π 也限定为是平滑的，那么可以得到一个平滑分布，它是原始分布的一个子集。

如图 15.15 所示，可以用一个平滑流形的切丛对分布进行说明。式（8.9）定义 $X = \mathbb{R}^n$ 的切丛，在这里将其推广到任意平滑流形 X，得到：

$$T(X) = \bigcup_{x \in X} T_x(X) \tag{15.67}$$

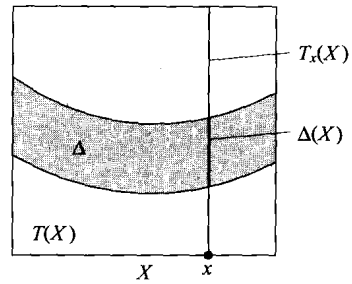


图 15.15 分布 Δ 可以被想象为一个切丛 $T(X)$ 的一个片段；它限制了 $x \in X$ 上的切空间

切丛是一个 $2n$ 维流形，其中 n 是 X 的维数。$x = (q, \dot{q})$ 的一个相空间实际上是 $T(C)$。在目前情形下，$T(X)$ 的每个元素产生了一个状态和一个速度 $(x, \dot{x})$。对一个无漂移仿射控制系统来说可能用到的是哪种组合对呢？每个 $\Delta(x)$ 指出对于一个固定的 x 来说的可能 $\dot{x}$ 值的集合。点 x 有时被称为基，$\Delta(x)$ 被称为 $T(X)$ 中 x 上的纤维（fiber）。分布 Δ 为每个 $x \in X$ 指定了 $T_x(X)$ 的一个子集。对于属于 Δ 的一个向量场 f，对所有 $x \in X$，它都必须满足 $f(x) \in \Delta(x)$。这只是对 $T(X)$ 的一个子集的一个限制。如果 $m = n$ 且系统向量场是无关的，那么任意向量场都是允许的。在这种情况下，Δ 包括能够由 $T(X)$ 中的向量来构造的任意向量场。

例 15.7（差动机构的分布） 系统（15.54）产生了一个二维分布：

$$\Delta = \text{span}\left\{\begin{bmatrix}\cos\theta & \sin\theta & 0\end{bmatrix}^T, \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T\right\} \tag{15.68}$$

分布是非奇异的，因为对于坐标邻域中的任意 (x, y, θ)，所产生的向量空间 $\Delta(x, y, \theta)$ 是二维的。 ■

例 15.8（一个奇异分布） 考虑如下系统，它是在文献[481]中给出的：

$$\begin{aligned}\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{pmatrix} &= h_1(x)u_1 + h_2(x)u_2 + h_3(x)u_3\\ &= \begin{pmatrix}x_1\\ 1+x_3\\ 1\end{pmatrix}u_1 + \begin{pmatrix}x_1x_2\\ (1+x_3)x_2\\ x_2\end{pmatrix}u_2 + \begin{pmatrix}x_1\\ x_1\\ 0\end{pmatrix}u_3\end{aligned} \tag{15.69}$$

分布是：

$$\Delta = \text{span}\{h_1, h_2, h_3\} \tag{15.70}$$

第一问题是对于任意 $x \in \mathbb{R}^3$，$h_2(x) = h_1(x)x_2$，这意味着向量场在所有 $\mathbb{R}^3$ 上是线性相关的。因此这个分布是奇异的，因为 $m=3$，而如果 $x_1 \neq 0$ 时 $\Delta(x)$ 的维数为 2。如果 $x_1 = 0$，那么 $\Delta(x)$ 的维数降为 1。Δ 的维数没有定义，因为 $\Delta(x)$ 的维数依赖于 x。 ■

分布也可以直接用 Pfaffian 约束来定义。每个 $g_i(x) = 0$ 称为一个**零化子**（annihilator），因为施加约束后已经从考虑的向量场中排除了许多向量场。在每个 $x \in X$，$\Delta(x)$ 定义为所有满足全部 k 个 Pfaffian 约束的速度向量的集合。约束本身就可以用来形成一个**补分布**（codistribution），它是一种对偶分布。余分布可以认为是一个向量空间，其中每个约束是一个基向量。约束可以相加或者与任意 $c \in \mathbb{R}$ 相乘，这对所产生的可容许向量场的分布没有影响。

15.4.2.3　李括号

要确定一个系统是否是非完整的，关键是构造将两个行动变量的效果组合在一起的运动，这可能在对于系统分布来说似乎是不可能的方向上产生。为了激发下面的思路，考虑式（15.54）的差动机构模型。在区间 $[0, 4\Delta t]$ 上应用如下分段常数行动轨迹：

$$u(t) = \begin{cases} (1,0) & t \in [0, \Delta t) \\ (0,1) & t \in [\Delta t, 2\Delta t) \\ (-1,0) & t \in [2\Delta t, 3\Delta t) \\ (0,-1) & t \in [3\Delta t, 4\Delta t] \end{cases} \tag{15.71}$$

行动轨迹是四个运动本原的序列：（1）向前平移；（2）向前旋转；（3）向后平移；（4）向后旋转。

从 $q_I = (0,0,0)$，这四个运动本原的结果如图 15.16 所示。在家里可以用一个轮轴和两个轮子做这个有趣的试验。结果是差动机构向侧面移动[1]。根据转移方程（15.54），这种运动是不可能的。这是非线性系统的一个良好性质。状态可在似乎不可能的方向上运动。一个更加熟悉的例子是平行泊车。一辆车不能直接侧向移动，但是通过前后运动实现的侧向运动就能够进入狭窄的停车位，此时所采取的行动类似于式（15.71）中的本原。

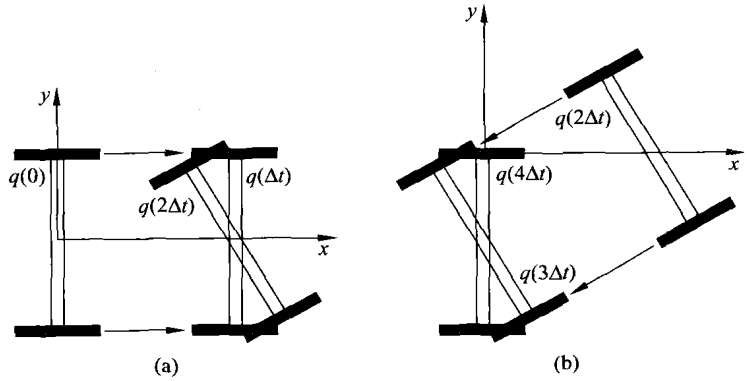


图 15.16　（a）前两个本原的效果；（b）后两个本原的效果

1　它也稍微向前移动，但是通过加长第三个本原的时间或者考虑当 Δ 接近于零时的极限，就可以消除它。

式（15.71）的运动以检查交换性的形式出现。回顾第 4.2.1 节中一个群 G 被称为是可交换的（或者 Abelian 的），如果对于任意 $a,b\in G$ 都有 $ab=ba$。一个**换位子**（commutator）是一个 $aba^{-1}b^{-1}$ 形式的群元素。如果群是可交换的，那么对于任意 $a,b\in G$，有 $aba^{-1}b^{-1}=e$（单位元）。如果换位子产生 G 的一个非单位元，那么群不是可交换的。同理，如果式（15.71）所产生的轨迹没有形成一个环（返回到起始点），那么运动本原不能进行交换。因此式（15.71）中运动本原序列被称为换位子运动。如果换位子运动不产生不被系统分布所允许的任意速度，那么系统是完全可积的。这意味着如果限制在一个曲面上，那么就不可能通过换位子运动来离开该曲面。

现在将差动机构推广到有两个行动变量的任意无漂移仿射控制系统：

$$\dot{x}=f(x)u_1+g(x)u_2 \tag{15.72}$$

用式（15.53）的符号，向量场是 h_1 和 h_2。但这里选择的 f 和 g 允许用下标来表示即将说明的向量场的分量。

假设将换位子运动式（15.71）应用到式（15.72），如图 15.17 所示。那么确定所产生的运动需要某些普通计算，这与为差动机构而构造的简单几何自变量不同。对 Δt 趋近于零时的极限速度进行表达将是很方便的。这可以通过使用泰勒级数来得到。由于行动历史是分段常数，那么这些都可以得到简化。

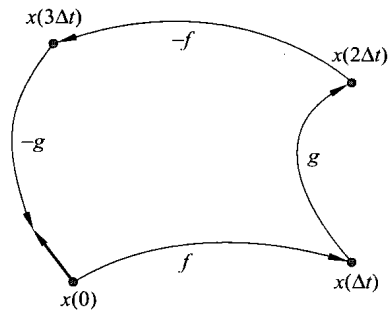


图 15.17 李括号所得到的速度可以用四个运动本原的序列来近似

在应用一个常数行动的情况下，下面的导数将要求 $\ddot{x}$ 的表达。对于每个行动，得到一个 $\dot{x}=f(x,u)$ 形式的向量场。通过微分，产生：

$$\ddot{x}=\frac{\mathrm{d}h}{\mathrm{d}t}=\frac{\partial h}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t}=\frac{\partial h}{\partial x}\dot{x}=\frac{\partial h}{\partial x}h(x) \tag{15.73}$$

这是因为 h 是 x 的函数，而 x 是 t 的函数。导数 $\partial h/\partial x$ 是一个 $n\times n$ 的 Jacobian 矩阵，它乘以向量 $\dot{x}$。为了进一步澄清式（15.73），每个分量可以被表示为：

$$\ddot{x}_i=\frac{\mathrm{d}}{\mathrm{d}t}h_i(x(t))=\sum_{j=1}^{n}\frac{\partial h_i}{\partial x_j}h_j \tag{15.74}$$

现在式（15.71）所得到的状态轨迹用式（14.17）给出的泰勒级数来确定。从第一个运动本原 $u=(1,0)$ 产生的状态轨迹可以表示为：

$$\begin{aligned}x(\Delta t)&=x(0)+\Delta t\cdot\dot{x}(0)+\frac{1}{2}(\Delta t)^2\ddot{x}(0)+\cdots\\&=x(0)+\Delta t\cdot f(x(0))+\frac{1}{2}(\Delta t)^2\left.\frac{\partial f}{\partial x}\right|_{x(0)}f(x(0))+\cdots\end{aligned} \tag{15.75}$$

其中第二行利用了式（15.73）。对于第二个本原的状态轨迹的泰勒级数展开为：

$$x(2\Delta t)=x(\Delta t)+\Delta t\cdot g(x(\Delta t))+\frac{1}{2}(\Delta t)^2\left.\frac{\partial g}{\partial x}\right|_{x(\Delta t)}f(x(\Delta t))+\cdots \tag{15.76}$$

用式（15.75）中的泰勒级数展开来表达 $x(\Delta t)$ 就可以得到 $g(x(\Delta t))$ 的表达。代入第一个项并简化后得到：

$$x(2\Delta t)=x(0)+\Delta t\cdot g+(\Delta t)^2\left(\frac{1}{2}\frac{\partial f}{\partial x}f+\frac{\partial g}{\partial x}f+\frac{1}{2}\frac{\partial g}{\partial x}g\right)+\cdots \tag{15.77}$$

为了简化表达式，$x(0)$ 的计算已经从每个 f 和 g 及它们的导数中提取出来。

当需要时，可以为其余两个运动本原而重复代入前面的泰勒级数展开。对于第三个本原，泰勒级数展开为：

$$x(3\Delta t)=x(0)+\Delta t\cdot g+(\Delta t)^2\left(\frac{\partial g}{\partial x}f-\frac{\partial f}{\partial x}g+\frac{1}{2}\frac{\partial g}{\partial x}g\right)+\cdots \tag{15.78}$$

最后，应用所有四个本原后的泰勒级数展开为：

$$x(4\Delta t)=x(0)+(\Delta t)^2\left(\frac{\partial g}{\partial x}f-\frac{\partial f}{\partial x}g\right)+\cdots \tag{15.79}$$

取极限产生：

$$\lim_{\Delta t\to 0}\frac{x(4\Delta t)-x(0)}{(\Delta t)^2}=\frac{\partial g}{\partial x}f-\frac{\partial f}{\partial x}g \tag{15.80}$$

它被称为 f 和 g 的李括号，用 $[f,g]$ 表示。类似于式（15.74），第 i 个分量可以被表示为：

$$[f,g]_i=\sum_{j=1}^{n}\left(f_j\frac{\partial g_i}{\partial x_j}-g_j\frac{\partial f_i}{\partial x_j}\right) \tag{15.81}$$

李括号是一个重要的运算，它与物理和数学中出现的 Poisson 和 Jacobi 括号有关。

例 15.9（差动机构的李括号） 李括号指出差动机构的侧向运动是可能的。考虑式（15.54）中用到的两个向量场的李括号。令 $f=[\cos\theta\ \sin\theta\ 0]^T$ 和 $g=[0\ 0\ 1]^T$。将 f 和 g 重新命名为 h_1 和 h_2 使得可以用下标来表示向量场的分量。

通过应用式（15.81），李括号 $[f,g]$ 为：

$$\begin{aligned}
[f,g]_1&=f_1\frac{\partial g_1}{\partial x}-g_1\frac{\partial f_1}{\partial x}+f_2\frac{\partial g_1}{\partial y}-g_2\frac{\partial f_1}{\partial y}+f_3\frac{\partial g_1}{\partial\theta}-g_3\frac{\partial f_1}{\partial\theta}=\sin\theta\\
[f,g]_2&=f_1\frac{\partial g_2}{\partial x}-g_1\frac{\partial f_2}{\partial x}+f_2\frac{\partial g_2}{\partial y}-g_2\frac{\partial f_2}{\partial y}+f_3\frac{\partial g_2}{\partial\theta}-g_3\frac{\partial f_2}{\partial\theta}=-\cos\theta\\
[f,g]_3&=f_1\frac{\partial g_3}{\partial x}-g_1\frac{\partial f_3}{\partial x}+f_2\frac{\partial g_3}{\partial y}-g_2\frac{\partial f_3}{\partial y}+f_3\frac{\partial g_3}{\partial\theta}-g_3\frac{\partial f_3}{\partial\theta}=0
\end{aligned} \tag{15.82}$$

所产生的向量场是 $[f,g]=[\sin\theta\ -\cos\theta\ 0]^T$，它表示想要的侧向运动。在 $q=(0,0,0)$，得到向量 $[0\ -1\ 0]^T$。这意味着进行短的换位子运动能够使得差动机构在 $-y$ 方向上侧向摆动，如图 15.16 所示。 ■

例 15.10（线性向量场的李括号） 假设每个向量场是 x 的一个线性函数。$n\times n$ Jacobian 矩阵 $\partial f/\partial x$ 和 $\partial g/\partial x$ 是常数。

作为一个例子，回顾非完整积分器（13.43）。在线性代数形式下，系统为：

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{pmatrix}=\begin{pmatrix}1\\ 0\\ -x_2\end{pmatrix}u_1+\begin{pmatrix}0\\ 1\\ x_1\end{pmatrix}u_2 \tag{15.83}$$

令 $f=h_1$ 和 $g=h_2$。Jacobian 矩阵为：

$$\frac{\partial f}{\partial x}=\begin{pmatrix}0&0&0\\0&0&0\\0&-1&0\end{pmatrix}\quad 和\quad \frac{\partial g}{\partial x}=\begin{pmatrix}0&0&0\\0&0&0\\1&0&0\end{pmatrix} \tag{15.84}$$

利用式（15.80）：

$$\frac{\partial g}{\partial x}f-\frac{\partial f}{\partial x}g=\begin{pmatrix}0&0&0\\0&0&0\\1&0&0\end{pmatrix}\begin{pmatrix}1\\0\\-x_2\end{pmatrix}-\begin{pmatrix}0&0&0\\0&0&0\\0&-1&0\end{pmatrix}\begin{pmatrix}0\\1\\x_1\end{pmatrix}=\begin{pmatrix}0\\0\\2\end{pmatrix} \tag{15.85}$$

该结果可以用式（15.81）来验证。 ■

15.4.2.4 Frobenius 定理

当需要确定一个系统是否是完全可积的（完整的）或者非完整的（不可积的），李括号是仅有的工具。假设给定一个系统（15.53）。用 m 个系统向量场 $h_1,\cdots,h_m$，那么可以形成 $\binom{m}{2}$ 个 $[h_i,h_j]$ 形式的李括号，$i<j$。分布 Δ 称为是对合的[135]，如果对于这些括号，存在 m 个系数 $c_k\in\mathbb{R}$ 使得：

$$\left[h_i,h_j\right]=\sum_{k=1}^{m}c_k h_k \tag{15.86}$$

也就是说，每个李括号都可以表示为系统向量场的一个线性组合，因此它属于 Δ。李括号不能够逃离 Δ 而产生新的运动方向。我们不需要考虑系统向量场的所有 n^2 个可能的李括号，因为 $[h_i,h_j]=-[h_j,h_i]$ 和 $[h_i,h_i]=0$。因此如果仅看 $\binom{m}{2}$ 个组合对，对合的定义并不发生改变。

如果系统是平滑的且分布是非奇异的，那么 Frobenius 定理立刻就能够刻画可积性的性质：一个系统是完全可积的，当且仅当它是对合的。

关于 Frobenius 定理的证明可以参考微分几何和控制论方面的书籍[135,158,481,848]。也有些情况不要求分布是非奇异的。

确定可积性涉及进行李括号和确定是否满足式（15.86）。为线性无关而进行的线性代数检验可以避免对系数的搜索。通过加入一个新的列 $[h_i,h_j]$，式（15.56）中定义的 $n\times m$ 矩阵 $H(x)$ 可以增广到一个 $n\times(m+1)$ 矩阵 $H'(x)$ 中。如果对于任意 h_i 和 h_j 组合对来说，$H'(x)$ 的秩都是 $m+1$，那么立刻知道系统是非完整的。如果对于所有 m 个李括号，$H'(x)$ 的秩为 m，那么系统是完全可积的。对于一个固定矩阵 B，用 $\dot{x}=Bu$ 所表示的无漂移线性系统是完全可积的，因为所有的李括号为零。

例 15.11（差动机构是非完整的） 对于式（15.54）中的差动机构模型，例 15.9 中的李括号 $[f,g]$ 为 $[\sin\theta\quad -\cos\theta\quad 0]^T$。矩阵 $H'(q)$ 为：

$$H'(q)=\begin{pmatrix}\cos\theta&0&\sin\theta\\\sin\theta&0&-\cos\theta\\0&1&0\end{pmatrix} \tag{15.87}$$

其中 $q=(x,y,\theta)$。对于所有 $q\in C$，$H'(q)$ 的秩为 3（$H'(q)$ 的行列式为 1）。因此根据 Frobenius 定理可知系统是非完整的。 ■

例 15.12（非完整积分器是非完整的） 我们希望非完整积分器是非完整的。在例 15.10 中，李括号为 $[0 \quad 0 \quad 2]^T$。矩阵 $H'(q)$ 为：

$$H'(q)=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ -x_2 & x_1 & 2\end{pmatrix} \tag{15.88}$$

显然对于所有 $q\in C$，它是满秩的。 ■

例 15.13（限制在一个球上） 假设给定如下系统：

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{pmatrix}=\begin{pmatrix}x_2\\ -x_1\\ 0\end{pmatrix}u_1+\begin{pmatrix}x_3\\ 0\\ -x_1\end{pmatrix}u_2 \tag{15.89}$$

对它来说，$X=\mathbb{R}^3$ 和 $U=\mathbb{R}^2$。因为向量场是线性的，Jacobian 矩阵是常数（如例 15.10 所示）：

$$\frac{\partial f}{\partial x}=\begin{pmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 0\end{pmatrix} \quad 和 \quad \frac{\partial g}{\partial x}=\begin{pmatrix}0 & 0 & 1\\ 0 & 0 & 0\\ -1 & 0 & 0\end{pmatrix} \tag{15.90}$$

采用式（15.80），那么：

$$\frac{\partial g}{\partial x}f-\frac{\partial f}{\partial x}g=\begin{pmatrix}0 & 0 & 1\\ 0 & 0 & 0\\ -1 & 0 & 0\end{pmatrix}\begin{pmatrix}x_2\\ -x_1\\ 0\end{pmatrix}-\begin{pmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}\begin{pmatrix}x_3\\ 0\\ -x_1\end{pmatrix}=\begin{pmatrix}0\\ x_3\\ -x_2\end{pmatrix} \tag{15.91}$$

这产生矩阵：

$$H'(q)=\begin{pmatrix}x_2 & -x_1 & 0\\ x_3 & 0 & -x_1\\ 0 & x_3 & -x_2\end{pmatrix} \tag{15.92}$$

对于所有 $x\in\mathbb{R}^3$，行列式为零，这意味着 $[f,g]$ 从来不与 f 和 g 线性无关。因此系统是完全可积的[1]。

可以通过对一个球的方程进行求微分来构造系统。令：

$$f(x)=x_1^2+x_2^2+x_3^2-r^2=0 \tag{15.93}$$

求关于时间的微分得到：

$$x_1\dot{x}_1+x_2\dot{x}_2+x_3\dot{x}_3=0 \tag{15.94}$$

这是一个 Pfaffian 约束。式（15.89）给出满足式（15.94）的向量集合的一个参数表示。对于每个 $(u_1,u_2)\in\mathbb{R}^2$，式（15.89）产生了一个满足式（15.94）的向量。因此这是限制在一个球上的例子，我们期望它是完全可积的。但是仅用式（15.89）很难得到相同的结果。 ■

15.4.3　确定可控性

确定完全可积性是迈向确定一个无漂移仿射控制系统是否是 STLC 的第一步。李括号试

1　这个系统在原点处是奇异的。从技术上讲，需要 Frobenius 定理的一个变化形式。

图在似乎不被系统分布所允许的方向上产生运动。在每个 q 上，李括号可能产生不在 $\Delta(q)$ 中的速度。进一步用李括号，有可能完全描述从每个 q 所有可能的方向。目前李括号仅应用于系统向量场 $h_1,\cdots,h_m$。李括号运算还可能进一步应用于李括号。例如可以计算 $[h_1,[h_1,h_2]]$。这可能产生一个向量场，它与第 15.4.2 节为 Frobenius 定理而考虑的所有向量场都线性无关。本节的主要思路是递归地应用李括号直到不再发现线性无关的向量场。结果被称为李代数。如果用这种方法得到的无关向量场的数量就是 X 的维数，那么系统是 STLC 的。

15.4.3.1 李代数

首先建立一般意义上李代数的概念。令 V 为系数是 $\mathbb{R}$ 的任意向量空间。在 V 中，向量能够相加或者与 $\mathbb{R}$ 中的元素相乘，但是两个向量无法“相乘”来得到第三个向量。李代数给 V 引入了一种乘法运算。乘积称为括号或者李括号（这里所考虑的李括号是前面李括号的推广），表示为 $[\cdot,\cdot]=V\times V\to V$。

为了得到 V 中的李代数，括号必须满足如下公理：

1. 双线性——对于任意 $a,b\in\mathbb{R}$ 和 $u,v,w\in V$：

$$\begin{aligned}[au+bv,w]&=a[u,w]+b[v,w]\\ [u,av+bw]&=a[u,w]+b[u,w]\end{aligned} \tag{15.95}$$

2. 反对称——对于任意 $u,v\in V$，有：

$$[u,v]=-[v,u] \tag{15.96}$$

即括号是反交换的。

3. Jacobi 恒等式——对于任意 $u,v,w\in V$：

$$[[u,v],w]+[[v,w],u]+[[w,u],v]=0 \tag{15.97}$$

注意到括号甚至不服从结合律。

令 $\mathcal{L}(V)$ 表示 V 的李代数。这是一个向量空间，它包括 V 的所有元素以及通过李括号运算所得到的新元素。李代数 $\mathcal{L}(V)$ 包括通过任意有限次嵌套李括号运算所得到的每个向量。因此描述一个李代数要求描述在括号运算的代数闭包下所得到的所有向量。因为 $\mathcal{L}(V)$ 是一个向量空间，这可以用线性无关的基向量来实现，$\mathcal{L}(V)$ 中的所有元素都可以表示为基向量的线性组合。

例 15.14（向量叉乘） 令 V 是 $\mathbb{R}^3$ 上的向量空间。基元素通常表示为 $\hat{i}$、$\hat{j}$ 和 $\hat{k}$。该空间的一个括号只是叉乘：

$$[u,v]=u\times v \tag{15.98}$$

可以验证它满足李括号所要求的公理。

通常在解析几何中利用了叉乘的一个特性是它在 u 和 v 张成的空间以外产生一个向量。例如令 W 为向量的二维子空间：

$$W=\text{span}\{\hat{i},\hat{j}\} \tag{15.99}$$

叉乘产生的向量是 $\hat{k}$ 的倍数，如果乘积非零，那么该向量在 V 的外面。这种行为类似于用第 15.4.2 节中的李括号来构造 Δ 外的向量场。■

例 15.15（李群上的李代数） 李群是李代数概念的最重要的应用。回顾第 4.2.1 节矩阵群的概念。贯串本书的重要例子是 $SO(n)$ 和 $SE(n)$。如果阐述为一个平滑流形，那么这些矩阵群是李群的例子[64]。一般情况下，一个李群 G 既是一个可微流形，也是一个关于某

运算“∘”的一个群，当且仅当:

（1）乘积 $a \circ b$，被阐述为 $G \times G \to G$ 中一个函数，它是平滑的。

（2）逆 a^{-1}，被阐述为一个从 G 到 G 的函数，它是平滑的。

需要这两个条件来防止群破坏平滑流形所具有的良好特性。在李群的研究中一个重要的结果是所有紧的有限维李群可以表示为矩阵群。

对于任意李群，一个李代数可以定义在向量场的一个特殊集合上。这些用**左平移**（left translation）映射来定义 $L_g : x \mapsto gx$。L_g 的微分所形成的向量场称为一个左不变向量场。一个李代数可以定义在这些向量场的集合上。李括号的定义依赖于特定的群。对于 $GL(n)$ 的情况，李括号是:

$$[A,B] = AB - BA \tag{15.100}$$

在这种情况下，李括号可用来检验交换律。如果矩阵关于乘法进行交换，那么李括号为零。$SO(n)$ 和 $SE(n)$ 的李括号在许多力学和控制类的书籍中给出[158,848]。在非线性系统和力学的研究中，左不变向量场的李代数是一个重要的结构。 ■

15.4.3.2　系统分布的李代数

现在假设给定向量场的一个集合 $h_1,\cdots,h_m$ 作为一个无漂移仿射控制系统，如式（15.53）所示。与它相关联的分布 Δ 阐述为一个实系数向量空间，李括号运算由式（15.81）给定。可以验证式（15.81）中的李括号运算满足李代数所要求的公理。

如例 15.9 和例 15.10 所示，李括号可能产生 Δ 以外的向量场。将 Δ 的李代数定义为用李括号运算所能得到的所有向量场，这样可以得到一个潜在的大的分布 $\mathcal{L}(\Delta)$。李代数可以用 $h_1,\cdots,h_m$ 和李括号产生的所有无关向量场所张成的空间来表达。但是可能产生的无关向量场不超过 n 个。

例 15.16（差动机构的李代数） 差动机构式（15.54）的李代数是:

$$\mathcal{L}(\Delta) = \operatorname{span}\left\{\begin{bmatrix}\cos\theta & \sin\theta & 0\end{bmatrix}^T, \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T, \begin{bmatrix}\sin\theta & -\cos\theta & 0\end{bmatrix}^T\right\} \tag{15.101}$$

这使用的李括号是在式（15.82）中计算以得到一个三维李代数。不再需要进一步的李括号，因为已经得到最大数量的无关向量场。 ■

例 15.17（涉及嵌套括号的李代数） 前面的例子很简单，.因为只计算一个李括号就产生李代数。假设 $X = \mathbb{R}^5$ 和 $U = \mathbb{R}^2$。在这种情况下，还可得到另外三个线性无关的向量场。李代数的维数可能是 2～5 的任意整数。

令系统为:

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\\ \dot{x}_5\end{pmatrix} = \begin{pmatrix}1\\ 0\\ x_2\\ x_3\\ x_4\end{pmatrix} u_1 + \begin{pmatrix}0\\ 1\\ 0\\ 0\\ 0\end{pmatrix} u_2 \tag{15.102}$$

这是一个链式系统，它在第 15.5.2 节中是一个重要概念。

第一李括号产生:

$$[h_1,h_2] = [0 \quad 0 \quad -1 \quad 0 \quad 0]^T \tag{15.103}$$

其他向量场可以由李括号来得到：

$$[h_1,[h_1,h_2]]=[0\quad 0\quad 0\quad 1\quad 0]^T \tag{15.104}$$

$$[h_1,[h_1,[h_1,h_2]]]=[0\quad 0\quad 0\quad 0\quad 1]^T \tag{15.105}$$

所产生的五个向量场在所有 $x\in\mathbb{R}^5$ 上是无关的。这包括 h_1、h_2 和三个用李括号运算得到的向量场。将它们置于一个 5×5 的矩阵里就可以建立无关性：

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ x_2 & 0 & -1 & 0 & 0 \\ x_3 & 0 & 0 & 1 & 0 \\ x_4 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{15.106}$$

对于所有 $x\in\mathbb{R}^5$，它都是满秩的。再也没有其他向量场是无关的了。因此五维李代数为：

$$\mathcal{L}(\Delta)=\text{span}\{h_1,h_2,[h_1,h_2],[h_1,[h_1,h_2]],[h_1,[h_1,[h_1,h_2]]]\} \tag{15.107}$$

■

15.4.3.3 李代数的 Philip Hall 基

确定李代数的基可能是一个烦琐的过程。例 15.7 中给出李括号的组合，但是并不预先知道哪个将产生无关向量场。可能需要大量的李括号运算，包括诸如 $[[h_1,h_2],h_3]$ 这样的嵌套。嵌套李括号运算的最大深度并没有预先知道。因此必须进行系统搜索（这实际上可以建模成一个离散规划问题）：从 $h_1,\cdots,h_m$ 开始并用李括号迭代生成新的无关向量场。

一个普通方法是生成李代数 $\mathcal{L}(\Delta)$ 的 Philip Hall（P. Hall）基。基的构造本质上按照广度优先搜索，其中搜索深度定义为括号运算的嵌套层数。一个李乘积的阶（或者深度）d 递归定义如下。对于基本情况，对于任意系统向量场，令 $d(h_i)=1$。对于李乘积 $[\phi_1,\phi_2]$，令：

$$d([\phi_1,\phi_2])=d(\phi_1)+d(\phi_2) \tag{15.108}$$

因此阶仅仅是李括号运算的嵌套深度（加 1）。例如，$d([h_1,h_2])=2$ 和 $d([[h_1,h_2],h_3])=3$。

除了进行标准广度优先搜索以外，应该还能够自动进行裁剪以确保总是能够利用反对称和 Jacobi 恒等式来消除冗余。一个 P. Hall 基是李乘积的一个序列 $\mathcal{PH}=(\phi_1,\phi_2,\cdots)$，对于该序列来说：

（1）系统向量场 h_i 是 $\mathcal{PH}$ 的前 m 个元素。

（2）如果 $d(\phi_i)<d(\phi_j)$，那么 $i<j$。

（3）每个 $[\phi_i,\phi_j]\in\mathcal{PH}$，当且仅当：a）$\phi_i,\phi_j\in\mathcal{PH}$ 和 $i<j$，并且 b）对于某些 i，有 $\phi_j=h_i$，或者对于某些 $\phi_l,\phi_r\in\mathcal{PH}$，$\phi_j=[\phi_l,\phi_r]$，使得 $l\leqslant i$。

许多代数书籍（例如文献[863]）指出：该过程产生李代数 $\mathcal{L}(\Delta)$ 的一个基。文献[301]对计算基的各种算法进行了评估。

例 15.18（深度达到 3 的 P.Hall 基） P.Hall 基将李乘积排列成如下序列，其深度 $d=3$：

h_1,　h_2,　h_3,

$[h_1,h_2]$,　$[h_2,h_3]$,　$[h_1,h_3]$,

$[h_1,[h_1,h_2]],[h_1,[h_1,h_3]],[h_2,[h_1,h_2]],[h_2,[h_1,h_3]]$,

$[h_2,[h_2,h_3]],[h_3,[h_1,h_2]],[h_3,[h_1,h_3]],[h_3,[h_2,h_3]]$。

到目前为止，被 Jacobi 恒等式排除掉的李乘积只有$[h_1,[h_2,h_3]]$，因为：

$$[h_1,[h_2,h_3]]=[h_2,[h_1,h_3]]-[h_3,[h_1,h_2]] \tag{15.109}$$

这里给出的所有李乘积可能不是线性无关向量场。对于一个特定系统，应该进行线性无关检验来从基中删除线性相关的向量场。■

序列 $\mathcal{PH}$ 什么时候终止呢？回顾 $\dim(\mathcal{L}(\Delta))$ 不超过 n，因为 $\mathcal{L}_x(\Delta)\subseteq T_x(X)$。也就是说，在每个状态 $x\in X$，可能的无关速度向量的数量不超过 x 处切空间的维数。因此一旦得到 n 个无关向量场，就可以终止 $\mathcal{PH}$，因为不可能再发现新的无关向量场。对于某些系统，可能在深度 k 以后所有李括号都为零。这种系统称为 k 阶幂零的。例如，如果所有向量场的所有分量都是多项式，那么就会发生这种情况。如果系统不是幂零的，那么实现终止可能有困难。$\dim(\mathcal{L}(\Delta))$ 可能严格小于 n，但是这通常不能预先知道。很难确定是否需要更多的李括号来增加维数或者已经到达极限。

15.4.3.4　无漂移系统的可控性

无漂移仿射控制系统(15.53)的可控性可以用李代数秩条件(Lie algebra rank condition，LARC)来描述。回顾第 15.1.3 节的 STLC 的定义。假设 $U=\mathbb{R}^m$ 或 U 至少包含一个含 $\mathbb{R}^m$ 原点的开集。Chow-Rashevskii 定理[114,158,848]论述到：

一个无漂移仿射控制系统（15.53）在一个点 $x\in X$ 上是短时局部可控的（STLC），当且仅当 $\dim(\mathcal{L}(\Delta))=n$，其中 n 是 X 的维数。

如果对于每个 $x\in X$ 条件都成立，那么整个系统是 STLC 的。可积性也可以用 $\dim(\mathcal{L}(\Delta))$ 来表达。通常假设 $m<n$。有三种情况：

（1）$\dim(\mathcal{L}(\Delta))=m$　　系统是完全可积的。

（2）$m<\dim(\mathcal{L}(\Delta))<n$　　系统是非完整的，但不是 STLC 的。　　(15.110)

（3）$\dim(\mathcal{L}(\Delta))=n$　　系统是非完整的和 STLC 的。

例 15.19（可控性实例） 根据 Chow-Rashevskii 定理，差动机构、非完整积分器、例 15.17 中的系统都是 STLC 的，因为 $\dim(\mathcal{L}(\Delta))=n$。这意味着状态可以在任意方向上发生变化，即使有微分约束。可以使状态尽可能接近 X 中的任意平滑曲线。第 15.5.1 节给出基于李括号来实现这种情况的方法。这些系统都是 STLC 的，这种事实确保了满足第 14.6.2 节拓扑特性的 LPM 的存在性。■

15.4.3.5　处理有漂移的仿射控制系统

很难确定一个有漂移的系统（15.52）是否是 STLC 的。想象诸如气垫船这样的一个机械系统在高速运动。由于动量的缘故，它不可能在任意短的时间内从多数状态转换到某确定方向。但是从 $h_0(x)=0$ 的一个状态 $x\in X$，一个系统是否是 STLC 的呢？对于一个机械系统，这通常意味着它从静止开始。如果一个有漂移系统是 STLC 的，那么这直观地意味着

对于机械系统来说，如果它徘徊在接近零速的状态附近，那么它可以向任意方向运动。

李代数可以扩展到确定有漂移系统的可控性，但是需要的工具更加复杂。完整的介绍见文献[158]的第 7 章。即使 $\dim(\mathcal{L}(\Delta))=n$，这也不一定意味着系统是 STLC 的。它至少意味着系统是可达的，这就启发产生了第 15.1.3 节中给出的定义。因此能够得到的速度集合的维数仍然是 n；但是由于漂移的缘故，向所有方向运动可能是不可能的。为了得到 STLC，一个充分条件是 $\dot{x}$ 的可能的值的集合包含含原点的一个开集。

例 15.20（可达的，但不是 STLC 的） 如下简单系统清楚地阐述了漂移所带来的困难，文献[744]中考虑的就是该系统。令 $X=\mathbb{R}^2$ 和 $U=\mathbb{R}$，状态转移方程为：

$$\begin{aligned}\dot{x}_1 &= u \\ \dot{x}_2 &= \dot{x}_1^2\end{aligned} \tag{15.111}$$

从任何意义上，该系统都不是可控的，因为 x_2 不能减少。向量场是 $h_0(x)=[0 \quad x_1^2]^T$ 和 $h_1(x)=[1 \quad 0]^T$。第一个无关李括号是：

$$[h_1,[h_0,h_1]]=[0 \quad -2] \tag{15.112}$$

二维李代数为：

$$\mathcal{L}(\Delta)=\text{span}\{h_1,[h_1,[h_0,h_1]]\} \tag{15.113}$$

这意味着系统是可达的。但是它不是 STLC 的，因为括号 $[h_1,[h_0,h_1]]$ 用 h_0 构造，并且它是用一种效果不好的方式组合而成的。这种括号指出 x_2 是可能改变的，但是我们已经知道不可能减少 x_2。因此用涉及 h_0 的李括号所得到的某些向量场可能有方向约束。 ■

在例 15.20 中，$[h_1,[h_0,h_1]]$ 是一个坏括号（bad bracket）[926]，因为它妨碍了可控性。已经产生一种区分括号的“好”与“坏”的方法，并且存在通过满足关于好括号和坏括号的一定条件来指示一个有漂移的系统是否是 STLC 的定理。直观地，一定存在足够多的好括号来抵消坏括号对可控性所造成的影响[158,926]。

15.5 非完整性系统的操控方法

本节简要总结某些为非完整系统而解决 BVP 的方法。这可以认为是一个有微分约束、没有障碍物的运动规划问题。对于线性系统来说，可以采用第 15.2.2 节的最优控制技术。对于全驱动机械系统，可以采用诸如式（8.47）基于加速度的控制模型这样的标准控制技术。如果一个机械系统是欠驱动的，那么它可能是非完整的。如第 15.4 节所述，可能生成开始时被禁止的运动。假设根据 Chow-Rashevskii 定理，一个无漂移是 STLC 的。这就指出应该可能设计一个成功连接任意初始和目标状态对的 LPM。接下来是寻找一个行动轨迹 $\tilde{u}$，它使得 x_I 通过式（14.1）中的积分后达到 x_G。第 14 章的许多方法都可以使用，但是它们的速度都太慢。本节的方法利用系统的结构（例如它的李代数）和没有障碍物这个情况来更高效地解决规划问题。

15.5.1 使用 Philip Hall 基

本节介绍的操控方法归功于 Lafferriere 和 Sussmann[577]。这里假设给定一个无漂移仿

射控制系统，其中 X 是例 15.15 介绍的一个李群，并且假设系统是 STLC 的。本节概略介绍的操控方法遵循李代数 $\mathcal{L}(\Delta)$。思路是利用分段常数运动本原来向 P.Hall 基给定的方向运动。如果系统是幂零的，那么用该方法就能够准确到达目标状态。否则，它产生一个近似方法，迭代进行该近似方法就可以任意接近目标，并且某些系统通过反馈是可幂零化的[445]。

主要思路是从式（15.53）开始并构造一个扩展系统：

$$\dot{x} = \sum_{i=1}^{s} b_i(x) v_i \tag{15.114}$$

其中每个 v_i 都是一个行动变量，b_i 是 P.Hall 基 $\mathcal{PH}$ 中的一个向量场。对于每个 $i \leqslant m$，式（15.114）的每项都是 $b_i(x)v_i = h_i(x)u_i$，这来自于原始系统。对于 $i > m$，每个 b_i 代表 $\mathcal{PH}$ 中的一个李乘积，v_i 是一个虚构的行动变量。称它是虚构的是因为 b_i 给出的速度不能用系统的单个行动变量来完成，$i > m$。一般情况下，s 可能大于 n，因为在每个点 $x \in X$，可能需要 $\mathcal{PH}$ 的不同子集来得到 n 个无关向量，并且包括更多的基元素能够简化某些计算。

例 15.21（非完整积分器的扩展系统） 非完整积分器（15.83）的扩展系统为：

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -x_2 \end{pmatrix} v_1 + \begin{pmatrix} 0 \\ 1 \\ x_1 \end{pmatrix} v_2 + \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} v_3 \tag{15.115}$$

前两项对应于原始系统。最后一项产生于李括号 $[h_1, h_2]$。只需要一个虚构的行动变量，因为在 $x \in X$ 上只有三个 P.Hall 向量场是无关的。

很容易使该系统沿着 $\mathbb{R}^3$ 中一条基于栅格的路径运动。用 $v_1 = u_1$ 和 $v_2 = u_2$ 分别得到 x_1 和 x_2 方向上的运动。为了产生 x_3 方向上的运动，应该进行式（15.71）中的换位子运动。这对应于应用 v_3。本节描述的操控方法是这种方法的推广。可以应用更高阶的李乘积，能够产生任意方向上的运动。 ■

假设给定某 x_I 和 x_G。操控方法有两个阶段：

（1）为扩展系统确定一个行动轨迹 $\tilde{v}$。对于该系统来说，$x(0) = x_I$；对于某 $t_F > 0$，$x(t_F) = x_G$。

（2）将 $\tilde{v}$ 转化为行动轨迹 $\tilde{u}$，$\tilde{u}$ 排除了虚构变量并使用实际的 m 个变量 $u_1, \cdots, u_m$。

第一个阶段很简单。对于扩展系统来说，可以生成切空间 $T_x(X)$ 中的任意速度。从任意平滑路径 $\tau : [0,1] \to X$ 开始使得 $\tau(0) = x_I$ 和 $\tau(1) = x_G$。沿着路径 τ 的速度 $\dot{\tau}(t)$ 是 $T_{\tau(t)}(X)$ 中的一个速度向量，它可以表示为向量 $b_i(\tau(t))$ 的一个线性组合。这种组合的系数是 v_i 的值。第二阶段则更加复杂，稍后对给阶段进行介绍。如果系统是幂零的，那么 $\tilde{u}$ 应该将系统从 x_I 精确地带到 x_G。通过构造它的方法，很清楚知道如何改进 $\tilde{u}$ 来任意接近 $\tilde{v}$ 所产生的轨迹。

正规计算　第二阶段采用正规代数来进行计算。这意味着可以忽略特定的向量场、微分、流形等。概念只涉及纯粹的代数运算。为了避免与前面的定义混淆，在接下来的许多定义中加入“正规的”这个术语。回顾第 4.4.1 节代数多项式的定义（例如 $\mathbb{F}[x_1, \cdots, x_n]$）。令 $A(y_1, \cdots, y_m)$ 表示变量 $y_1, \cdots, y_m$ 的多项式的**正规非可交换代数**（formal noncommutative

algebra）[1]。这里的 y_i 被当成一个符号来对待，它没有其他假设性质（例如它们不必是向量场）。当在代数中进行多项式乘法时，不能基于交换律来进行简化。通过定义一个李括号可以将代数转化为一个李代数。对于任意两个多项式 $p,q \in A(y_1,\cdots,y_m)$，定义正规李括号为 $[p,q]=pq-qp$。正规李括号产生代数上的一个等价关系，这产生了一个正规李代数 $L(y_1,\cdots,y_m)$（当应用正规李括号时，代数的相同元素有许多等价表示）。正规代数和正规李代数的幂零版本可以通过使所有 $k+1$ 阶单项式为零来产生。它们分别用 $A_k(y_1,\cdots,y_m)$ 和 $L_k(y_1,\cdots,y_m)$ 来表示。可以应用 P.Hall 基来得到正规李代数的一个基。用正规计算，例 15.18 实际上对应于 $L_3(h_1,h_2,h_3)$ 的基。

指数映射 操控问题将用 $L_k(y_1,\cdots,y_m)$ 上的计算来解决。$A(y_1,\cdots,y_m)$ 的正规幂级数是单项式的所有线性组合的集合，其中包括那些有无限多项的情况。相似地，也可以定义 $L(y_1,\cdots,y_m)$ 的正规李级数。

对于任意 $p \in A(y_1,\cdots,y_m)$，正规指数映射定义为：

$$e^p = 1 + p + \frac{1}{2!}p^2 + \frac{1}{3!}p^3 + \cdots \tag{15.116}$$

在幂零情况下，对于任意 $p \in A_k(y_1,\cdots,y_m)$，正规指数映射定义为：

$$e^p = \sum_{i=0}^{k} \frac{p^i}{i!} \tag{15.117}$$

正规级数被裁剪，因为所有大于 k 的指数项成为零。

构造正规李群：

$$G_k(y_1,\cdots,y_m) = \left\{ e^p \,|\, p \in L_k(y_1,\cdots,y_m) \right\} \tag{15.118}$$

如果正规李代数不是幂零的，那么可以将一个正规李群 $G(y_1,\cdots,y_m)$ 定义为所有 e^p 的集合，其中 p 用一个正规李级数来表达。

如下例子来自于文献[577]。

例 15.22（正规李群） 假设给定生成元 x 和 y。正规李群 $G(x,y)$ 的某些元素为：

$$e^x = I + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \cdots \tag{15.119}$$

$$e^{[x,y]} = I + [x,y] + \frac{1}{2}[x,y]^2 + \cdots \tag{15.120}$$

$$e^{x-y+3[x,y]} = I + x - y + 3[x,y] + \cdots \tag{15.121}$$

其中 I 是正规李群的单位元。正规李群 $G_2(x,y)$ 的某些元素为：

$$e^x = I + x + \frac{1}{2}x^2 \tag{15.122}$$

$$e^{[x,y]} = I + [x,y] \tag{15.123}$$

$$e^{x-y+3[x,y]} = I + x - y + 3[x,y] + \frac{1}{2}(x-y)^2 \tag{15.124}$$

■

为了成为一个群，第 4.2.1 节中给出的公理必须得到满足。单位元为 I，结合律遵循级

1 从直觉上看，一个代数意味着多项式可以进行加法和乘法。对于所有所要求的公理，参见文献[472]。

数表示。每个 e^p 都有一个逆元 e^{-p}，因为 $e^p e^{-p} = I$。还需要满足的一个公理是闭包。这由 Campbell-Baker-Hausdorff-Dynkin 公式（或者 CBHD 公式）给出，对于它来说，对于任意 $p,q \in G(y_1,\cdots,y_m)$ 的第一项为：

$$\exp(p)\exp(q) = \exp(p+q+\frac{1}{2}[p,q]+\frac{1}{12}[[p,q],q]-\frac{1}{12}[[p,q],p]+\frac{1}{24}[p,[q,[p,q]]]+\cdots) \tag{15.125}$$

其中 $\exp(x)$ 表示 e^x。公式也应用于 $G_k(y_1,\cdots,y_m)$，但是它被裁剪成一个有限级数。这个事实将在后面用到。注意到 $e^p e^q \neq e^{p+q}$，这不同于标准指数定义。

CBHD 公式通常表示为：

$$e^p e^q e^{-p} = \exp\left(\sum_{i=0}^{\infty}\frac{Ad_p^i q}{i!}\right) \tag{15.126}$$

其中 $Ad_p^0 q = q$ 和 $Ad_p^i q = [p, Ad_p^{i-1} q]$。算子 Ad 提供了一种表达嵌套李括号运算的紧凑方式。式（15.125）的另外的项可以用式（15.126）来得到。

Chen-Fliess 级数　第 15.4.3 节的 P.Hall 基一般情况下可以应用于任意李代数。令 $B_1,\cdots,B_s$ 表示幂零正规李代数 $L_k(y_1,\cdots,y_m)$ 的一个 P.Hall 基。正规李群的研究中有一个重要的定理是：每个 $S \in G_k(y_1,\cdots,y_m)$ 可以用它的正规李代数的 P.Hall 基来表达：

$$S = e^{z_s B_s} e^{z_{s-1} B_{s-1}} \cdots e^{z_2 B_2} e^{z_1 B_1} \tag{15.127}$$

这被称为 Chen-Fliess 级数。z_i 有时称为 S 的后向 P.Hall 坐标（有一个前向版本，（15.127）中的项是从 1 到 s，而不是从 s 到 1）。

回到系统向量场　现在，正规代数的概念可以被应用于操控问题。变量变成系统向量场：对于所有从 1 到 m 的 i，$y_i = h_i$。对于 P.Hall 基元素，每个 B_i 都变成 b_i。李群变成状态空间 X，李代数是第 15.4.3 节介绍的向量场上的李代数。考虑李群的一个元素必须如何随时间进行演化。这可以用微分方程表示：

$$\dot{S}(t) = S(t)(v_1 b_1 + v_2 b_2 + \cdots + v_s b_s) \tag{15.128}$$

它被初始化为 $S(0) = I$。这里 S 可以阐述为一个矩阵，比如它可能属于 $SE(3)$。

每个时刻 $t > 0$ 的解可以用 Chen-Fliess 级数（15.127）写为：

$$S(t) = e^{z_s(t) b_s} e^{z_{s-1}(t) b_{s-1}} \cdots e^{z_2(t) b_2} e^{z_1(t) b_1} \tag{15.129}$$

即通过对 b_1 进行 $z_1(t)$ 时间长度积分、对 b_2 进行 $z_2(t)$ 时间长度积分，如此进行下去直到对 b_s 进行 $z_s(t)$ 时间长度积分，这样可以得到 $S(t)$。注意到后向 P.Hall 坐标现在随时间而变化。如果我们知道它们是如何随时间而演化的，那么就可以解式（15.128）中的微分方程。

下一步是发现后向 P.Hall 坐标是如何演化的。式（15.129）对时间的微分产生：

$$\dot{S}(t) = \sum_{j=1}^{s} e^{z_s b_s} \cdots e^{z_{j+1} b_{j+1}} \dot{z}_j b_j e^{z_j b_j} \cdots e^{z_1 b_1} \tag{15.130}$$

Chen-Fliess-Sussmann 方程　现在 $\dot{S}$ 有两种表达方式，它们由式(15.128)和式(15.130)给出。使它们相等，那么得到 s 方程：

$$\sum_{j=1}^{s} p_{j,k} \dot{z}_j = v_k \tag{15.131}$$

其中 $p_{j,k}$ 是关于 z_i 的多项式。这用到指数的级数表示，参见例 15.23。

因此 Chen-Fliess-Sussmann （CFS）方程：

$$\dot{z} = Q(z)v \tag{15.132}$$

给出后向 P.Hall 坐标的演化过程，其中 $Q(z)$ 是一个 $s \times s$ 矩阵，$z(0)=0$。$Q(z)$ 中的项都是多项式，因此可以对系统进行解析积分得到 $z_i(t)$ 的表达式。

给出一个简单例子，它来自于文献[301]。

例 15.23（非完整性积分器的 CFS 方程） 式（15.115）给出非完整性积分器的扩展系统。李群的微分方程（15.128）是：

$$\dot{S}(t) = S(t)(v_1 b_1 + v_2 b_2 + v_3 b_3) \tag{15.133}$$

因为 $s=3$。

它的解有两种表达式。Chen-Fliess 级数（15.129）变成：

$$S(t) = e^{z_3(t)b_3} e^{z_2(t)b_2} e^{z_1(t)b_1} \tag{15.134}$$

如果对于 i 从 1 到 3，有 $z_i(0)=0$，那么满足初始条件 $S(0)=I$。$\dot{S}(t)$ 的第二种表达式是（15.130），它在非完整性积分器情况下变成：

$$\dot{S}(t) = \dot{z}_3(t)b_3 e^{z_3(t)b_3} e^{z_2(t)b_2} e^{z_1(t)b_1} + e^{z_3(t)b_3}\dot{z}_2(t)b_2 e^{z_2(t)b_2} e^{z_1(t)B_1} + e^{z_3(t)b_3} e^{z_2(t)b_2}\dot{z}_1(t)b_1 e^{z_1(t)b_1} \tag{15.135}$$

注意到：

$$S^{-1}(t) = e^{-z_1(t)B_1} e^{-z_2(t)B_2} e^{-z_3(t)B_3} \tag{15.136}$$

使式（15.133）和式（15.135）相等就产生：

$$S^{-1}\dot{S} = v_1 b_1 + v_2 b_2 + v_3 b_3 = e^{-z_1 b_1} e^{-z_2 b_2} e^{-z_3 b_3}\dot{z}_3 b_3 e^{z_3 b_3} e^{z_2 b_2} e^{z_1 b_1} + e^{-z_1 b_1} e^{-z_2 b_2}\dot{z}_2 b_2 e^{z_2 b_2} e^{z_1 b_1} + e^{-z_1 b_1}\dot{z}_1 b_1 e^{z_1 b_1} \tag{15.137}$$

其中为了缩短表达式而省去了有关时间的符号。现在使用指数的正规李级数扩展。对于 $i=1,2$，有：

$$e^{z_i b_i} = \left(I + z_i b_i + \frac{1}{2} z_i^2 b_i^2\right) \tag{15.138}$$

$$e^{-z_i b_i} = \left(I - z_i b_i - \frac{1}{2} z_i^2 b_i^2\right) \tag{15.139}$$

也有：

$$e^{z_3 b_3} = (I + z_3 b_3) \tag{15.140}$$

$$e^{-z_3 b_3} = (I - z_3 b_3) \tag{15.141}$$

在式（15.140）和式（15.141）中很显然进行了裁剪。没有 b_3^2 项是因为 b_3 是一个二阶多项式，它的平方是四阶的。

代入式（15.137），进行非可交换的乘法，并应用李括号定义，这样就产生了：

$$\dot{z}_1 b_1 + \dot{z}_2 (b_2 - z_1 b_3) + \dot{z}_3 b_3 = v_1 b_1 + v_2 b_2 + v_3 b_3 \tag{15.142}$$

使相应项相等，产生 Chen-Fliess-Sussmann 方程：

$$\begin{aligned} \dot{z}_1 &= v_1 \\ \dot{z}_2 &= v_2 \\ \dot{z}_3 &= v_3 + z_1 v_2 \end{aligned} \tag{15.143}$$

回顾到 $\tilde{v}$ 是给定的。从 $z(0)=0$ 对式（15.143）进行积分，得到后向 P.Hall 坐标轨迹 $\tilde{z}$。

使用原始行动变量　一旦确定 CFS 方程，问题基本上就解决了。行动轨迹 $\tilde{v}$ 从给定的状态轨迹 $\tilde{x}$ 来确定，后向 P.Hall 坐标轨迹 $\tilde{z}$ 由式（15.143）来确定。那么剩下的问题就是从 v_{m+1} 到 v_s 的行动变量是虚构的，因为与它们相关的向量场不是系统的一部分。它们是从李括号运算中产生的。当应用时会它们会相互影响，因为其中一些可能会在相同的时间试图应用原始系统的相同的 u_i 变量。

CBHD 公式用来根据系统行动变量 $u_1,\cdots,u_m$ 来确定问题的解。微分方程现在变成：

$$\dot{S}(t)=S(t)(u_1b_1+u_2b_2+\cdots+u_mb_m) \tag{15.144}$$

它用 $S(0)=I$ 来初始化，并使用原始系统，而不是扩展系统。

当应用向量场时，CBHD 公式变成：

$$\exp(tf)\exp(tg)=\exp(tf+tg+\frac{t^2}{2}[f,g]+\frac{t^3}{12}[[f,g],g]-\frac{t^3}{12}[[f,g],f]+\frac{t^4}{24}[f,[g,[f,g]]]+\cdots) \tag{15.145}$$

如果系统是幂零的，那么该级数将被裁剪，并且可以确定常数运动本原序列所产生的准确效果。这导致了确定常数运动本原的一个有限序列的过程，该过程在扩展系统所规定的方向上生成一个运动以及行动轨迹 $\tilde{v}$。

15.5.2　采用正弦行动轨迹

本节提出的操控方法是基于 Brockett 的工作[144]以及 Murray 和 Sastry 对它的推广[730]。方法应用于多种类型的系统，对于这些系统来说，无关向量场尽可能快地增长。这意味着当构造 P.Hall 基时，由于在以前李乘积或者系统向量场上线性相关的缘故，不需要去除元素。对于这种系统，方法采用将频率与某些行动变量完整地联系起来的正弦形式。这改变了某些状态变量，而其他一些则自动地保持固定。更详细的内容见文献[599,728,730,848]。

15.5.2.1　操控非完整性积分器

对于如下非完整性积分器，方法的主要思路可以清楚地得到阐述：

$$\begin{aligned}\dot{x}_1&=u_1\\ \dot{x}_2&=u_2\\ \dot{x}_3&=x_1u_2-x_2u_1,\end{aligned} \tag{15.146}$$

整个第 15.5.1 节考虑的都是它。这种情况将得到详细说明，并将论述通过对原理的推广所得到的方法。这里给出的表示是基于文献[730,848]的。

如同以前所指出的，无关向量场尽快地增长是非常重要的。对于非完整性系统，$[h_1,h_2]$ 与 h_1 和 h_2 线性无关，如第 15.12 节所述，因此它满足该特性。考虑操控系统从某 $x_I=x(0)$ 到达某 $x_G=x(1)$，同时最优化代价泛函：

$$\int_0^1(u_1(t)^2+u_2(t)^2)\mathrm{d}t \tag{15.147}$$

问题可以用第 13.4.3 节给出的受约束的拉格朗日表述来解决。第一步是去除变量 u。根据式（15.146），通过 $\dot{x}_1=u_1$ 和 $\dot{x}_2=u_2$，代价可以用 $\dot{x}_1$ 和 $\dot{x}_2$ 来表达。式（15.146）的第三个方程可以记为：

$$\dot{x}_3=x_1\dot{x}_2-x_2\dot{x}_1 \tag{15.148}$$

并阐述为拉格朗日函数上的一个约束，它用第 13.4.3 节介绍的一个（标量）拉格朗日乘子进行组合。拉格朗日函数定义为：

$$L(x,\dot{x})=\left(\dot{x}_1^2+\dot{x}_2^2\right)+\lambda\left(\dot{x}_3-x_1\dot{x}_2+x_2\dot{x}_1\right) \tag{15.149}$$

其中第一项来自于式（15.147）的被积函数，第二项来自于式（15.148）。

欧拉-拉格朗日方程（13.118）产生：

$$\begin{aligned}\ddot{x}_1+\lambda\dot{x}_2&=0\\ \ddot{x}_2-\lambda\dot{x}_1&=0\\ \dot{\lambda}&=0\end{aligned} \tag{15.150}$$

注意到 $\dot{\lambda}=0$ 意味着 $\lambda(t)$ 是常数。为了得到一个描述最优行动轨迹的微分方程，要用到这样一个事实：对于 $i=1,2$，$\dot{x}_i=u_i$ 且 $\ddot{x}_i=\dot{u}_i$。这产生方程 $\dot{u}_1=-\lambda u_2$ 和 $\dot{u}_2=\lambda\dot{u}_1$。这可以阐述为一个二阶线性微分方程。基于它的根，解为：

$$\begin{aligned}u_1(t)&=u_1(0)\cos\lambda t-u_2(0)\sin\lambda t\\ u_2(t)&=u_1(0)\sin\lambda t+u_2(0)\cos\lambda t\end{aligned} \tag{15.151}$$

给定初始状态和目标状态，通过确定 $u_1(0)$、$u_2(0)$ 和 λ 就可以得到最优行动轨迹。假设 $x_I=x(0)=(0,0,0)$ 和 $x_G=x(1)=(0,0,a)$，其中 $a\in\mathbb{R}$。通过对解进行 $SE(3)$ 中的变换可以得到其他情况。

对式（15.151）进行积分可以得到 x_1 和 x_2 的状态轨迹，因为对于 $i=1$ 和 $i=2$，有 $u_i=\dot{x}_i$。从 $x_1(0)=x_2(0)=0$ 开始，这产生：

$$\begin{aligned}x_1(t)&=\frac{1}{\lambda}\left(u_1(0)\sin\lambda t+u_2(0)\cos\lambda t-u_2(0)\right)\\ x_2(t)&=\frac{1}{\lambda}\left(-u_1(0)\cos\lambda t+u_2(0)\sin\lambda t+u_1(0)\right)\end{aligned} \tag{15.152}$$

为了保持约束 $x_1(1)=x_2(1)=0$，必须选择 λ 为 $\lambda=2k\pi$，对某整数 k。$\dot{x}_3$ 的积分产生

$$x_3(t)=\int_0^1\left(x_1u_2-x_2u_1\right)\mathrm{d}t=\frac{1}{\lambda}\left(u_1^2(0)+u_2^2(0)\right)=a \tag{15.153}$$

代价是：

$$\int_0^1\left(u_1^2(t)+u_2^2(t)\right)\mathrm{d}t=u_1^2(0)+u_2^2(0)=\lambda a \tag{15.154}$$

因此 $k=-1$ 时得到最小值，产生 $\lambda=2\pi$ 和 $\|u\|=2\pi a$。这确定了 $u(0)$ 的值，但是可以选择任意方向。

操控问题的解决分为两个阶段：

（1）应用任意行动轨迹将 x_1 和 x_2 驱动到所要求的值，同时忽略 x_3。

（2）应用得到的解将 x_3 驱动到目标，同时 x_1 和 x_2 返回到它们在第一阶段得到的值上。

这种思路可以推广到其他系统。

15.5.2.2 一阶可控系统

对非完整性积分器所采用的方法可推广到如下形式的系统：

$$\begin{aligned}\dot{x}_i&=u_i\qquad(1\leqslant i\leqslant m)\\ \dot{x}_{ij}&=x_iu_j-x_ju_i\qquad(i<j,1\leqslant j\leqslant m)\end{aligned} \tag{15.155}$$

和

$$\begin{aligned}\dot{x}_i &= u_i \qquad (1 \leqslant i \leqslant m)\\ \dot{x}_{ij} &= x_i u_j \qquad (i < j, 1 \leqslant j \leqslant m)\end{aligned} \tag{15.156}$$

Brockett 在文献[144]中指出，对于这种一阶可控系统来说，最优行动轨迹是通过为 m 个行动变量中的每个变量应用相关频率的正弦曲线之和来得到的。如果 m 是偶数，那么每个变量的轨迹是频率为 $2\pi, 2\cdot 2\pi, \cdots, (m/2)\cdot 2\pi$ 的 $m/2$ 个正弦曲线之和。如果 m 是奇数，那么有 $(m-1)/2$ 个正弦曲线，频率的序列还是相同的。假设 m 是偶数（奇数情况类似），选择的每个行动为：

$$u_i = \sum_{k=1}^{m/2} a_{ik} \sin 2\pi kt + b_{ik} \cos 2\pi kt \tag{15.157}$$

其他状态变量的演化过程为：

$$x_{ij} = x_{ij}(0) + \frac{1}{2}\sum_{k=1}^{m/2} \frac{1}{k}\left(a_{jk} b_{ik} - a_{ik} b_{jk}\right) \tag{15.158}$$

它提供了一个类似于式（15.53）的约束。在驱动 x_{ij} 到它们所需要的值的同时，行动变量的这种周期行为引起 x_i 变量返回到它们的原始值。从某种意义上讲，这是一个基于向量的推广，其中的标量情况就是以前的非完整积分器。

操控方法仍分为两阶段：

（1）应用将每个 x_i 带到其目标值的任意行动轨迹。这一阶段不考虑 x_{ij} 状态的演化过程。

（2）将相关频率的正弦曲线应用于行动变量。选择每个行动 $u_i(0)$ 使得 x_{ij} 到达它的目标值。在这个阶段不考虑 x_i 变量，因为它们将返回到它们在第一阶段所得到的值。

这种方法已经进一步推广到二阶可控系统：

$$\begin{aligned}\dot{x}_i &= u_i \qquad (1 \leqslant i \leqslant m)\\ \dot{x}_{ij} &= x_i u_j \qquad (i < j, 1 \leqslant j \leqslant m)\\ \dot{x}_{ijk} &= x_{ij} u_k \qquad ((i,j,k) \in J)\end{aligned} \tag{15.159}$$

其中 J 是不同的 $i, j, k \in \{1, \cdots, m\}$ 所形成的所有三元组的集合并除去由李括号的 Jacobi 恒等式所产生的不必要的排列。对于这个问题，应用类似于一阶可控系统中所用的思路可以得到一个三阶段操控方法。第一阶段确定 x_i，第二阶段处理 x_{ij}，第三阶段解决 x_{ijk}。详细内容见文献[730,848]。

15.5.2.3　链式系统

例 15.17 考虑了链式（chained-form）系统的一种特殊情况。式（15.102）中的系统可以被推广到任意 n：

$$\begin{aligned}\dot{x}_1 &= u_1 \qquad & \dot{x}_4 &= x_3 u_1\\ \dot{x}_2 &= u_2 \qquad & &\vdots\\ \dot{x}_3 &= x_2 u_1 \qquad & \dot{x}_n &= x_{n-1} u_1\end{aligned} \tag{15.160}$$

这可以认为是一个高阶可控系统。对于这些系统，得到的是一个多阶段方法：

（1）应用将 x_1 和 x_2 带到其目标值的 u_1 和 u_2 的任意行动轨迹。这一阶段不考虑其他状态的演化过程。

（2）对于从 3 到 n 的每个 k，本阶段重复进行。将 x_k 驱动到它的目标值，应用：

$$u_1 = a\sin 2\pi kt \quad 和 \quad u_2 = b\cos 2\pi kt \tag{15.161}$$

其中选择的 a 和 b 满足约束：

$$\dot{x}_k(1) = x_k(0) + \left(\frac{a}{4\pi}\right)^{(k-2)} \frac{b}{(k-2)!} \tag{15.162}$$

本阶段的每次执行过程都引起前面的 $k-1$ 个状态变量返回到它们以前的值。

关于第二阶段正确性的证明以及更多一般信息，参见文献[730,848]。只有很少系统符合本节给出的形式，但是有时可能将系统进行变换以符合这种形式。回顾式（13.15）中的原始简易汽车模型被简化到式（15.54）。现在有许多方法能够将系统变换成链的形式。对于仍然不能变换为这种形式的系统，可以采用傅里叶变换来得到与本节相似的近似操控方法。当链式系统用 Pfaffian 约束来表达时，其结果通常被称为 Goursat 正规形式。方法还可以进一步扩展到多链系统。

15.5.3 其他操控方法

到目前为止所介绍的操控方法是广为人知的。但是还存在其他方法，这些方法大都按照第 15.5.1 节和第 15.5.2 节的思路，即利用了系统的特殊类型的特征。这里简要介绍某些方法。这是一个很活跃的研究领域，在未来的一些年里很可能会产生许多新的方法。

微分平滑系统 在操控方法的研究中，微分平滑已经成为一个重要概念。它是 Fliess、Lévine、Martin 和 Rouchon 在文献[346]中提出的，也可参见文献[729]。从直观上看，系统被称为是微分平滑的，那么如果能够发现一个被称为平滑输出的变量的集合，对于该集合来说所有状态和行动都可以由该集合来确定，而不需要进行积分。这意味着，对于一个系统 $\dot{x} = f(x,u)$，$X = \mathbb{R}^n$ 和 $U = \mathbb{R}^m$，存在如下形式的平滑输出：

$$y = h(x,u,\dot{u},\cdots,u^{(k)}) \tag{15.163}$$

使得存在函数 g 和 g'，有：

$$x = g(y,\dot{y},\cdots,y^{(j)}) \tag{15.164}$$

$$u = g'(y,\dot{y},\cdots,y^{(j)}) \tag{15.165}$$

在式（13.19）中所表达的牵引拖车的汽车就是这样一个例子，平滑输出是最后一辆拖车在 $\mathcal{W} = \mathbb{R}^2$ 中的位置。这种特性在文献[581]中为运动规划所用。最近关于微分平滑系统的操控方面的工作包括文献[581,816,836]。

解耦向量场 对于考虑动力学的机械系统，漂移使操控问题复杂化。最近有一种方法是基于建立系统的运动可控性，也就是说如果行进的轨迹以零速状态开始和结束，那么系统在 C-空间上是 STLC 的[159]。这种方法发现 C-空间上的解耦向量场。C-空间中一个解耦向量场的积分曲线所构成的任何路径对所有动力学系统来说都是可执行的。如果一个机械系统允许这种向量场，那么文献[159]证明：C 上的一个操控方法可以提升到机械系统的相空间 X 上。这种思路在文献[226]中的一个 RRT 规划器中被用来生成一种有效的 LPM。

平均法 通过将状态轨迹分解为一个考虑状态的大范围演化的低频部分和一个考虑小幅波动的高频部分，**平均法**（averaging method）对状态轨迹产生系统的扰动。这产生基于正弦行动轨迹的其他操控方法[114,423,626,627]。

变分法　可能如所期望的那样，第 14.7 节的一般意义的基于梯度的优化技术可以用来操控非完整系统。该方法是基于在可能状态轨迹的空间上进行牛顿迭代。这导致了一种到达局部最优的梯度下降，同时还能够满足微分约束。关于应用该技术来操控非完整系统的更详细内容请参见文献[278,336,599,902,927]。

庞特里亚金最小值原理　最小值原理在操控方法的研究中非常有用。由于欧拉-拉格朗日方程和哈密顿方程之间的密切联系，这并不令人感到吃惊。第 15.5.2 节中用欧拉-拉格朗日方程来确定非完整积分器的最优操控方法。文献[848]描述了一种基于最小值原理的操控方法。第 15.3 节的最优曲线实际上表示了用最小值原理所得到的操控方法。然而遗憾的是，对于绝大多数问题来说，需要用数值技术来解微分方程。通常期望为诸如幂零的、链式的或者微分平滑的等特殊类型的系统而开发的方法比应用于欧拉-拉格朗日方程、哈密顿方程或者庞特里亚金最小值原理的一般目的的数值技术性能更好。

动态规划　可以用第 14.5 节的数值动态规划方法来产生任何系统的最优操控方法。为了在这里应用它，障碍区域 X_{free} 为空。但是主要的不足之处是计算量通常非常大，特别是如果 X 的维数很高时。另一方面，它应用在很一般的情况下，可以用李群对称性来从初始状态应用预先计算好的轨迹。对于状态空间是 $SE(2)$ 或者 $SO(3)$ 的系统来说，这是一个可行的方法。

进一步阅读

许多关于控制的书籍中，特别是专门研究非线性控制的书中，都介绍了第 15.1 节中的基本的稳定性和可控性的概念；关于非线性控制的入门书籍见文献[526,848]。更多先进概念出现在文献[158]中。为了阐述向量场的许多收敛特性，可以参考文献[44]。对于线性系统理论，参见文献[194]。Brockett 条件及推广出现在文献[145,996]中。更加关于非完整系统的镇定和反馈控制的内容参见文献[158,848,964]。基于李亚普诺夫的反馈控制设计方法参见文献[280]。

对 Hamilton-Jacobi-Bellman 方程的进一步阅读见文献[87,97,495,792,913]。求解的数值方法（除值迭代以外）参见文献[2,255,710]。文献[28,573]介绍了线性二次型问题。庞特里亚金的原始工作提供了对最小值原理的一种不平常的解释和说明[804]。其他方面的内容参见文献[97,412,792]。融合状态空间约束的一种推广出现在文献[928]中。

第 15.3 节基于文献[65,130,213,296,817,904,905,924]。最优曲线在其他一些情况下已经部分地得到描述，参见文献[229,904]。一种复杂性是最优曲线通常涉及无限次切换[372,1000]。我们应致力于研究非最优曲线，尽管它们不是最优的，但是却具有良好的特性，特别是用作车形机器人局部规划方法[31,360,523,797,850]。车形机器人的反馈控制参见文献[114,666]。

关于非完整系统理论的进一步阅读资料，有许多好的文献（文献 [85,114,115,158,481,728,744,848]）。文献[727]提出将 Chow-Rashevskii 定理推广到组合系统。文献[597]研究了汽车牵引拖车的稳定性问题。文献[672]考虑了带推进器的二维气垫船的可控性问题。“完整的”这个术语来源于意思为“可积的”和“规律”的两个希腊词汇[137]。

第 15.5 节主要是基于文献[577]（见第 15.5.1 节）和文献[144,730]（见第 15.5.2 节）中的操控方法。文献[301]将第 15.5.1 节的方法扩展到时变系统。文献[716]研究开发了一种多速率版本。在文献[483]中，用 Lyndon 基能够提高方法的性能，而不是用 P.Hall 基。另外的涉及级数的操控方法参见文献[156,157]。更多关于链式系统，参见文献[860,903]。用多项式和 Goursat 正规形式代替正弦曲线的变化版本参见文献[848]。其他操控方法见第 15.5.3 节所建议的参考文献。

习　题

1. 描述 $X=\mathbb{R}^2$ 上向量场在 $(0,0)$ 点的稳定性，给定 $\dot{x}_1=x_2$ 和 $\dot{x}_2=-x_2^2-x_1$。用李亚普诺夫函数 $\phi(x_1,x_2)=x_1^2+x_2^2$。

2. 重复例 15.4，但是代价项用 $l(x,u)=u^2$。

3. 重复例 15.4，但是用一个三重积分器 $q^{(3)}=u$ 和 $U=[-1,1]$。

4. 确定使得例 15.4 中四种情况分别发生的精确条件。定义一个产生时间最优的运动的反馈运动规划。

5. 注意到 Dubins 车的六个最优字中某些字对于 Reed-Shepp 来说并不出现。对于每种情况，阐述不出现的原因。

6. 回顾第 15.4.2 节中用泰勒级数产生李括号的步骤。回到式（15.81）详细叙述所有步骤（第 15.4.2 节跳过了一些步骤细节）。

7. 确定如下系统是否是非完整的和 STLC 的：

$$\begin{aligned} \dot{q}_1&=u_1 & \dot{q}_4&=q_2^2u_1 \\ \dot{q}_2&=u_2 & \dot{q}_5&=q_1^2u_2 \\ \dot{q}_3&=q_1u_2-q_2u_1 \end{aligned} \tag{15.166}$$

8. 证明常系数矩阵 A 和 B 的线性系统 $\dot{x}=Ax+Bu$ 不能是非完整性的。

9. 确定如下系统是否是非完整的和 STLC 的：

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \\ -\sin\psi \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} u_2 \tag{15.167}$$

10. 对差动机构用换位子运动和常数行动，在它的位形空间上开发一个网格。

11. 考虑一个平滑非线性系统，它仅有一个行动变量和一个 n 维状态空间，$n>1$。这种系统总是完全可积的、非完整的吗？或者是两者之一？

12. 将例 15.17 推广到 $\mathbb{R}^n$，并有两个行动变量。对于任意 $n>5$，确定系统是否是 STLC 的。

13. 证明当向量叉乘用作一个括号运算时，它在 $\mathbb{R}^3$ 上确实能够产生一个李代数。

14. 产生如下系统的 CFS 方程：

$$\begin{aligned} \dot{q}_1 &= u_1 & \dot{q}_3 &= q_1 u_2 - q_2 u_1 \\ \dot{q}_2 &= u_2 & \dot{q}_4 &= q_2^2 u_1 \end{aligned} \tag{15.168}$$

实现

15. 用软件实现任意阶的 P.Hall 基的计算（这仅仅是符号计算，不扩展李括号）。

16. 用软件显示 Dubins 车任意给定的 q_I 和 q_G 之间的最优路径。

17. 用第 14.4.2 节中的规划算法来数值确定 Dubins 曲线。用 Dijkstra 算法来进行搜索、用一个高分辨率的栅格。你的软件能够得到与 Dubins 曲线相同的曲线集合吗？

18. 进行试验，将 Dubins 曲线用作一个局部规划方法（LPM）和基于 RRT 的规划算法中的度量。使用曲线能够改进执行时间吗？它们能够导致更好的解吗？

参 考 文 献

[1] D.Aarno,D.Kragic,and H.I.Christensen, Artificial potential biased probabilistic roadmap method. In *Proceedings IEEE International Conference on Robotics & Automation*, 2004.

[2] R.Abgrall.Numerical discretization of the first-order Hamilton-Jacobi equation on triangular meshes. *Communications on Pure and Applied Mathematics*, 49(12):1339–1373, December 1996.

[3] R.Abraham and J. Marsden. *Foundations of Mechanics*. Addison-Wesley, Reading, MA, 2002.

[4] R.Abraham, J.Marsden, and T. Ratiu. *Manifolds, Tensor Analysis*, and *Applications, 2nd Ed.* Springer -Verlag, Berlin, 1988.

[5] A.Abrams and R.Ghrist. Finding topology in a factory: Configuration spaces. *The American Mathematics Monthly*, 109:140–150, February 2002.

[6] E.U.Acar and H.Choset. Complete sensor-based coverage with extended-range detectors: A hierarchical decomposition in terms of critical points and Voronoi diagrams. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[7] E.U.Acar and H. Choset. Robust sensor-based coverage of unstructured environments. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[8] C.C.Adams. The Knot Book: *An Elementary Introduction to the Mathematical Theory of Knots*. W. H. Freeman, New York, 1994.

[9] P.Agarwal, M.de Berg, D.Halperin, and M. Sharir. Efficient generation of k-directional assembly sequences. In *ACM Symposium on Discrete Algorithms*, pages 122–131, 1996.

[10] P.K.Agarwal, N.Amenta, B.Aronov, and M.Sharir. Largest placements and motion planning of a convex polygon. In J.-P. Laumond and M.Overmars, editors, *Robotics: The Algorithmic Perspective*. A.K. Peters, Wellesley, MA, 1996.

[11] P.K.Agarwal, B.Aronov, and M.Sharir. Motion planning for a convex polygon in a polygonal environment. *Discrete and Computational Geometry*, 22:201–221, 1999.

[12] P.K.Agarwal, J.-C.Latombe, R.Motwani, and P. Raghavan. Nonholonomic path planning for pushing a disk among obstacles. In *Proceedings IEEE International Conference on Robotics & Automation*, 1997.

[13] P.K.Agarwal, P.Raghavan, and H.Tamaki. Motion planning for a steering constrained robot through moderate obstacles. In *Proceedings ACM Symposium on Computational Geometry*, 1995.

[14] S. Akella, W. H. Huang, K. M. Lynch, and M. T. Mason. Parts feeding on a conveyor with a one joint robot. *Algorithmica*, 26(3/4):313–344, March/April 2000.

[15] S.Akella and S.Hutchinson. Coordinating the motions of multiple robots with specified trajectories. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 624–631, 2002.

[16] R. Alami, J.-P. Laumond, and T. Siméon. Two manipulation planning algorithms. In J.-P. Laumond and M. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*. A.K. Peters, Wellesley, MA, 1997.

[17] R. Alami, T. Siméon, and J.-P. Laumond. A geometrical approach to planning manipulation tasks. In *Proceedings International Symposium on Robotics Research*, pages 113–119, 1989.

[18] G. Allgower and K. Georg. *Numerical Continuation Methods*. Springer-Verlag, Berlin, 1990.

[19] E. Alpaydin. *Machine Learning*. MIT Press, Cambridge, MA, 2004.

[20] H. Alt, R. Fleischer, M. Kaufmann, K. Mehlhorn, S. Näher, S. Schirra, and C. Uhrig. Approximate motion planning and the complexity of the boundary of the union of simple geometric figures. In *Proceedings*

ACM Symposium on Computational Geometry, pages 281–289, 1990.

[21] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 630–637, 1998.

[22] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, pages 155–168, 1998.

[23] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Transactions on Robotics & Automation*, 16(4):442– 447, Aug 2000.

[24] N. M. Amato, K. A. Dill, and G. Song. Using motion planning to map protein folding landscapes and analyze folding kinetics of known native structures. In *Proceedings 6th ACM International Conference on Computational Molecular Biology (RECOMB)*, pages 2–11, 2002.

[25] N. M. Amato and G. Song. Using motion planning to study protein folding pathways. *Journal of Computational Biology*, 9(2):149–168, 2002.

[26] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 113–120, 1996.

[27] F. Ancona and A. Bressan. Patchy vector fields and asymptotic stabilization. *ESAIM-Control, Optimisation and Calculus of Variations*, 4:445– 471, 1999.

[28] B. D. Anderson and J. B. Moore. *Optimal Control: Linear-Quadratic Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1990.

[29] J. Angeles. *Spatial Kinematic Chains. Analysis, Synthesis, and Optimization*. Springer-Verlag, Berlin, 1982.

[30] J. Angeles. *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*. Springer -Verlag, Berlin, 2003.

[31] D. A. Anisi, J. Hamberg, and X. Hu. Nearly time-optimal paths for a ground vehicle. *Journal of Control Theory and Applications*, November 2003.

[32] E. Anshelevich, S. Owens, F. Lamiraux, and L. E. Kavraki. Deformable volumes in path planning applications. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2290–2295, 2000.

[33] M. Apaydin, D. Brutlag, C. Guestrin, D. Hsu J.-C. Latombe, and C. Varm. Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion. *Journal of Computational Biology*, 10:257–281, 2003.

[34] M. D. Ardema and J. M. Skowronski. Dynamic game applied to coordination control of two arm robotic system. In R. P. Hämäläinen and H. K. Ehtamo, editors, *Differential Games-Developments in Modelling and Computation*, pages 118–130. Springer-Verlag, Berlin, 1991.

[35] O. Arikan and D. Forsyth. Interactive motion generation from examples. In *Proceedings ACM SIGGRAPH*, 2002.

[36] E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering traveling salesman problem. *Discrete Applied Mathematics*, 55:194–218, 1994.

[37] B. Armstrong, O. Khatib, and J. Burdick. The explicit dynamic model and inertial parameters of the Puma 560 arm. In *Proceedings IEEE International Conference on Systems, Man, & Cyberetics*, pages 510–518, 1986.

[38] M. A. Armstrong. *Basic Topology*. Springer-Verlag, New York, 1983.

[39] V. I. Arnold. *Mathematical Methods of Classical Mechanics, 2nd Ed.* Springer-Verlag, Berlin, 1989.

[40] D. S. Arnon. Geometric reasoning with logic and algebra. *Artificial Intelligence Journal*, 37(1-3):37– 60,

1988.

[41] B. Aronov, M. de Berg, A. F. van der Stappen, P. Svestka, and J. Vleugels. Motion planning for multiple robots. *Discrete and Computational Geometry*, 22:505–525, 1999.

[42] B. Aronov and M. Sharir. On translational motion planning of a convex polyhedron in 3-space. *SIAM Journal on Computing*, 26(6):1875–1803, December 1997.

[43] K. Arras, N. Tomatis, B. Jensen, and R. Siegwart. Multisensor on-the-fly localization: Precision and reliability for applications. *Robotics and Autonomous Systems*, 34(2-3):131–143, 2001.

[44] D. K. Arrowsmith and C. M. Place. *Dynamical Systems: Differential Equations, Maps, and Chaotic Behaviour*. Chapman & Hall/CRC, New York, 1992.

[45] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[46] J. Arvo. Fast random rotation matrices. In D. Kirk, editor, *Graphics Gems III*, pages 117–120. Academic, New York, 1992. 928 BIBLIOGRAPHY.

[47] S. Arya and D. M. Mount. Algorithms for fast vector quantization. In *IEEE Data Compression Conference*, pages 381–390, March 1993.

[48] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proceedings ACM-SIAM Symposium on Discrete Algorithms*, pages 271–280, 1993.

[49] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45:891–923, 1998.

[50] R. B. Ash. *Information Theory*. Dover, New York, 1990.

[51] K. J. Astrom and T. Hagglund. *PID Controllers: Theory, Design, and Tuning, 2nd Ed.* The Instrument, Systems, and Automation Society, Research Triangle Park, NC, 1995.

[52] K. E. Atkinson. *An Introduction to Numerical Analysis*. Wiley, New York, 1978.

[53] A. Atramentov and S. M. LaValle. Efficient nearest neighbor searching for motion planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 632– 637, 2002.

[54] J.-P. Aubin and A. Cellina. *Differential Inclusions*. Springer-Verlag, Berlin, 1984.

[55] F. Aurenhammer. Voronoi diagrams – A survey of a fundamental geometric structure. *ACM Computing Surveys*, 23:345– 405, 1991.

[56] F. Avnaim, J.-D. Boissonnat, and B. Faverjon. A practical exact planning algorithm for polygonal objects amidst polygonal obstacles. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1656–1660, 1988.

[57] J. Bañon. Implementation and extension of the ladder algorithm. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1548–1553, 1990.

[58] T. Ba¸sar. Game theory and H∞-optimal control: The continuous-time case. In R. P. Hämäläinen and H. K. Ehtamo, editors, *Differential Games – Developments in Modelling and Computation*, pages 171–186. Springer-Verlag, Berlin, 1991.

[59] T. Ba¸sar and P. R. Kumar. On worst case design strategies. *Computers and Mathematics with Applications*, 13(1-3):239–245, 1987.

[60] T. Ba¸sar and G. J. Olsder. *Dynamic Noncooperative Game Theory, 2nd Ed.* Academic, London, 1995.

[61] R. A. Baeza, J. C. Culberson, and G. J. E. Rawlins. Searching in the plane. *Information and Computation*, 106(2):234–252, 1993.

[62] B. Baginski. The Z3 method for fast path planning in dynamic environments. In *Proceedings IASTED Conference on Applications of Control and Robotics*, pages 47–52, 1996.

[63] B. Baginski. *Motion Planning for Manipulators with Many Degrees of Freedom – The BB-Method*. PhD

thesis, Technical University of Munich, 1998.

[64] A. Baker. *Matrix Groups*. Springer-Verlag, Berlin, 2002.

[65] D. J. Balkcom and M. T. Mason. Time optimal trajectories for bounded velocity differential drive vehicles. *International Journal of Robotics Research*, 21(3):199–217, 2002.

[66] D. J. Balkcom and M. T. Mason. Introducing robotic origami folding. In *Proceedings IEEE International Conference on Robotics & Automation*, 2004.

[67] J. E. Banta, Y. Zhien, X. Z. Wang, G. Zhang, M. T. Smith, and M. A. Abidi. A "best-next-view" algorithm for three-dimensional scene reconstruction using range images. In *Proceedings SPIE, vol.* 2588, pages 418–29, 1995.

[68] J. Barraquand and P. Ferbach. A penalty function method for constrained motion planning. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1235–1242, 1994.

[69] J. Barraquand and P. Ferbach. Motion planning with uncertainty: The information space approach. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1341–1348, 1995.

[70] J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, and P. Raghavan. A random sampling scheme for robot path planning. In G. Giralt and G. Hirzinger, editors, *Proceedings International Symposium on Robotics Research*, pages 249–264. Springer-Verlag, New York, 1996.

[71] J. Barraquand and J.-C. Latombe. A Monte-Carlo algorithm for path planning with many degrees of freedom. In *Proceedings IEEE International Conference onRobotics & Automation*, pages 1712–1717, 1990.

[72] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2328–2335, 1991.

[73] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10(6):628–649, December 1991.

[74] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots:Controllability and motion planning in the presence of obstacles. *Algorithmica*,10:121–155, 1993.

[75] A. G. Barto and R. S. Sutton. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[76] A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins. Learning and sequential decision making. In M. Gabriel and J.W. Moore, editors, *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, pages 539–602. MIT Press, Cambridge, MA, 1990.

[77] J. Basch, L. J. Guibas, D. Hsu, and A. T. Nguyen. Disconnection proofs for motion planning. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1765–1772, 2001.

[78] S. Basu, R. Pollack, and M. F. Roy. Computing roadmaps of semi-algebraic sets on a variety. *Journal of the American Society of Mathematics*, 3(1):55–82, 1999.

[79] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Springer-Verlag, Berlin, 2003.

[80] K. Basye and T. Dean. Map learning with indistinguishable locations. In M. Henrion, L. N. Kanal, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 5*, pages 331–340. Elsevier Science, New York, 1990.

[81] K. Basye, T. Dean, J. Kirman, and M. Lejter. A decision-theoretic approach to planning, perception, and control. *IEEE Expert*, 7(4):58–65, August 1992.

[82] T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53, 1763.

[83] C. Becker, H. Gonźalez-Baños, J.-C. Latombe, and C. Tomasi. An intelligent observer. In Preprints of

International Symposium on Experimental Robotics, pages 94–99, 1995.

[84] K. E. Bekris, B. Y. Chen, A. Ladd, E. Plaku, and L. E. Kavraki. Multiple query probabilistic roadmap planning using single query primitives. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[85] A. Bellaiche, F. Jean, and J. J. Risler. Geometry of nonholonomic systems. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 55–92.Springer-Verlag, Berlin, 1998.

[86] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[87] R. E. Bellman and S. E. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, NJ, 1962.

[88] I. Belousov, C. Esteves, J.-P. Laumond, and E. Ferre. Motion planning for large space manipulators with complicated dynamics. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[89] M. A. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *Proceedings Annual Symposium on Foundations of Computer Science*, 1998.

[90] A. Benveniste, M. Metivier, and P. Prourier. *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag, Berlin, 1990.

[91] J. O. Berger. *Statistical Decision Theory*. Springer-Verlag, Berlin, 1980.

[92] M. Bern. Triangulations and mesh generation. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, 2nd Ed., pages 563–582. Chapman and Hall/CRC Press, New York, 2004.

[93] J. Bernard, J. Shannan, and M. Vanderploeg. Vehicle rollover on smooth surfaces. In *Proceedings SAE Passenger Car Meeting and Exposition*, Dearborn, MI, 1989.

[94] D. P. Bertsekas. Convergence in discretization procedures in dynamic programming. *IEEE Transactions on Automatic Control*, 20(3):415– 419, June 1975.

[95] D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[96] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

[97] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I, 2nd Ed.* Athena Scientific, Belmont, MA, 2001.

[98] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II, 2nd Ed.* Athena Scientific, Belmont, MA, 2001.

[99] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[100] J. T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control*, and *Dynamics*, 21(2):193–207, March-April 1998.

[101] A. Beygelzimer, S. M. Kakade, and J. Langford. Cover trees for nearest neighbor. University of Pennsylvania, Available from http://www.cis.upenn.edu/~skakade/papers/ml/cover tree.pdf, 2005.

[102] A. Bhatia and E. Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In R. Alur and G. J. Pappas, editors, *Hybrid Systems: Computation and Control*, pages 67–78. Springer-Verlag, Berlin, 2004. Lecture Notes in Computer Science, 2993.

[103] S. Bhattacharya and S. K. Agrawal. Design, experiments and motion planning of a spherical rolling robot. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1207–1212, 2000.

[104] A. Bicchi, A. Marigo, and B. Piccoli. On the reachability of quantized control systems. *IEEE Transactions on Automatic Control*, 47(4):546–563, April 2002.

[105] A. Bicchi, D. Prattichizzo, and S. Sastry. Planning motions of rolling surfaces. In *Proceedings IEEE Conference Decision & Control*, 1995.

[106] Z. Bien and J. Lee. A minimum-time trajectory planning method for two robots. *IEEE Transactions on Robotics & Automation*, 8(3):414– 418, June 1992.

[107] D. Bienstock and P. Seymour. Monotonicity in graph searching. *Journal of Algorithms*, 12:239–245, 1991.

[108] J. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report ICSI-TR-97-021, International Computer Science Institute (ICSI), Berkeley, CA,1997.

[109] R. L. Bishop and S. I. Goldberg. *Tensor Analysis on Manifolds*. Dover, New York, 1980.

[110] H. S. Black. Stabilized feedback amplifiers. *Bell Systems Technical Journal*,13:1–18, 1934.

[111] D. Blackwell and M. A. Girshik. *Theory of Games and Statistical Decisions*. Dover, New York, 1979.

[112] S. Blind, C. McCullough, S. Akella, and J. Ponce. Manipulating parts with an array of pins: A method and a machine. *International Journal of Robotics Research*, 20(10):808–818, December 2001.

[113] A. Bloch. *Murphy's Law and Other Reasons Why Things Go Wrong*. Price Stern Sloan Adult, New York, 1977.

[114] A. M. Bloch. *Nonholonomic Mechanics and Control*. Springer-Verlag, Berlin, 2003.

[115] A. M. Bloch and P. E. Crouch. Nonholonomic control systems on Riemannian manifolds. *SIAM Journal on Control & Optimization*, 33:126 –148, 1995.

[116] V. D. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, September 2000.

[117] A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. In *Proceedings IEEE Symposium on Foundations of Computer Science*, 2003. 932 BIBLIOGRAPHY

[118] A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrains. In Proceedings *ACM Symposium on Computational Geometry*, pages 494 –504, 1991.

[119] A. L. Blum and M. L. Furst. Fast planing through planning graph analysis. In *Proceedings International Joint Conference on Artificial Intelligence*, pages 1636–1642, 1995.

[120] L. Blum, F. Cucker, and M. Schub abd S. Smale. *Complexity and Real Computation*. Springer-Verlag, Berlin, 1998.

[121] M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *Proceedings Annual Symposium on Foundations of Computer Science*, pages 132–142, 1978.

[122] J. E. Bobrow. Optimal robot path planning using the minimum-time criterion. *IEEE Transactions on Robotics & Automation*, 4(4):443– 450, August 1988.

[123] J. E. Bobrow, S. Dubowsky, and J. S. Gibson. Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotics Research*, 4(3):3–17, 1985.

[124] H. Bode. Feedback: The history of an idea. In R. Bellman and R. Kalaba, editors, *Selected Papers on Mathematical Trends in Control Theory*, pages 106–123. Dover, New York, 1969.

[125] R. Bohlin. Path planning in practice; lazy evaluation on a multi-resolution grid. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[126] R. Bohlin. *Robot Path Planning*. PhD thesis, Chalmers University, Gothenburg, Sweden, 2002.

[127] R. Bohlin and L. Kavraki. Path planning using Lazy PRM. In *Proceedings IEEE International Conference on Robotics & Automation*, 2000.

[128] K.-F. Böhringer, B. R. Donald, and N. C. MacDonald. Upper and lower bounds for programmable vector fields with applications to MEMS and vibratory plate parts feeders. In J.-P. Laumond and M. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*. A.K. Peters, Wellesley, MA, 1997.

[129] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, Cambridge, U.K.,

1998.

[130] J.-D. Boissonnat, A. Cérézo, and J. Leblond. Shortest paths of bounded curvature in the plane. *Journal of Intelligent and Robotic Systems*, 11:5–20, 1994.

[131] J.-D. Boissonnat and S. Lazard. A polynomial-time algorithm for computing a shortest path of bounded curvature amidst moderate obstacles. In *Proceedings ACM Symposium on Computational Geometry*, pages 242–251, 1996.

[132] V. G. Boltyanskii. Sufficient conditions for optimality and the justification of the dynamic programming method. *SIAM Journal on Control*, 4:326–361, 1966.

[133] G. Boo and K. Goldberg. Orienting polygonal parts without sensors: An implementation in Java. Alpha Lab, UC Berkeley. Available from http://www.ieor.berkeley.edu/~goldberg/feeder-S05/, 2005.

[134] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1018–1023, 1999.

[135] W. M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry*. Revised 2nd Ed. Academic, New York, 2003.

[136] J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques*. A.K. Peters, Wellesley, MA, 1996.

[137] A. V. Borisov and I .S. Mamaev. On the history of the development of nonholonomic dynamics. *Regular and Chaotic Dynamics*, 7(1):43– 47, 2002.

[138] P. Bose, A. Lubiv, and J. I. Munro. Efficient visibility queries in simple polygons. In *Proceedings Canadian Conference on Computational Geometry*, pages 23–28,1992.

[139] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31– 45, 1998.

[140] M. S. Branicky, M. M. Curtiss, J. Levine, and S. Morgan. RRTs for nonlinear,discrete, and hybrid planning and control. In *Proceedings IEEE Conference Decision & Control*, 2003.

[141] M. Bridson and A. Haefliger. *Metric Spaces of Non-Positive Curvature*. Springer-Verlag, Berlin, 1999.

[142] A. Briggs. An efficient algorithm for one-step compliant motion planning with uncertainty. In *Proceedings ACM Symposium on Computational Geometry*, 1989.

[143] A. J. Briggs and B. R. Donald. Robust geometric algorithms for sensor planning. In J.-P. Laumond and M. Overmars, editors, *Proceedings Workshop on Algorithmic Foundations of Robotics*. A.K. Peters, Wellesley, MA, 1996.

[144] R. W. Brockett. Control theory and singular Riemannian geometry. In P. A.Fuhrman, editor, *New Directions in Applied Mathematics*, pages 11–27. Springer-Verlag, Berlin, 1981.

[145] R. W. Brockett. Asymptotic stability and feedback stabilization. In R. W.Brockett, R. S. Millman, and H. J. Sussmann, editors, *Differential Geometric Control Theory*, pages 181–191. Birkhäuser, Boston, MA, 1983.

[146] R. A. Brooks and T. Lozano-Pérez. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, & Cybernetics*, SMC-15(2):224–233, 1985.

[147] R. C. Brost. Automatic grasp planning in the presence of uncertainty. *International Journal of Robotics Research*, 7(1):3–17, 1988.

[148] R. C. Brost. *Analysis and Planning of Planar Manipulation Tasks*. PhD thesis,Carnegie Mellon University, Pittsburgh, PA, 1991.

[149] R. C. Brost and A. D. Christiansen. Probabilistic analysis of manipulation tasks:A research agenda. In *Proceedings IEEE International Conference on Robotics & Automation*, volume 3, pages 549–556, 1993.

[150] R. C. Brost and A. D. Christiansen. Probabilistic analysis of manipulation tasks:A computational

framework. Technical Report SAND92-2033, Sandia National Laboratories, Albuquerque, NM, January 1994.

[151] R. C. Brost and A. D. Christiansen. Probabilistic analysis of manipulation tasks:A computational framework. *International Journal of Robotics Research*, 15(1):1–23, February 1996.

[152] J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation.In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[153] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Hemisphere Publishing Corp., New York, 1975.

[154] M. Buckland. *AI Techniques for Game Programming*. Premier Press, Portland, OR, 2002.

[155] S. J. Buckley. Fast motion planning for multiple moving robots. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 322–326, 1989.

[156] F. Bullo. Series expansions for the evolution of mechanical control systems. *SIAM Journal on Control & Optimization*, 40(1):166–190, 2001.

[157] F. Bullo. Series expansions for analytic systems linear in control. *Automatica*, 38(9):1425–1432, September 2002.

[158] F. Bullo and A. D. Lewis. *Geometric Control of Mechanical Systems*. Springer-Verlag, Berlin, 2004.

[159] F. Bullo and K. M. Lynch. Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems. *IEEE Transactions on Robotics & Automation*, 17(4):402– 412, 2001.

[160] J. W. Burdick. *Kinematic Analysis and Design of Redundant Manipulators*. PhD thesis, Stanford University, Stanford, CA, 1988.

[161] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proceedings AAAI National Conference on Artificial Intelligence*, pages 11–18, 1998.

[162] J. J. Burken, P. Lu, and Z.Wu. Reconfigurable flight control designs with application to the X-33 vehicle. Technical Report TM-1999-206582, NASA, Washington, DC, 1999.

[163] B. Burns and O. Brock. Sampling-based motion planning using predictive models.In *Proceedings IEEE International Conference on Robotics & Automation*, 2005.

[164] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research*,18(6):534–555, 1999.

[165] L. G. Bushnell, D. M. Tilbury, and S. S. Sastry. Steering three-input nonholonomic systems: the fire truck example. *International Journal of Robotics Research*, 14(4):366–381, 1995.

[166] Z. J. Butler, A. A. Rizzi, and R. L. Hollis. Contact sensor-based coverage of rectilinear environments. In *IEEE Symposium on Intelligent Control*, 1999.

[167] Z. J. Butler and D. Rus. Distributed motion planning for modular robots with unit-compressible modules. *International Journal of Robotics Research*, 22(9):699–716, 2003.

[168] S. Cambon, F. Gravot, and R. Alami. A robot task planner and merges symbolic and geometric reasoning. *In Proceedings European Conference* on Artificial Intelligence, 2004.

[169] S. Cameron. A comparison of two fast algorithms for computing the distance between convex polyhedra. *IEEE Transactions on Robotics & Automation*, 13(6):915–920, December 1997.

[170] F. Camilli and M. Falcone. Approximation of optimal control problems with state constraints: Estimates and applications. In B. S. Mordukhovich and H. J. Sussmann, editors, *Nonsmooth Analysis and Geometric Methods in Deterministic Optimal Control*, pages 23–57. Springer-Verlag, Berlin, 1996. Mathematics and its Applications, Vol. 78.

[171] J. Canny. Constructing roadmaps of semi-algebraic sets I. *Artificial Intelligence Journal*, 37:203–222, 1988.

[172] J. Canny. Computing roadmaps of general semi-algebraic sets. *The Computer Journal*, 36(5):504–514,

1993.

[173] J. Canny, A. Rege, and J. Reif. An exact algorithm for kinodynamic planning in the plane. *Discrete and Computational Geometry*, 6:461– 484, 1991.

[174] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 49–60, 1987.

[175] J. F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[176] J. F. Canny. On computability of fine motion plans. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 177–182, 1989.

[177] J. F. Canny and K. Y. Goldberg. "RISC" industrial robots: Recent results and current trends. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1951–1958, 1994.

[178] J. F. Canny and M. Lin. An opportunistic global path planner. *Algorithmica*,10:102–120, 1993.

[179] S. Carpin and E. Pagello. On parallel RRTs for multi-robot systems. In *Proceedings 8th Conference of the Italian Association for Artificial Intelligence*, pages 834–841, 2002.

[180] S. Carpin and G. Pillonetto. Merging the adaptive random walks planner with the randomized potential field planner. In *Proceedings IEEE International Workshop on Robot Motion and Control*, pages 151–156, 2005.

[181] S. Carpin and G. Pillonetto. Robot motion planning using adaptive random walks. *IEEE Transactions on Robotics & Automation*, 21(1):129–136, 2005.

[182] A. Casal. *Reconfiguration Planning for Modular Self-Reconfigurable Robots.* PhD thesis, Stanford University, Stanford, CA, 2002.

[183] S. Caselli and M. Reggiani. ERPP: An experience-based randomized path planner. In *Proceedings IEEE International Conference on Robotics & Automation*, 2000.

[184] J. Castellanos, J. Montiel, J. Neira, and J. Tardós. The SPmap: A probabilistic framework for simultaneous localization and mapping. *IEEE Transactions on Robotics & Automation*, 15(5):948–953, 1999.

[185] D. Challou, D. Boley, M. Gini, and V. Kumar. A parallel formulation of informed randomized search for robot motion planning problems. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 709–714, 1995.

[186] D. D. Champeaux. Bidirectional heuristic search again. *Journal of the ACM*,30(1):22–32, January 1983.

[187] D. D. Champeaux and L. Sint. An improved bidirectional heuristic search algorithm. *Journal of the ACM*, 24(2):177–191, April 1977.

[188] H. Chang and T. Y. Li. Assembly maintainability study with motion planning.In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1012–1019, 1995.

[189] S. Charentus. *Modeling and Control of a Robot Manipulator Composed of Several Stewart Platforms.* PhD thesis, Université Paul Sabatier, Toulouse, France, 1990. In French.

[190] S. Chawla. *Graph Algorithms for Planning and Partitioning.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, June 2005.

[191] B. Chazelle. Approximation and decomposition of shapes. In J. T. Schwartz and C. K. Yap, editors, *Algorithmic and Geometric Aspects of Robotics*, pages 145–185. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[192] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6(5):485–524, 1991.

[193] B. Chazelle. *The Discrepancy Method.* Cambridge University Press, Cambridge,U.K., 2000.

[194] C.-T. Chen. *Linear System Theory and Design.* Holt, Rinehart, and Winston, New York, 1984.

[195] P. C. Chen and Y. K. Hwang. SANDROS: A motion planner with performance proportional to task

difficulty. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2346–2353, 1992.

[196] P. C. Chen and Y. K. Hwang. SANDROS: A dynamic search graph algorithm for motion planning. *IEEE Transactions on Robotics & Automation*, 14(3):390– 403, 1998.

[197] Y.-B. Chen and D. J. Ierardi. The complexity of oblivious plans for orienting and distinguishing polygonal parts. *Algorithmica*, 14:367–397, 1995.

[198] P. Cheng. *Sampling-Based Motion Planning with Differential Constraints*. PhD thesis, University of Illinois, Urbana, IL, August 2005.

[199] P. Cheng, E. Frazzoli, and S. M. LaValle. Exploiting group symmetries to improve precision in kinodynamic and nonholonomic planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[200] P. Cheng, E. Frazzoli, and S. M. LaValle. Improving the performance of samplingbased planners by using a symmetry-exploiting gap reduction algorithm. In *Proceedings IEEE International Conference on Robotics and Automation*, 2004.

[201] P. Cheng and S. M. LaValle. Reducing metric sensitivity in randomized trajectory design. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 43– 48, 2001.

[202] P. Cheng and S. M. LaValle. Resolution complete rapidly-exploring random trees. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 267–272, 2002.

[203] P. Cheng, Z. Shen, and S. M. LaValle. Using randomization to find and optimize feasible trajectories for nonlinear systems. In *Proceedings Annual Allerton Conference on Communications, Control, Computing*, pages 926–935, 2000.

[204] P. Cheng, Z. Shen, and S. M. LaValle. RRT-based trajectory design for autonomous automobiles and spacecraft. *Archives of Control Sciences*, 11(3-4):167–194, 2001.

[205] M. Cherif. Kinodynamic motion planning for all-terrain wheeled vehicles. In *Proceedings IEEE International Conference on Robotics & Automation*, 1999.

[206] F. L. Chernousko, N. N. Bolotnik, and V. G. Gradetsky. *Manipulation Robots*.CRC Press, Boca Raton, FL, 1994.

[207] L. P. Chew and K. Kedem. A convex polygon among polygonal obstacles: Placement and high-clearance motion. *Computational Geometry: Theory and Applications*, 3:59–89, 1993.

[208] D. Chibisov, E. W. Mayr, and S. Pankratov. Spatial planning and geometric optimization: Combining configuration space and energy methods. In H. Hong and D. Wang, editors, *Automated Deduction in Geometry -ADG 2004*. Springer-Verlag, Berlin, 2006.

[209] S. Chien, R. Sherwood, D. Tran, B. Cichy, D. Mandl, S. Frye, B. Trout, S. Shulman, and D. Boyer. Using autonomy flight software to improve science return on Earth Observing One. *Journal of Aerospace Computing, Information, and Communication*, 2:196–216, April 2005.

[210] W.-P. Chin and S. Ntafos. Optimum watchman routes. *Information Processing Letters*, 28:39–44, 1988.

[211] G. Chirikjian, A. Pamecha, and I. Ebert-Uphoff. Evaluating efficiency of selfreconfiguration in a class of modular robots. *Journal of Robotic Systems*, 13(5):717–338, 1996.

[212] G. S. Chirikjian and A. B. Kyatkin. *Engineering Applications of Noncommutative Harmonic Analysis*. CRC Press, Boca Raton, FL, 2001.

[213] H. Chitsaz, S. M. LaValle, D. J. Balkcom, and M. T. Mason. Minimum wheelrotation paths for differential-drive mobile robots. In *Proceedings IEEE International Conference on Robotics and Automation*, 2006.

[214] H. Chitsaz, J. M. O'Kane, and S. M. LaValle. Pareto-optimal coordination of two translating polygonal robots on an acyclic roadmap. In *Proceedings IEEE International Conference on Robotics and*

Automation, 2004.

[215] S. Chitta, P. Cheng, E. Frazzoli, and V. Kumar. RoboTrikke: A novel undulatory locomotion system. In *Proceedings IEEE International Conference on Robotics & Automation*, 2005.

[216] S. Chitta and V. Kumar. Dynamics and generation of gaits for a planar rollerblader. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[217] J. Choi, J. Sellen, and C. K. Yap. Precision-sensitive Euclidean shortest path in 3-space. In *Proceedings ACM Symposium on Computational Geometry*, pages 350–359, 1995.

[218] H. Choset. Coverage of known spaces: The boustrophedon cellular decomposition. *Autonomous Robots*, 9:247–253, 2000.

[219] H. Choset. Coverage for robotics – A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113–126, 2001.

[220] H. Choset and J. Burdick. Sensor based motion planning: Incremental construction of the hierarchical generalized Voronoi graph. *International Journal of Robotics Research*, 19(2):126–148, 2000.

[221] H. Choset and J. Burdick. Sensor based motion planning: The hierarchical generalized Voronoi graph. *International Journal of Robotics Research*, 19(2):96–125, 2000.

[222] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, 2005.

[223] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (T-SLAM). *IEEE Transactions on Robotics & Automation*, 17(2):125–137, 2001.

[224] H. Choset and P. Pignon. Cover path planning: The boustrophedron decomposition. In *Proceedings International Conference on Field and Service Robotics*, Canberra, Australia, December 1997.

[225] P. Choudhury and K. Lynch. Rolling manipulation with a single control. In *Proceedings Conference on Control Applications*, September 2002.

[226] P. Choudhury and K. Lynch. Trajectory planning for second-order underactuated mechanical systems in presence of obstacles. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, 2002.

[227] K.-C. Chu. Team decision theory and information structures in optimal control problems – Part II. *IEEE Transactions on Automatic Control*, 17(1):22–28, February 1972.

[228] C. K. Chui and G. Chen. *Kalman Filtering*. Springer-Verlag, Berlin, 1991.

[229] M. Chyba, H. Sussmann, H. Maurer, and G. Vossen. Underwater vehicles: The minimum time problem. In *Proceedings IEEE Conference Decision & Control*, The Bahamas, December 2004.

[230] C. M. Clark, S. M. Rock, and J.-C. Latombe. Motion planning for multiple mobile robots using dynamic networks. In *Proceedings IEEE International Conference on Robotics & Automation*, 2003.

[231] D. E. Clark, G. Jones, P. Willett P. W. Kenny, and R. C. Glen. Pharmacophoric pattern matching in files of three-dimensional chemical structures: Comparison of conformational searching algorithms for flexible searching. *Journal Chemical Information and Computational Sciences*, 34:197–206, 1994.

[232] K. L. Clarkson. Nearest neighbor searching in metric spaces: Experimental results for sb(s). Bell Labs. Available from http://cm.belllabs. com/who/clarkson/Msb/readme.html, 2003.

[233] F. S. Cohen and D. B. Cooper. Simple parallel hierarchical and relaxation algorithms for segmenting noncausal Markovian random fields. *IEEE Transactions Pattern Analysis Machine Intelligence*, 9(2):195–219, March 1987.

[234] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proceedings Second GI Conference on Automata Theory and Formal Languages*, pages 134–183, Berlin, 1975. Springer-Verlag. Lecture Notes in Computer Science, 33.

[235] G. E. Collins. Quantifier elimination by cylindrical algebraic decomposition-twenty years of progress. In B. F. Caviness and J. R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic*

Decomposition, pages 8–23. Springer-Verlag, Berlin, 1998.

[236] L. Conlon. *Differentiable Manifolds, 2nd Ed.* Birkhäuser, Boston, MA, 2001.

[237] D. C. Conner, A. A. Rizzi, and H. Choset. Composition of local potential functions for global robot control and navigation. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3546–3551, 2003.

[238] C. Connolly and R. Grupen. The application of harmonic potential functions to robotics. *Journal of Robotic Systems*, 10(7):931–946, 1993.

[239] C. Connolly, R. Grupen, and K. Souccar. A Hamiltonian framework for kinodynamic planning. In *Proceedings IEEE International Conference on Robotics & Automation*, 1995.

[240] C. I. Connolly. The determination of next best views. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 432– 435, 1985.

[241] C. I. Connolly. Applications of harmonic functions to robotics. In *IEEE Symposium on Intelligent Control*, pages 498–502, 1992.

[242] C. I. Connolly, J. B. Burns, and R.Weiss. Path planning using Laplace's equation. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2102–2106, May 1990.

[243] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices, and Groups*. Springer-Verlag, Berlin, 1999.

[244] H. W. Corley. Some multiple objective dynamic programs. *IEEE Transactions on Automatic Control*, 30(12):1221–1222, December 1985.

[245] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (2nd Ed.)*. MIT Press, Cambridge, MA, 2001.

[246] J. Cortés. *Motion Planning Algorithms for General Closed-Chain Mechanisms*. PhD thesis, Institut National Polytechnique de Toulouse, Toulouse, France, 2003.

[247] J. Cortés, T. Siméon M. Remaud-Siméon, and V. Tran. Geometric algorithms for the conformational analysis of long protein loops. *Journal of Computational Chemistry*, 25:956–967, 2004.

[248] J. Cortés, T. Siméon, and J.-P. Laumond. A random loop generator for planning the motions of closed kinematic chains using PRM methods. In *Proceedings IEEE International Conference on Robotics & Automation*, 2002.

[249] M. G. Coutinho. *Dynamic Simulations of Multibody Systems*. Springer-Verlag, Berlin, 2001.

[250] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, New York, 1991.

[251] N. Cowan. Composing navigation functions on Cartesian products of manifolds with boundary. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, Zeist, The Netherlands, July 2004.

[252] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer-Verlag, Berlin, 1992.

[253] H. S. M. Coxeter. *Regular Polytopes*. Dover, New York, 1973.

[254] J. J. Craig. *Introduction to Robotics*. Addison-Wesley, Reading, MA, 1989.

[255] M. G. Crandall and P.-L. Lions. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 277(1):1–42, 1983.

[256] D. Crass, I. Suzuki, and M. Yamashita. Searching for a mobile intruder in a corridor-The open edge variant of the polygon search problem. *International Journal Computational Geometry & Applications*, 5(4):397– 412, 1995.

[257] J. C. Culberson. Sokoban is PSPACE-complete. In *Proceedings International Conference on Fun with Algorithms (FUN 98)*, pages 65–76, Waterloo, Ontario, Canada, June 1998. Carleton Scientific.

[258] M. R. Cutkosky. *Robotic Grasping and Fine Manipulation*. Kluwer, Boston, MA, 1985.

[259] L. K. Dale and N. M. Amato. Probabilistic roadmap methods are embarrassingly parallel. In *Proceedings IEEE International Conference on Robotics & Automation*, 1999.

[260] F. Dallaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In

Proceedings IEEE International Conference on Robotics & Automation, 1999.

[261] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

[262] A. Datta, C. A. Hipke, and S. Schuierer. Competitive searching in polygons-beyond generalized streets. In J. Staples, P. Eades, N. Katoh, and A. Moffat, editors, *Algorithms and Computation, ISAAC '95*, pages 32– 41. Springer-Verlag, Berlin, 1995.

[263] R. S. Datta. Using computer algebra to compute Nash equilibria. In *Proceedings International Symposium on Symbolic and Algebraic Computation*, 2003.

[264] J. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5:29–35, 1988.

[265] S. Davies. Multidimensional triangulation and interpolation for reinforcement learning. In *Proceedings Neural Information Processing Systems*, 1996.

[266] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications, 2nd Ed.* Springer-Verlag, Berlin, 2000.

[267] M. J. de Smith. *Distance and Path: The Development, Interpretation and Application of Distance Measurement in Mapping and Modelling*. PhD thesis, University College, University of London, London, 2003.

[268] T. Dean and S. Kambhampati. Planning and scheduling. In A. B. Tucker, editor, *The CRC Handbook of Computer Science and Engineering*, pages 614–636. CRC Press, Boca Raton, FL, 1997.

[269] T. L. Dean and M. P. Wellman. *Planning and Control*. Morgan Kaufman, San Francisco, CA, 1991.

[270] M. H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, New York, 1970.

[271] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, Ser. B., 39:1–38, 1977.

[272] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment I: The rectilinear case. Available from http://www.cs.berkeley.edu/~christos/, 1997.

[273] P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[274] R. Dial. *Algorithm 360: Shortest path forest with topological ordering*. Communications of the ACM, 12:632–633, 1969.

[275] E. W. Dijkstra. *A note on two problems in connexion with graphs*. Numerische Mathematik, 1:269–271, 1959.

[276] D. V. Dimarogonas, M. M. Zavlanos, S. G. Loizou, and K. J. Kyriakopoulos. Decentralized motion control of multiple holonomic agents under input constraints. In *Proceedings IEEE Conference Decision & Control*, 2003.

[277] G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Transactions on Robotics & Automation*, 17(3):229–241, 2001.

[278] A. W. Divelbiss and J. T. Wen. Nonholonomic path planning with inequality constraints. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 52–57, 1994.

[279] A.W. Divelbiss and J. T.Wen. A path-space approach to nonholonomic planning in the presence of obstacles. *IEEE Transactions on Robotics & Automation*, 13(3):443– 451, 1997.

[280] W. E. Dixon, A. Behal, D. M. Dawson, and S. Nagarkatti. *Nonlinear Control of Engineering Systems*: A *Lyapunov-Based Approach*. Birkhäuser, Boston, MA, 2003.

[281] M. P. do Carmo. *Riemannian Geometry*. Birkhäuser, Boston, MA, 1992.

[282] B. R. Donald. Motion planning with six degrees of freedom. Technical Report AI-TR-791, Artificial Intelligence Lab., Massachusetts Institute of Technology, Cambridge, MA, 1984.

[283] B. R. Donald. *Error Detection and Recovery for Robot Motion Planning with Uncertainty*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1987.

[284] B. R. Donald. A search algorithm for motion planning with six degrees of freedom. *Artificial Intelligence Journal*, 31:295–353, 1987.

[285] B. R. Donald. The complexity of planar compliant motion planning under uncertainty. In *Proceedings ACM Symposium on Computational Geometry*, pages 309–318, 1988.

[286] B. R. Donald. A geometric approach to error detection and recovery for robot motion planning with uncertainty. *Artificial Intelligence Journal*, 37:223–271, 1988.

[287] B. R. Donald. Planning multi-step error detection and recovery strategies. *International Journal of Robotics Research*, 9(1):3–60, 1990.

[288] B. R. Donald. On information invariants in robotics. *Artificial Intelligence Journal*, 72:217–304, 1995.

[289] B. R. Donald and J. Jennings. Sensor interpretation and task-directed planning using perceptual equivalence classes. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 190–197, 1991.

[290] B. R. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for Cartesian robots and open chain manipulators. *Algorithmica*, 14(6):480–530, 1995.

[291] B. R. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning: Robots with decoupled dynamics bounds. *Algorithmica*, 14(6):443– 479, 1995.

[292] B. R. Donald, P. G. Xavier, J. Canny, and J. Reif. Kinodynamic planning. *Journal of the ACM*, 40: 1048– 66, November 1993.

[293] S. K. Donaldson. Self-dual connections and the topology of smooth 4-manifold. *Bulletin of the American Mathematical Society*, 8:81–83, 1983.

[294] A. L. Dontchev. Discrete approximations in optimal control. In B. S. Mordukhovich and H. J. Sussmann, editors, *Nonsmooth Analysis and Geometric Methods in Deterministic Optimal Control*, pages 59–80. Springer-Verlag, Berlin, 1996. Mathematics and Its Applications, Vol. 78.

[295] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, Berlin, 2001.

[296] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[297] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, 2nd Ed.* Wiley, New York, 2000.

[298] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, Cambridge, U.K., 2000.

[299] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Map validation and self-location in a graph-like world. In *Proceedings AAAI National Conference on Artificial Intelligence, pages* 1648–1653, 1993.

[300] G. Dudek, K. Romanik, and S. Whitesides. Global localization: Localizing a robot with minimal travel. *SIAM Journal on Computing*, 27(2):583–604, April 1998.

[301] I. Duleba. *Algorithms of Motion Planning for Nonholonomic Robots*. Technical University of Wroclaw, Wroclaw, Poland, 1998.

[302] G. E. Dullerud and R. D'Andrea. Distributed control of heterogeneous systems. *IEEE Transactions on Automatic Control*, 49(12):2113–2128, 2004.

[303] G. E. Dullerud and F. Paganini. *A Course in Robust Control Theory*. Springer-Verlag, Berlin, 2000.

[304] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, Berlin, 1987.

[305] C. Edwards and S. K. Spurgeon. *Sliding Mode Control: Theory and Applications*. CRC Press, Ann Arbor, MI, 1998.

[306] A. Efrat, L. J. Guibas, S. Har-Peled, D. C. Lin, J. S. B. Mitchell, and T. M. Murali. Sweeping simple

polygons with a chain of guards. In *Proceedings ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[307] M. Egerstedt and X. Hu. Formation constrained multi-agent control. *IEEE Transactions on Robotics & Automation*, 17(6):947–951, December 2001.

[308] S. Ehmann and M. C. Lin. Accurate and fast proximity queries between polyhedra using convex surface *decomposition. In Proceedings Eurographics*, 2001.

[309] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer*, 22(6):46–57, June 1989.

[310] G. Ellis. *Observers in Control Systems*. Elsevier, New York, 2002.

[311] I. Z. Emiris and B. Mourrain. Computer algebra methods for studying and computing molecular conformations. Technical report, INRIA, Sophia-Antipolis, France, 1997.

[312] A. G. Erdman, G. N. Sandor, and S. Kota. *Mechanism Design: Analysis and Synthesis, 4th Ed., Vol. 1*. Prentice Hall, Englewood Cliffs, NJ, 2001.

[313] M. A. Erdmann. *On motion planning with uncertainty*. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, August 1984.

[314] M. A. Erdmann. Using backprojections for fine motion planning with uncertainty. *International Journal of Robotics Research*, 5(1):19– 45, 1986.

[315] M. A. Erdmann. *On Probabilistic Strategies for Robot Tasks*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1989.

[316] M. A. Erdmann. Randomization in robot tasks. *International Journal of Robotics Research*, 11(5):399–436, October 1992.

[317] M. A. Erdmann. Randomization for robot tasks: Using dynamic programming in the space of knowledge states. *Algorithmica*, 10:248–291, 1993.

[318] M. A. Erdmann. On a representation of friction in configuration space. *International Journal of Robotics Research*, 13(3):240–271, 1994.

[319] M. A. Erdmann. Understanding action and sensing by designing action-based sensors. *International Journal of Robotics Research*, 14(5):483–509, 1995.

[320] M. A. Erdmann. An exploration of nonprehensile two-palm manipulation using two zebra robots. In J.-P. Laumond and M. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 239–254. A.K. Peters, Wellesley, MA, 1997.

[321] M. A. Erdmann and T. Lozano-Pérez. On multiple moving objects. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1419–1424, 1986.

[322] M. A. Erdmann and T. Lozano-Pérez. On multiple moving objects. *Algorithmica*, 2:477–521, 1987.

[323] M. A. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE Transactions on Robotics & Automation*, 4(4):369–379, August 1988.

[324] M. A. Erdmann, M. T. Mason, and G. Vaněček. Mechanical parts orienting: The case of a polyhedron on a table. *Algorithmica*, 10:206–247, 1993.

[325] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics & Automation*, 8(3):313–326, June 1992.

[326] J. Esposito, J. W. Kim, and V. Kumar. Adaptive RRTs for validating hybrid robotic control systems. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, Zeist, The Netherlands, July 2004.

[327] M. Farber. Topological complexity of motion planning. *Discrete and Computational Geometry*, 29:211–221, 2003.

[328] M. Farber, S. Tabachnkov, and S. Yuzvinsky. Topological robotics: Motion planning in projective spaces. *International Mathematical Research Notices*, 34:1853–1870, 2003.

[329] M. Farber and S. Yuzvinsky. Topological robotics: Subspace arrangements and collision free motion

planning. Technical Report math.AT/0210115, arXiv (online), 2004.

[330] B. Faverjon. Obstacle avoidance using an octree in the configuration space of a manipulator. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 504–512, 1984.

[331] B. Faverjon. Hierarchical object models for efficient anti-collision algorithms. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 333–340, 1989.

[332] B. Faverjon and P. Tournassoud. A local based method for path planning of manipulators with a high number of degrees of freedom. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1152–1159, 1987.

[333] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer, Boston, MA, 1987.

[334] S. P. Fekete, R. Klein, and A. Nüchter. Online searching with an autonomous robot. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, Zeist, The Netherlands, July 2004.

[335] P. Ferbach. A method of progressive constraints for nonholonomic motion planning. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2949–2955, 1996.

[336] C. Fernandes, L. Gurvits, and Z. X. Li. A variational approach to optimal nonholonomic motion planning. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 680–685, 1991.

[337] C. Fernandes, L. Gurvits, and Z. X. Li. Near-optimal nonholonomic motion planning for a system of coupled rigid bodies. *IEEE Transactions on Automatic Control*, 30(3):450–463, March 1994.

[338] R. Fierro, A. Das, V. Kumar, and J. P. Ostrowski. Hybrid control of formations of robots. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 157–162, 2001.

[339] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving. *Artificial Intelligence Journal*, 2:189–208, 1971.

[340] A. F. Filippov. Differential equations with discontinuous right-hand sides. *American Mathematical Society Translations*, Ser. 2, 64:199–231, 1964.

[341] P. W. Finn, D. Halperin, L. E. Kavraki, J.-C. Latombe, R. Motwani, C. Shelton, and S. Venkatasubramanian. Geometric manipulation of flexible ligands. In M. C. Lin and D. Manocha, editors, *Applied Computational Geometry*, pages 67–78. Springer-Verlag, Berlin, 1996. Lecture Notes in Computer Science, 1148.

[342] R. J. Firby. An investigation into reactive planning in complex domains. In *Proceedings AAAI National Conference on Artificial Intelligence,* 1987.

[343] G. F. Fishman. *Monte Carlo: Concepts, Algorithms, and Applications*. Springer-Verlag, Berlin, 1996.

[344] R. Fleischer, K. Romanik, S. Schuierer, and G. Trippen. Optimal localization in trees. *Information and Computation*, 171(2):224 –247, 2002.

[345] S. Fleury, P. Souères, J.-P. Laumond, and R. Chatila. Primitives for smoothing mobile robot trajectories. *IEEE Transactions on Robotics & Automation*, 11(3):441– 448, 1995.

[346] M. Fliess, J. Lévine, P. Martin, and P. Rouchon. Flatness and defect of non-linear systems: Introductory theory and examples. *International Journal of Control*, 61(6):1327–1361, 1995.

[347] H. Flordal, M. Fabian, and K. Akesson. Automatic implementation and verification of coordinating PLC-code for robotcells. In *Proceedings IFAC Symposium of Information Control Problems in Manufacturing*, 2004.

[348] G. B. Folland. *Real Analysis: Modern Techniques and Their Applications*. Wiley, New York, 1984.

[349] S. Fortune and G. Wilfong. Planning constrained motion. In *Proceedings ACM Symposium on Theory of Computing*, pages 445–459, 1988.

[350] M. Foskey, M. Garber, M. Lin, and D. Manocha. A Voronoi-based hybrid motion planner. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[351] A. Fox and S. Hutchinson. Exploiting visual constraints in the synthesis of uncertainty-tolerant motion

plans. *IEEE Transactions on Robotics & Automation*, 1(11):56–71, February 1995.

[352] D. Fox, S. Thrun, W. Burgard, and F. Dallaert. Particle filters for mobile robot localization. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 401–428. Springer-Verlag, Berlin, 2001.

[353] D. Hähnel D. Fox, W. Burgard, and S. Thrun. A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[354] J. Fraden. *Handbook of Modern Sensors: Physics, Designs, and Applications*. Springer-Verlag, Berlin, 2003.

[355] T. Fraichard. Dynamic trajectory planning with dynamic constraints: A 'statetime space' approach. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1393–1400, 1993.

[356] T. Fraichard and J.-M. Ahuactzin. Smooth path planning for cars. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3722–3727, 2001.

[357] T. Fraichard and H. Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics*, pages 1001–1024, 2004.

[358] T. Fraichard and C. Laugier. Kinodynamic planning in a structured and timevarying 2D workspace. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2: 1500–1505, 1992.

[359] T. Fraichard and A. Scheuer. Car-like robots and moving obstacles. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 64 – 69, 1994.

[360] T. Fraichard and A. Scheuer. From Reeds and Shepp's to continuous-curvature paths. *IEEE Transactions on Robotics*, 20(6):1025–1035, December 2004.

[361] T. Frankel. *The Geometry of Physics*. Cambridge University Press, Cambridge, U.K., 2004.

[362] E. Frazzoli. *Robust Hybrid Control of Autonomous Vehicle Motion Planning*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2001.

[363] E. Frazzoli and F. Bullo. Decentralized algorithms for vehicle routing in a stochastic time-varying environment. In *Proceedings IEEE Conference Decision & Control*, pages 3357–3363, 2004.

[364] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance and Control*, 25(1):116–129, 2002.

[365] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, December 2005.

[366] E. Freund and H. Hoyer. Path finding in multi robot systems including obstacle avoidance. *International Journal of Robotics Research*, 7(1):42–70, February 1988.

[367] J. H. Friedman, J. L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Soft-ware*, 3(3):209–226, September 1977.

[368] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill, New York, 1987.

[369] K. Fujimura and H. Samet. A hierarchical strategy for path planning among moving obstacles. Technical Report CAR-TR-237, Center for Automation Research, University of Maryland, November 1986.

[370] K. Fujimura and H. Samet. Planning a time-minimal motion among moving obstacles. *Algorithmica*, 10:41–63, 1993.

[371] T. Fukuda, S. Ito, N. Oota, F. Arai, Y. Abe, K. Tanake, and Y. Tanaka. Navigation system based on ceiling landmark recognition for autonomous mobile robot. In *Proceedings International Conference on Industrial Electronics*, Control, and Instrumentation, pages 1466–1471, 1993.

[372] A. T. Fuller. Relay control systems optimized for various performance criteria. In *Automatic and Remote*

Control (Proceedings First World Congress IFAC, Moscow, 1960), pages 510–519. Butterworths, London, 1961.

[373] J. Funge. *Artificial Intelligence for Computer Games*. A. K. Peters, Wellesley, MA, 2004.

[374] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. Technical report, Dept. of Mechanical Engineering, Technion, Israel Institute of Technology, December 1999.

[375] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1927–1933, 2001.

[376] Y. Gabriely and E. Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Computational Geometry: Theory and Applications*, 24(3):197–224, April 2003.

[377] Y. Gabriely and E. Rimon. Competitive complexity of mobile robot on line motion planning problems. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, pages 249–264, 2004.

[378] J. Gallier. *Curves and Surfaces in Geometric Modeling*. Morgan Kaufmann, San Francisco, CA, 2000.

[379] D. Geman and S. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions Pattern Analysis Machine Intelligence*, 6(6):721–741, November 1984.

[380] M. R. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Francisco, CA, 1987.

[381] R. Geraerts and M. Overmars. Sampling techniques for probabilistic roadmap planners. In *Proceedings International Conference on Intelligent Autonomous Systems*, 2004.

[382] R. Geraerts and M. H. Overmars. A comparative study of probabilistic roadmap planners. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, December 2002.

[383] B. Gerkey, S. Thrun, and G. Gordon. Clear the building: Pursuit-evasion with teams of robots. In *Proceedings AAAI National Conference on Artificial Intelligence*, 2004.

[384] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufman, San Francisco, CA, 2004.

[385] A. K. Ghose, M. E. Logan, A. M. Treasurywala, H.Wang, R. C.Wahl, B. E. Tomczuk, M. R. Gowravaram, E. P. Jaeger, and J. J. Wendoloski. Determination of pharmacophoric geometry for collagenase inhibitors using a novel computational method and its verification using molecular dynamics, NMR, and X-ray crystallography. *Journal of the American Chemical Society*, 117:4671– 4682, 1995.

[386] S. K. Ghosh and D. M. Mount. An output sensitive algorithm for computing visibility graphs. *SIAM Journal on Computing*, 20:888–910, 1991.

[387] R. Ghrist. Shape complexes for metamorphic robot systems. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, December 2002.

[388] R. Ghrist, J. M. O'Kane, and S. M. LaValle. Computing Pareto Optimal Coordinations on Roadmaps. *The International Journal of Robotics Research*, 24(11):997–1010, 2005.

[389] E. G. Gilbert and D. W. Johnson. Distance functions and their application to robot path planning in the presence of obstacles. *IEEE Transactions on Robotics & Automation*, 1(1):21–30, March 1985.

[390] E. G. Gilbert, D. W. Johnson, and S. S. Keerth. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics & Automation*, RA-4(2):193–203, Apr 1988.

[391] T. N. Gillespie. Fundamentals of Vehicle Dynamics. *Society of Automotive Engineers*, Warrendale, PA, 1992.

[392] B. Glavina. Solving findpath by combination of goal-directed and randomized search. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1718–1723, May 1990.

[393] B. Glavina. *Planning collision free motions for manipulators through a combination of goal oriented*

search and the creation of intermediate random subgoals. PhD thesis, Technical University of Munich, 1991. In German.

[394] P. J. Gmytrasiewicz, E. H. Durfee, and D. K. Wehe. A decision-theoretic approach to coordinating multi-agent interactions. In *Proceedings International Joint Conference on Artificial Intelligence*, pages 62–68, 1991.

[395] J. Go, T. Vu, and J. J. Kuffner. Autonomous behaviors for interactive vehicle animations. In *Proceedings SIGGRAPH Symposium on Computer Animation*, 2004.

[396] K. Y. Goldberg. *Stochastic Plans for Robotic Manipulation*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, August 1990.

[397] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

[398] K. Y. Goldberg and M. T. Mason. Bayesian grasping. In *Proceedings IEEE International Conference on Robotics & Automation*, 1990.

[399] H. Goldstein. *Classical Mechanics*. Addison-Wesley, Reading, MA, 1980.

[400] M. Goldwasser and R. Motwani. Intractability of assembly sequencing: Unit disks in the plane. In F. Dehne, A. Rau-Chaplin, J.-R. Sack, and R. Tamassia, editors, *WADS '97 Algorithms and Data Structures*, pages 307–320. Springer-Verlag, Berlin, 1997. Lecture Notes in Computer Science, 1272.

[401] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed)*. Johns Hopkins University Press, Baltimore, MD, 1996.

[402] R. Gonzalez and E. Rofman. On deterministic control problems: An approximation procedure for the optimal cost, parts Ⅰ, Ⅱ. *SIAM Journal on Control & Optimization*, 23:242–285, 1985.

[403] H. H. Gonźalez-Baños, L. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, R. Motwani, and C. Tomasi. Motion planning with visibility constraints: Building autonomous observers. In Y. Shirai and S. Hirose, editors, *Proceedings Eighth Inter-national Symposium on Robotics Research*, pages 95–101. Springer-Verlag, Berlin, 1998.

[404] H. H. Gonźalez-Baños, C.-Y. Lee, and J.-C. Latombe. Real-time combinatorial tracking of a target moving unpredictably among obstacles. In *Proceedings IEEE International Conference on Robotics & Automation*, 2002.

[405] J. E. Goodman and J. O'Rourke (eds). *Handbook of Discrete and Computational Geometry, 2nd Ed*. Chapman and Hall/CRC Press, New York, 2004.

[406] M. T. Goodrich and R. Tammasia. *Algorithm Design: Foundations, Analysis, and Internet Examples*. Wiley, New York, 2002.

[407] B. R. Gossick. *Hamilton's Principle and Physical Systems*. Academic, New York, 1967.

[408] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Proceedings ACM SIGGRAPH*, 1996.

[409] V. E. Gough and S. G. Whitehall. Universal tyre test machine. In *Proceedings 9th International Technical Congress F.I.S.I.T.A.*, May 1962.

[410] E. J. Griffith and S. Akella. Coordinating multiple droplets in planar array digital microfluidic systems. *International Journal of Robotics Research*, 24(11):933–949, 2005.

[411] R. Grossman, A. Nerode, A. Ravn, and H. Rischel (eds). *Hybrid Systems*. Springer-Verlag, Berlin, 1993.

[412] L. Grüne. An adaptive grid scheme for the discrete Hamilton-Jacobi-Bellman equation. *Numerische Mathematik*, 75:319–337, 1997.

[413] G. Sánchez and J.-C. Latombe. On delaying collision checking in PRM planning: Application to multi-robot coordination. *International Journal of Robotics Research*, 21(1):5–26, 2002.

[414] L. Guibas and R. Seidel. Computing convolution by reciprocal search. In *Proceedings ACM Symposium on Computational Geometry*, pages 90–99, 1986.

[415] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, 1985.

[416] L. J. Guibas, D. Hsu, and L. Zhang. H-Walk: Hierarchical distance computation for moving convex bodies. In *Proceedings ACM Symposium on Computational Geometry*, pages 265–273, 1999.

[417] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. *International Journal of Computational Geometry and Applications*, 9(5):471–494, 1999.

[418] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 269–282. A.K. Peters, Wellesley, MA, 1995.

[419] L. Guilamo, B. Tovar, and S. M. LaValle. Pursuit-evasion in an unknown environment using gap navigation trees. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[420] K. Gupta and Z. Guo. Motion planning with many degrees of freedom: Sequential search with backtracking. *IEEE Transactions on Robotics & Automation*, 6(11):897–906, 1995.

[421] K. Gupta and X. Zhu. Practical motion planning for many degrees of freedom: A novel approach within sequential framework. *Journal of Robotic Systems*, 2(12):105–118, 1995.

[422] S. K. Gupta, D. A. Bourne, K. Kim, and S. S. Krishnan. Automated process planning for robotic sheet metal bending operations. *Journal of Manufacturing Systems*, 17(5):338–360, 1998.

[423] L. Gurvits. Averaging approach to nonholonomic motion planning. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2541–2546, 1992.

[424] J.-S. Gutmann, T. Weigel, and B. Nebel. A fast, accurate, and robust method for self-localization in polygonal environments using laser-range-finders. *Advanced Robotics*, 14(8):651–668, 2001.

[425] O. Hájek. *Pursuit Games*. Academic, New York, 1975.

[426] K. Haji-Ghassemi. On differential games of fixed duration with phase coordinate restrictions on one player. *SIAM Journal on Control & Optimization*, 28(3):624– 652, May 1990.

[427] H. Halkin. Mathematical foundation of system optimization. In G. Leitman, editor, *Topics in Optimization*. Academic, New York, 1967.

[428] P. R. Halmos. *Measure Theory*. Springer-Verlag, Berlin, 1974.

[429] D. Halperin. Arrangements. In J. E. Goodman and J. O'Rourke, editors, *Hand-book of Discrete and Computational Geometry, 2nd Ed.*, pages 529–562. Chapman and Hall/CRC Press, New York, 2004.

[430] D. Halperin, J.-C. Latombe, and R. H. Wilson. A general framework for assembly planning: the motion space approach. In *Proceedings ACM Symposium on Computational Geometry*, pages 9–18, 1998.

[431] D. Halperin and M. Sharir. A near-quadratic algorithm for planning the motion of a polygon in a polygonal environment. *Discrete and Computational Geometry*, 16:121–134, 1996.

[432] D. Halperin and R.Wilson. Assembly partitioning along simple paths: the case of multiple translations. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1585–1592, 1995.

[433] J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2:84–90, 1960.

[434] J. M. Hammersley. Monte-Carlo methods for solving multivariable problems. *Annals of the New York Academy of Science*, 86:844–874, 1960.

[435] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 233–246. A.K. Peters, Wellesley, MA, 2001.

[436] H. Harrison and T. Nettleton. *Advanced Engineering Dynamics*. Elsevier, New York, 1997.

[437] R. S. Hartenberg and J. Denavit. A kinematic notation for lower pair mechanisms based on matrices.

Journal of Applied Mechanics, 77:215–221, 1955.

[438] R. S. Hartenberg and J. Denavit. *Kinematic Synthesis of Linkages*. McGraw-Hill, New York, 1964.

[439] J.W. Hartmann. *Counter-Intuitive Behavior in Locally Optimal Solar Sail Escape Trajectories*. PhD thesis, University of Illinois, Urbana, IL, May 2005.

[440] J. W. Hartmann, V. L. Coverstone, and J. E. Prussing. Optimal counter-intuitive solar sail escape trajectories. In *Proceedings AIAA/AAS Space Flight Mechanics Conference*, 2004. Paper AAS 04-279.

[441] R. Hartshorne. *Algebraic Geometry*. Springer-Verlag, Berlin, 1977.

[442] A. Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge, U.K., 2002. Available at http://www.math.cornell.edu/~hatcher/AT/ATpage.html.

[443] M. T. Heath. *Scientific Computing: An Introductory Survey, 2nd Ed.* McGraw-Hill, New York, 2002.

[444] G. Heinzinger, P. Jacobs, J. Canny, and B. Paden. Time-optimal trajectories for a robotic manipulator: A provably good approximation algorithm. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 150–155,Cincinnati, OH, 1990.

[445] H. Hermes, A. Lundell, and D. Sullivan. Nilpotent bases for distributions and control systems. *Journal of Differential Equations*, 55(3):385–400, 1984.

[446] J. Hershberger and S. Suri. Efficient computation of Euclidean shortest paths in the plane. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 508–517, 1995.

[447] S. Hert, S. Tiwari, and V. Lumelsky. A terrain-covering algorithm for an AUV. *Autonomous Robots*, 3:91–119, 1996.

[448] F. J. Hickernell. Lattice rules: How well do they measure up? In P. Bickel, editor, *Random and Quasi-Random Point Sets*, pages 109–166. Springer-Verlag, Berlin, 1998.

[449] F. J. Hickernell, H. S. Hong, P. L'Ecuyer, and C. Lemieux. Extensible lattice sequences for quasi-monte carlo quadrature. *SIAM Journal on Scientific Computing*, 22:1117–1138, 2000.

[450] R. Hinkel and T. Knieriemen. Environment perception with a laser radar in a fast moving robot. In *Proceedings Symposium on Robot Control*, pages 68.1–68.7, Karlsruhe, Germany, 1988.

[451] Y. Hirano, K. Kitahama, and S. Yoshizawa. Image-based object recognition and dextrous hand/arm motion planning using RRTs for grasping in cluttered scene. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[452] M. W. Hirsch. *Differential Topology*. Springer-Verlag, Berlin, 1994.

[453] Y.-C. Ho and K.-C. Chu. Team decision theory and information structures in optimal control problems-Part I. In *IEEE Transactions on Automatic Control*, pages 15–22, 1972.

[454] J. G. Hocking and G. S. Young. *Topology*. Dover, New York, 1988.

[455] J. K. Hodgins and W. L. Wooten. Animating human athletes. In Y. Shirai and S. Hirose, editors, *Proceedings International Symposium on Robotics Research*, pages 356–367. Springer-Verlag, Berlin, 1998.

[456] R. L. Hoffman. Automated assembly in a CSG domain. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 210–215, 1989.

[457] C. M. Hoffmann. *Geometric and Solid Modeling*. Morgan Kaufmann, San Francisco, CA, 1989.

[458] C. Holleman and L. E. Kavraki. A framework for using the workspace medial axis in PRM planners. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1408–1413, 2000.

[459] J. Hollerbach. Dynamic scaling of manipulator trajectories. Technical report, MIT A.I. Lab Memo 700, 1983.

[460] J. Hollerbach. Dynamic scaling of manipulator trajectories. In *Proceedings American Control Conference*, pages 752–756, 1983.

[461] L. S. Homem de Mello and A. C. Sanderson. Representations of mechanical assembly sequences. *IEEE*

Transactions on Robotics & Automation, 7(2):211–227, 1991.

[462] H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, San Francisco, 2004.

[463] J. Hopcroft, D. Joseph, and S. Whitesides. Movement problems for 2-dimensional linkages. In J .T .Schwartz, M. Sharir, and J. Hopcroft, editors, *Planning, Geometry, and Complexity of Robot Motion*, pages 282–329. Ablex, Norwood, NJ, 1987.

[464] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the "warehouseman's problem". *International Journal of Robotics Research*, 3(4):76–88, 1984.

[465] J. E. Hopcroft, J. D. Ullman, and R. Motwani. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 2000.

[466] T. Horsch, F. Schwarz, and H. Tolle. Motion planning for many degrees of freedom: Random reflections at C-space obstacles. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3318–3323, San Diego, CA, April 1994.

[467] A. E. Howe and E. Dahlman. A critical assessment of benchmark comparison in planning. *Journal of Artificial Intelligence Research*, pages 1–33, 2002.

[468] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proceedings IEEE International Conference on Robotics & Automation*, 2003.

[469] D. Hsu, R. Kindel, J-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*. A.K. Peters, Wellesley, MA, 2001.

[470] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *International Journal Computational Geometry & Applications*, 4:495–512, 1999.

[471] W. Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 27–32, 2001.

[472] T. W. Hungerford. *Algebra*. Springer-Verlag, Berlin, 1984.

[473] S. A. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. IEEE *Transactions on Robotics & Automation*, 12(5):651–670, October 1996.

[474] M. Hutter. *Universal Artificial Intelligence*. Springer-Verlag, Berlin, 2005.

[475] Y. K. Hwang and N. Ahuja. Gross motion planning–A survey. *ACM Computing Surveys*, 24(3):219–291, September 1992.

[476] S. Iannitti and K. M. Lynch. Exact minimum control switch motion planning for the snakeboard. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[477] C. Icking, G. Rote, E. Welzl, and C.-K. Yap. Shortest paths for line segments. *Algorithmica*, 10:182–200, 1992.

[478] P. Indyk. Nearest neighbors in high-dimensional spaces. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 877–892. Chapman and Hall/CRC Press, New York, 2004.

[479] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings ACM Symposium on Theory of Computing*, pages 604–613, 1998.

[480] R. Isaacs. *Differential Games*. Wiley, New York, 1965.

[481] A. Isidori. *Nonlinear Control Systems, 2nd Ed.* Springer-Verlag, Berlin, 1989.

[482] P. Isto. Constructing probabilistic roadmaps with powerful local planning and path optimization. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2323–2328, 2002.

[483] G. Jacob. Lyndon discretization and exact motion planning. In *Proceedings European Control Conference*, 1991.

[484] H. Jacob, S. Feder, and J. Slotine. Real-time path planning using harmonic potential functions in dynamic environment. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 874–881, 1997.

[485] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2–7, 1989.

[486] P. Jensfelt. *Approaches to Mobile Robot Localization.* PhD thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, 2001.

[487] P. Jensfelt and H. I. Christensen. Pose tracking using laser scanning and minimalistic environmental models. *IEEE Transactions on Robotics & Automation*, 17(2):138–147, 2001.

[488] P. Jensfelt and S. Kristensen. Active global localisation for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics & Automation*, 17(5):748–760, October 2001.

[489] X. Ji and J. Xiao. Planning motion compliant to complex contact states. *International Journal of Robotics Research*, 20(6):446–465, 2001.

[490] Y.-B. Jia. Computation on parametric curves with an application in grasping. *International Journal of Robotics Research*, 23(7-8):825–855, 2004.

[491] P. Jiménez, F. Thomas, and C. Torras. Collision detection algorithms for motion planning. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 1–53. Springer-Verlag, Berlin, 1998.

[492] D. Jordan and M. Steiner. Configuration spaces of mechanical linkages. *Discrete and Computational Geometry*, 22:297–315, 1999.

[493] D. A. Joseph and W. H. Plantiga. On the complexity of reachability and motion planning questions. In *Proceedings ACM Symposium on Computational Geometry*, pages 62–66, 1985.

[494] V. Jurdjevic. The geometry of the plate-ball problem. *Archives for Rational Mechanics and Analysis*, 124:305–328, 1993.

[495] V. Jurdjevic. *Geometric Control Theory.* Cambridge University Press, Cambridge, U.K., 1997.

[496] L. P. Kaelbling, A. Cassandra, and J. Kurien. Acting under uncertainty: Discrete Bayesian models for mobile robot navigation. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots* and *Systems*, pages 963–972, 1996.

[497] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence Journal*, 101, 1998.

[498] S. Kagami, J. Kuffner, K. Nishiwaki, and K. Okada M. Inaba. Humanoid arm motion planning using stereo vision and RRT search. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[499] D. W. Kahn. *Topology: An Introduction to the Point-Set and Algebraic Areas.* Dover, New York, 1995.

[500] H. Kaindl and G. Kainz. Bidirectional heuristic search reconsidered. *Journal of Artificial Intelligence Research*, pages 283–317, December 1997.

[501] M. Kallmann and M. Mataric. Motion planning using dynamic roadmaps. In *Proceedings IEEE International Conference on Robotics & Automation*, 2004.

[502] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann. Planning collision-free reaching motions for interactive object manipulation and grasping. *Eurographics*, 22(3), 2003.

[503] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82:35–45, 1960.

[504] R. E. Kalman, Y.-C. Ho, and K. S. Narendra. Controllability of dynamical systems. *Contributions to Differential Equations*, 1:189–213, 1963.

[505] M. H. Kalos and P. A. Whitlock. *Monte Carlo Methods*. Wiley, New York, 1986.

[506] T. Kameda, M. Yamashita, and I. Suzuki. On-line polygon search by a seven-state boundary 1-searcher. *IEEE Transactions on Robotics*, 2006. To appear.

[507] I. Kamon and E. Rivlin. Sensory-based motion planning with global proofs. *IEEE Transactions on Robotics & Automation*, 13(6):814–822, December 1997.

[508] I. Kamon, E. Rivlin, and E. Rimon. Range-sensor based navigation in three dimensions. In *Proceedings IEEE International Conference on Robotics & Automation*, 1999.

[509] K. Kant and S. W. Zucker. Toward efficient trajectory planning: The pathvelocity decomposition. *International Journal of Robotics Research*, 5(3):72–89, 1986.

[510] M.-Y. Kao, J. H. Reif, and S. R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *In SODA: ACM-SIAM Symposium on Discrete Algorithms*, pages 441–447, 1993.

[511] W. Kaplan. *Advanced Calculus*. Addison-Wesley, Reading, MA, 1984.

[512] T. Karatas and F. Bullo. Randomized searches and nonlinear programming in trajectory planning. In *IEEE Conference on Decision and Control*, 2001.

[513] R. M. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? In *Proceedings World Computer Congress*, 1992.

[514] L. Kauffman. *Knots and Applications*. World Scientific, River Edge, NJ, 1995.

[515] H. Kautz, D. McAllester, and B. Selman. Encoding plans in propositional logic. In *Proceedings International Conference on Knowledge Representation and Reasoning*, 1996.

[516] L. E. Kavraki. Computation of configuration-space obstacles using the Fast Fourier Transform. *IEEE Transactions on Robotics & Automation*, 11(3):408– 413, 1995.

[517] L. E. Kavraki. Geometry and the discovery of new ligands. In J.-P. Laumond and M. H. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 435–445. A.K. Peters, Wellesley, MA, 1997.

[518] L. E. Kavraki and M. Kolountzakis. Partitioning a planar assembly into two connected parts is NP-complete. *Information Processing Letters*, 55(3):159–165, 1995.

[519] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics & Automation*, 12(4):566–580, June 1996.

[520] Y. Ke and J. O'Rourke. Lower bounds on moving a ladder in two and three dimensions. *Discrete and Computational Geometry*, 3:197–217, 1988.

[521] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete and Computational Geometry*, 1:59–71, 1986.

[522] J. M. Keil. Polygon decomposition. In J. R. Sack and J. Urrutia, editors, *Hand-book on Computational Geometry*. Elsevier, New York, 2000.

[523] A. Kelly and B. Nagy. Reactive nonholonomic trajectory generation via parametric optimal control. *International Journal of Robotics Research*, 22(7-8):583–601, 2003.

[524] J. F. Kenney and E. S. Keeping. *Mathematics of Statistics, Part 2, 2nd ed.* Van Nostrand, Princeton, NJ, 1951.

[525] L. Kettner. Designing a data structure for polyhedral surfaces. In *Proceedings ACM Symposium on Computational Geometry*, pages 146 –154, 1998.

[526] H. K. Khalil. *Nonlinear Systems*. Macmillan, New York, 2002.

[527] W. Khalil and J. F. Kleinfinger. A new geometric notation for open and closedloop robots. *In Proceedings IEEE International Conference on Robotics & Automation*, volume 3, pages 1174 –1179, 1986.

[528] O. Khatib. *Commande dynamique dans l'espace opérational des robots manipulateurs en présence*

d'obstacles. PhD thesis, Ecole Nationale de la Statistique et de l'Administration Economique, France, 1980.

[529] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[530] A. A. Kilin. The dynamics of Chaplygin ball: The qualitative and computer analysis. *Regular and Chaotic Dynamics*, 6(3):291–306, 2001.

[531] J. Kim and J. P. Ostrowski. Motion planning of aerial robot using rapidlyexploring random trees with dynamic constraints. In *Proceedings IEEE International Conference on Robotics & Automation*, 2003.

[532] J.-O. Kim and P. Khosla. Real-time obstacle avoidance using harmonic potential functions. Technical report, Carnegie Mellon University, Pittsburgh, PA, 1990.

[533] K. H. Kim and F. W. Roush. *Team Theory*. Ellis Horwood Limited, Chichester, U.K., 1987.

[534] J. T. Kimbrell. *Kinematic Analysis and Synthesis*. McGraw-Hill, New York, 1991.

[535] R. Kimmel, N. Kiryati, and A. M. Bruckstein. Multivalued distance maps for motion planning on surfaces with moving obstacles. *IEEE Transactions on Robotics & Automation*, 14(3):427–435, June 1998.

[536] R. Kimmel and J. Sethian. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences*, USA, 95(15):8431–8435, 1998.

[537] R. Kimmel and J. Sethian. Optimal algorithm for shape from shading and path planning. *Journal of Mathematical Imaging and Vision*, 14(3):234–244, 2001.

[538] C. L. Kinsey. *Topology of Surfaces*. Springer-Verlag, Berlin, 1993.

[539] G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1), 1996.

[540] J. M. Kleinberg. On-line algorithms for robot navigation and server problems. Technical Report MIT/LCS/TR-641, MIT, Cambridge, MA, May 1994.

[541] J. M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Proceedings ACM Symposium on Theory of Computing*, pages 599–608, May 1997.

[542] S. A. Klugman, H. H. Panjer, and G. E. Willmot. *Loss Models: From Data to Decisions, 2nd Ed.* Wiley, New York, 2004.

[543] D. E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms, 3rd Ed.* Addison-Wesley, Reading, MA, 1998.

[544] D. E. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1–6, 1987.

[545] D. E. Koditschek. An approach to autonomous robot assembly. *Robotica*, 12:137–155, 1994.

[546] S. Koenig and M. Likhachev. D* lite. In *Proceedings AAAI National Conference on Artificial Intelligence*, pages 476–483, 2002.

[547] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe. Planning motions with intentions. *Proceedings ACM SIGGRAPH*, pages 395–408, 1994.

[548] D. Koller, N. Megiddo, and B. von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14:247–259, 1996.

[549] A. N. Kolmogorov and S. V. Fomin. *Introductory Real Analysis*. Dover, New York, 1975.

[550] V. Koltun. Pianos are not flat: Rigid motion planning in three dimensions. In *Proceedings ACM-SIAM Symposium on Discrete Algorithms*, 2005.

[551] K. Kondo. Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration. *IEEE Transactions on Robotics & Automation*, 7(3):267–277, 1991.

[552] K. Konolige. Markov localization using correlation. In *Proceedings International Joint Conference on*

Artificial Intelligence, 1999.

[553] R. E. Korf. Search: A survey of recent results. In H. E. Shrobe, editor, *Exploring Artificial Intelligence: Survey Talks from the National Conference on Artificial Intelligence*. Moran Kaufmann, San Francisco, CA, 1988.

[554] R. E. Korf. Artificial intelligence search algorithms. In *Algorithms and Theory of Computation Handbook*. CRC Press, Boca Raton, FL, 1999.

[555] K. Kotay, D. Rus, M. Vora, and C. McGray. The self-reconfiguring robotic molecule: Design and control algorithms. In P. K. Agarwal, L. E. Kavraki, and M. T. Mason, editors, *Robotics: The Algorithmic Perspective*. A.K. Peters, Natick, MA, 1998.

[556] L. Kovar and M. Gleicher. Automated extraction and parameterization of motions in large data sets. In *Proceedings ACM SIGGRAPH*, 2004.

[557] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proceedings ACM SIGGRAPH*, 2002.

[558] K. Kozlowski, P. Dutkiewicz, and W. Wróblewski. *Modeling and Control of Robots*. Wydawnictwo Naukowe PWN, Warsaw, Poland, 2003. In Polish.

[559] P. S. Krishnaprasad and D. P. Tsakaris. Oscillations, SE(2)-snakes and motion control: A study of the roller racer. Technical report, Center for Dynamics and Control of Smart Structures, University of Maryland, 1998.

[560] J. J. Kuffner. *Autonomous Agents for Real-time Animation*. PhD thesis, Stanford University, Stanford, CA, 1999.

[561] J. J. Kuffner. *Some Computed Examples* [using RRT-Connect]. [Online], 2001. Available at http://www.kuffner.org/james/plan/examples.html.

[562] J. J. Kuffner. Effective sampling and distance metrics for 3D rigid body path planning. In *Proceedings IEEE International Conference on Robotics & Automation*, 2004.

[563] J. J. Kuffner and S. M. LaValle. An efficient approach to path planning using balanced bidirectional RRT search. Technical Report CMU-RI-TR-05-34, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2005.

[564] J. J. Kuffner, K. Nishiwaki, M. Inaba, and H. Inoue. Motion planning for humanoid robots. In *Proceedings International Symposium on Robotics Research*, 2003.

[565] H. W. Kuhn. Extensive games and the problem of information. In H. W. Kuhn and A.W. Tucker, editors, *Contributions to the Theory of Games*, pages 196–216. Princeton University Press, Princeton, NJ, 1953.

[566] J. B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, Princeton, NJ, 2002.

[567] P. R. Kumar and P. Varaiya. *Stochastic Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1986.

[568] H. J. Kushner. Numerical methods for continuous control problems in continuous time. *SIAM Journal on Control & Optimization*, 28:999–1048, 1990.

[569] H. J. Kushner and D. S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer-Verlag, Berlin, 1978.

[570] H. J. Kushner and P. G. Dupuis. *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer-Verlag, Berlin, 1992.

[571] K. N. Kutulakos, C. R. Dyer, and V. J. Lumelsky. Provable strategies for visionguided exploration in three dimensions. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1365–1371, 1994.

[572] J. B. H. Kwa. BS*: An admissible bidirectional staged heuristic search algorithm. *Artificial Intelligence Journal*, 38:95–109, 1989.

[573] H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems*. Wiley, New York, 1972.

[574] A. Ladd and L. E. Kavraki. Motion planning for knot untangling. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, Nice, France, December 2002.

[575] A. Ladd and L. E. Kavraki. Fast exploration for robots with dynamics. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, Zeist, The Netherlands, July 2004.

[576] A. Ladd and L. E. Kavraki. Measure theoretic analysis of probabilistic path planning. *IEEE Transactions on Robotics & Automation*, 20(2):229–242, 2004.

[577] G. Laffierriere and H. J. Sussmann. Motion planning for controllable systems without drift. In *Proceedings IEEE International Conference on Robotics & Automation*, 1991.

[578] F. Lamiraux, D. Bonnafous, and O. Lefebvre. Reactive path deformation for non-holonomic mobile robots. *IEEE Transactions on Robotics*, 20(6):967–977, December 2004.

[579] F. Lamiraux, E. Ferre, and E. Vallee. Kinodynamic motion planning: Connecting exploration trees using trajectory optimization methods. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3987–3992, 2004.

[580] F. Lamiraux and L. Kavraki. Path planning for elastic plates under manipulation constraints. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 151–156, 1999.

[581] F. Lamiraux and J.-P. Laumond. Flatness and small-time controllability of multibody mobile robots: Application to motion planning. *IEEE Transactions on Automatic Control*, 45(10):1878–1881, April 2000.

[582] F. Lamiraux, S. Sekhavat, and J.-P. Laumond. Motion planning and control for Hilare pulling a trailer. *IEEE Transactions on Robotics & Automation*, 15(4):640–652, August 1999.

[583] A. S. Lapaugh. Recontamination does not help to search a graph. *Journal of the ACM*, 40(2):224–245, April 1993.

[584] P.-S. Laplace. *Théorie Analityque des Probabilités*. Courceir, Paris, 1812.

[585] R. E. Larson. A survey of dynamic programming computational procedures. *IEEE Transactions on Automatic Control*, 12(6):767–774, December 1967.

[586] R. E. Larson and J. L. Casti. *Principles of Dynamic Programming, Part II*. Dekker, New York, 1982.

[587] R. E. Larson and W. G. Keckler. Optimum adaptive control in an unknown environment. *IEEE Transactions on Automatic Control*, 13(4):438–439, August 1968.

[588] J. P. LaSalle. Stability theory for ordinary differential equations. *Journal of Differential Equations*, 4:57–65, 1968.

[589] A. Lasota and M. C. Mackey. *Chaos, Fractals, and Noise: Stochastic Aspects of Dynamics, 2nd Ed.* Springer-Verlag, Berlin, 1995.

[590] J.-C. Latombe. A fast path planner for a car-like indoor mobile robot. In *Proceedings AAAI National Conference on Artificial Intelligence*, pages 659–665, 1991.

[591] J.-C. Latombe. *Robot Motion Planning*. Kluwer, Boston, MA, 1991.

[592] J.-C. Latombe. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research*, 18(11):1119–1128, 1999.

[593] J.-C. Latombe, A. Lazanas, and S. Shekhar. Robot motion planning with uncertainty in control and sensing. *Artificial Intelligence Journal*, 52:1–47, 1991.

[594] M. Lau and J. J. Kuffner. Behavior planning for character animation. In *Proceedings Eurographics/SIGGRAPH Symposium on Computer Animation*, 2005.

[595] S. L. Laubach and J.W. Burdick. An autonomous sensor-based path-planning for planetary microrovers. In *Proceedings IEEE International Conference on Robotics & Automation*, 1999.

[596] J.-P. Laumond. Trajectories for mobile robots with kinematic and environment constraints. In *Proceedings International Conference on Intelligent Autonomous Systems*, pages 346–354, 1986.

[597] J.-P. Laumond. Controllability of a multibody mobile robot. *IEEE Transactions on Robotics & Automation*, 9(6):755–763, December 1993.

[598] J.-P. Laumond. *Robot Motion Planning and Control*. Springer-Verlag, Berlin, 1998. Available online at http://www.laas.fr/~jpl/book.html.

[599] J.-P. Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in nonholonomic motion planning for mobile robots. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 1–53. Springer-Verlag, Berlin, 1998.

[600] S. M. LaValle. *A Game-Theoretic Framework for Robot Motion Planning*. PhD thesis, University of Illinois, Urbana, IL, July 1995.

[601] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Computer Science Dept., Iowa State University, Oct. 1998.

[602] S. M. LaValle. Robot motion planning: A game-theoretic foundation. *Algorithmica*, 26(3):430–465, 2000.

[603] S. M. LaValle, M. S. Branicky, and S. R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics Research*, 23(7/8):673–692, July/August 2004.

[604] S. M. LaValle, P. Finn, L. Kavraki, and J.-C. Latombe. A randomized kinematics-based approach to pharmacophore-constrained conformational search and database screening. *J. Computational Chemistry*, 21(9):731–747, 2000.

[605] S. M. LaValle, H. H. Gonźalez-Baños, C. Becker, and J.-C. Latombe. Motion strategies for maintaining visibility of a moving target. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 731–736, 1997.

[606] S. M. LaValle and J. Hinrichsen. Visibility-based pursuit-evasion: The case of curved environments. *IEEE Transactions on Robotics and Automation*, 17(2):196–201, April 2001.

[607] S. M. LaValle and S. A. Hutchinson. An objective-based stochastic framework for manipulation planning. In *Proceedings IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, pages 1772–1779, September 1994.

[608] S. M. LaValle and S. A. Hutchinson. An objective-based framework for motion planning under sensing and control uncertainties. *International Journal of Robotics Research*, 17(1):19–42, January 1998.

[609] S. M. LaValle and S. A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *IEEE Trans. on Robotics and Automation*, 14(6):912–925, December 1998.

[610] S. M. LaValle and P. Konkimalla. Algorithms for computing numerical optimal feedback motion strategies. *International Journal of Robotics Research*, 20(9):729–752, September 2001.

[611] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 473– 479, 1999.

[612] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Proceedings Workshop on the Algorithmic Foundations of Robotics*, 2000.

[613] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378– 400, May 2001.

[614] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, Wellesley, MA, 2001.

[615] S. M. LaValle, D. Lin, L. J. Guibas, J.-C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 737–742, 1997.

[616] S. M. LaValle and R. Sharma. On motion planning in changing, partially predictable environments.

International Journal of Robotics Research, 16(6):775–805, December 1997.

[617] S. Lazebnik. Visibility-based pursuit evasion in three-dimensional environments. Technical Report CVR TR 2001-01, Beckman Institute, University of Illinois, 2001.

[618] A. R. Leach and I. D. Kuntz. Conformational analysis of flexible ligands in macromolecular receptor sites. *Journal of Computational Chemistry*, 13(6):730–748, 1992.

[619] D. T. Lee and R. L. Drysdale. Generalization of Voronoi diagrams in the plane. *SIAM Journal on Computing*, 10:73–87, 1981.

[620] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars with human motion data. In *Proceedings ACM SIGGRAPH*, 2002.

[621] J.-H. Lee, S. Y. Shin, and K.-Y. Chwa. Visibility-based pursuit-evasions in a polygonal room with a door. In *Proceedings ACM Symposium on Computational Geometry*, 1999.

[622] D. H. Lehmer. Mathematical methods in large-scale computing units. In *Proceedings 2nd Symposium* on *Large-Scale Digital Computing Machinery*, pages 141–146, Cambridge, MA, 1951. Harvard University Press.

[623] J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg. Real-time robot motion planning using rasterizing computer graphics hardware. *Computer Graphics*, 24(4):327–335, August 1990.

[624] S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings IEEE International Conference on Robotics & Automation*, 2000.

[625] J. Leonard, H. Durrant-Whyte, and I. Cox. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4):89–96, 1992.

[626] N. E. Leonard and P. S. Krishnaprasad. Averaging for attitude control and motion planning. In *Proceedings IEEE Conference Decision & Control*, pages 3098–3104, December 1993.

[627] N. E. Leonard and P. S. Krishnaprasad. Motion control of drift-free left-invariant systems on lie groups. *IEEE Transactions on Automatic Control*, 40(9):1539–1554, 1995.

[628] D. Leven and M. Sharir. An efficient and simple motion planning algorithm for a ladder moving in a 2-dimensional space amidst polygonal barriers. *Journal of Algorithms*, 8:192–215, 1987.

[629] D. Leven and M. Sharir. Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagrams. *Discrete and Computational Geometry*, 2:9–31, 1987.

[630] P. Leven and S. A. Hutchinson. Real-time path planning in changing environments. *IEEE Transactions on Robotics & Automation*, 21(12):999–1030, December 2002.

[631] P. Leven and S. A. Hutchinson. Using manipulability to bias sampling during the construction of probabilistic roadmaps. *IEEE Transactions on Robotics & Automation*, 19(6):1020–1026, December 2003.

[632] A. D. Lewis, J. P. Ostrowski, J. W. Burdick, and R. M. Murray. Nonholonomic mechanics and locomotion: The snakeboard example. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2391–2400, 1994.

[633] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, Berlin, 1997.

[634] T.-Y. Li and Y.-C. Shie. An incremental learning approach to motion planning with roadmap management. In *Proceedings IEEE International Conference on Robotics & Automation*, 2002.

[635] Z. Li and J. F. Canny. Motion of two rigid bodies with rolling constraint. *IEEE Transactions on Robotics & Automation*, 6(1):62–72, February 1990.

[636] Z. Li and J. F. Canny. *Nonholonomic Motion Planning*. Kluwer, Boston, MA, 1993.

[637] D. Liberzon. *Switching in Systems and Control*. Birkhäuser, Boston, MA, 2003.

[638] J.-M. Lien, S. L. Thomas, and N. M. Amato. A general framework for sampling on the medial axis of the

free space. In *Proceedings IEEE International Conference on Robotics & Automation*, 2003.

[639] M. C. Lin and J. F. Canny. Efficient algorithms for incremental distance computation. In *Proceedings IEEE International Conference on Robotics & Automation*, 1991.

[640] M. C. Lin and D. Manocha. Collision and proximity queries. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 787–807. Chapman and Hall/CRC Press, New York, 2004.

[641] M. C. Lin, D. Manocha, J. Cohen, and S. Gottschalk. Collision detection: Algorithms and applications. In J.-P. Laumond and M. H. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 129–142. A.K. Peters, Wellesley, MA, 1997.

[642] S. R. Lindemann and S. M. LaValle. Incremental low-discrepancy lattice methods for motion planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2920–2927, 2003.

[643] S. R. Lindemann and S. M. LaValle. Current issues in sampling-based motion planning. In P. Dario and R. Chatila, editors, *Proceedings International Symposium on Robotics Research*. Springer-Verlag, Berlin, 2004.

[644] S. R. Lindemann and S. M. LaValle. Incrementally reducing dispersion by increasing Voronoi bias in RRTs. *In Proceedings IEEE International Conference on Robotics and Automation*, 2004.

[645] S. R. Lindemann and S. M. LaValle. Steps toward derandomizing RRTs. In *IEEE Fourth International Workshop on Robot Motion and Control*, 2004.

[646] S. R. Lindemann and S. M. LaValle. Smoothly blending vector fields for global robot navigation. In *Proceedings IEEE Conference Decision & Control*, pages 3353–3559, 2005.

[647] S. R. Lindemann, A. Yershova, and S. M. LaValle. Incremental grid sampling strategies in robotics. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, pages 297–312, 2004.

[648] A. Lingas. The power of non-rectilinear holes. In *Proceedings 9th International Colloquium on Automata, Languange, and Programming*, pages 369–383. Springer-Verlag, 1982. Lecture Notes in Computer Science, 140.

[649] F. Lingelbach. Path planning using probabilistic cell decomposition. In *Proceedings IEEE International Conference on Robotics & Automation*, 2004.

[650] M. Littman. The witness algorithm: Solving partially observable Markov decision processes. Technical Report CS-94-40, Brown University, Providence, RI, 1994.

[651] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially-observable environments: Scaling up. In *Proceedings International Conference on Machine Learning*, pages 362–370, 1995.

[652] C. K. Liu and Z. Popovic. Synthesis of complex dynamic character motion from simple animations. In *Proceedings ACM SIGGRAPH*, pages 408– 416, 2002.

[653] Y. Liu and S. Arimoto. Path planning using a tangent graph for mobile robots among polygonal and curved obstacles. *International Journal of Robotics Research*, 11(4):376 –382, 1992.

[654] S. G. Loizou and K. J. Kyriakopoulos. Closed loop navigation for multiple holonomic vehicles. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[655] S. G. Loizou and K. J. Kyriakopoulos. Closed loop navigation for multiple nonholonomic vehicles. In *Proceedings IEEE International Conference on Robotics & Automation*, 2003.

[656] I. Lotan, F. Schwarzer, D. Halperin, and J.-C. Latombe. Efficient maintenance and self-collision testing for kinematic chains. In *Proceedings ACM Symposium on Computational Geometry*, pages 43–52, 2002.

[657] I. Lotan, H. van den Bedem, A. M. Deacon, and J.-C. Latombe. Computing protein structures from electron density maps: The missing loop problem. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, 2004.

[658] W. S. Lovejoy. Computationally feasible bounds for partially observed Markov decision processes. *Operations Research*, 39(1):162–175, 1991.

[659] T. Lozano-Pérez. Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems, Man, & Cybernetics*, 11(10):681– 698, 1981.

[660] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computing*, C-32(2):108–120, 1983.

[661] T. Lozano-Pérez. A simple motion-planning algorithm for general robot manipulators. *IEEE Journal of Robotics & Automation*, RA-3(3):224–238, Jun 1987.

[662] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of finemotion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, 1984.

[663] T. Lozano-Pérez and M. A.Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[664] L. Lu and S. Akella. Folding cartons with fixtures: A motion planning approach. IEEE *Transactions on Robotics & Automation*, 16(4):346 –356, Aug 2000.

[665] P. Lu and J. M. Hanson. Entry guidance for the X-33 vehicle. *Journal of Spacecraft and Rockets*, 35(3):342–349, 1998.

[666] A. De Luca, G. Oriolo, and C. Samson. Feedback control of a nonholonomic car-like robot. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 171–253. Springer-Verlag, Berlin, 1998.

[667] D. G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Wiley, New York, 1973.

[668] V. J. Lumelsky and K. R. Harinarayan. Decentralized motion planning for multiple mobile robots: The cocktail party model. *Autonomous Robots*, 4(1):121–135, 1997.

[669] V. J. Lumelsky and T. Skewis. Incorporating range sensing in the robot navigation function. *IEEE Transactions on Systems*, Man, & Cybernetics, 20(5):1058–1069, 1990.

[670] V. J. Lumelsky and A. A. Stepanov. Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[671] V. J. Lumelsky and S. Tiwari. An algorithm for maze searching with azimuth input. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 111–116, 1994.

[672] K. M. Lynch. Controllability of a planar body with unilateral thrusters. *IEEE Transactions on Automatic Control*, 44(6):1206–1211, 1999.

[673] K. M. Lynch and M. T. Mason. Pulling by pushing, slip with infinite friction, and perfectly rough surfaces. *International Journal of Robotics Research*, 14(2):174 –183, 1995.

[674] K. M. Lynch and M. T. Mason. Stable pushing: Mechanics, controllability, and planning. *International Journal of Robotics Research*, 15(6):533–556, 1996.

[675] K. M. Lynch, N. Shiroma, H. Arai, and K. Tanie. Collision free trajectory planning for a 3-dof robot with a passive joint. *International Journal of Robotics Research*, 19(12):1171–1184, 2000.

[676] I. M. Makarov, T. M. Vinogradskaya, A. A. Rubchinsky, and V. B. Sokolov. *The Theory of Choice and Decision Making*. Mir Publishers, Moscow, 1987.

[677] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for on-line problems. In *Proceedings ACM Symposium on Theory of Computing*, pages 322–333, 1988.

[678] D. Manocha and J. Canny. Real time inverse kinematics of general 6R manipulators. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 383–389, Nice, May 1992.

[679] A. Marigo and A. Bicchi. Rolling bodies with regular surface: Controllability theory and applications. *IEEE Transactions on Automatic Control*, 45(9):1586–1599, 2000.

[680] J. E. Marsden and T. S. Ratiu. *Introduction to Mechanics and Symmetry*. Springer-Verlag, Berlin, 1999.

[681] M. T. Mason. Compliance and force control for computer controlled manipulators. In M. Brady et al.,

editor, *Robot Motion: Planning and Control*, pages 373–404. MIT Press, Cambridge, MA, 1982.

[682] M. T. Mason. The mechanics of manipulation. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 544–548, 1985.

[683] M. T. Mason. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research*, 5(3):53–71, 1986.

[684] M. T. Mason. *Mechanics of Robotic Manipulation*. MIT Press, Cambridge, MA, 2001.

[685] J. Matousek. *Geometric Discrepancy*. Springer-Verlag, Berlin, 1999.

[686] J. Matousek and J. Nesetril. *Invitation to Discrete Mathematics*. Oxford University Press, Oxford, U.K., 1998.

[687] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, January 1998.

[688] L. Matthies and A. Elfes. Integration of sonar and stereo range data using a gridbased representation. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 727–733, 1988.

[689] O. Mayr. *The Origins of Feedback Control*. MIT Press, Cambridge, MA, 1970.

[690] E. Mazer, J. M. Ahuactzin, and P. Bessière. The Ariadne's clew algorithm. *Journal of Artificial Intelligence Research*, 9:295–316, November 1998.

[691] E. Mazer, G. Talbi, J. M. Ahuactzin, and P. Bessière. The Ariadne's clew algorithm. In *Proceedings International Conference of Society of Adaptive Behavior*, Honolulu, 1992.

[692] J. M. McCarthy. *Geometric Design of Linkages. Springer-Verlag*, Berlin, 2000.

[693] D. J. McGill and W. W. King. *An Introduction to Dynamics*. PWS, Boston, MA, 1995.

[694] R. McKelvey and A. McLennan. Computation of equilibria in finite games. In H. Amman, D. A. Kendrick, and J .Rust, editors, *The Handbook of Computational Economics*, pages 87–142. Elsevier, New York, 1996.

[695] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *Journal of the ACM*, 35(1):18–44, January 1988.

[696] J.-P. Merlet. *Parallel Robots*. Kluwer, Boston, MA, 2000.

[697] A. W. Merz. The game of two identical cars. *Journal of Optimization Theory & Applications*, 9(5):324–343, 1972.

[698] N. C. Metropolis and S. M. Ulam. The Monte-Carlo method. *Journal of the American Statistical Association*, 44:335–341, 1949.

[699] A. N. Michel and C. J. Herget. *Applied Algebra and Functional Analysis*. Dover, New York, 1993.

[700] R. J. Milgram and J. C. Trinkle. Complete path planning for closed kinematic chains with spherical joints. *International Journal of Robotics Research*, 21(9):773–789, 2002.

[701] R. J. Milgram and J. C. Trinkle. The geometry of configuration spaces for closed chains in two and three dimensions. *Homology, Homotopy, and Applications*, 6(1):237–267, 2004.

[702] D. A. Miller and S. W. Zucker. Copositive-plus Lemke algorithm solves polymatrix games. *Operations Research Letters*, 10:285–290, 1991.

[703] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Separators for spherepackings and nearest neighbor graphs. *Journal of the ACM*, 44(1):1–29, January 1997.

[704] J. W. Milnor. *Morse Theory*. Princeton University Press, Princeton, NJ, 1963.

[705] B. Mirtich. V-Clip: Fast and robust polyhedral collision detection. Technical Report TR97-05, Mitsubishi Electronics Research Laboratory, 1997.

[706] B. Mirtich. Efficient algorithms for two-phase collision detection. In K. Gupta and A.P. del Pobil, editors, *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, pages 203–223.

Wiley, New York, 1998.

[707] B. Mishra. *Algorithmic Algebra*. Springer-Verlag, New York, 1993.

[708] B. Mishra. Computational real algebraic geometry. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 537–556. CRC Press, New York, 1997.

[709] I. Mitchell, A. Bayen, and C. Tomlin. Computing reachable sets for continuous dynamic games using level set methods. *IEEE Transactions on Automatic Control*, 2003. Submitted.

[710] I. Mitchell and C. J. Tomlin. Overapproximating reachable sets by Hamilton-Jacobi projections. *Journal of Scientific Computation*, 19(1):323–346, 2003. planning. International Journal of Robotics Research, 15(6):533–556, 1996.

[711] J. S. B. Mitchell. Shortest paths among obstacles in the plane. *International Journal Computational Geometry & Applications*, 6(3):309–332, 1996.

[712] J. S. B. Mitchell. Shortest paths and networks. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 607–641. Chapman and Hall/CRC Press, New York, 2004.

[713] J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem. *Journal of the ACM*, 38:18–73, 1991.

[714] T. M. *Mitchell. Machine Learning*. McGraw-Hill, New York, 1997.

[715] L. Molina-Tanco and A. Hilton. Realistic synthesis of novel human movements from a database of motion capture examples. In *Proceedings IEEE Workshop on Human Motion*, 2000.

[716] S. Monaco and D. Normand-Cyrot. An introduction to motion planning under multirate digital control. In *Proceedings IEEE Conference Decision & Control*, pages 1780–1785, 1992.

[717] G. Monahan. A survey of partially observable Markov decision processes. *Management Science*, 101(1):1–16, 1982.

[718] B. Monien and I. H. Sudborough. Min cut is NP-complete for edge weighted graphs. *Theoretical Computer Science*, 58:209–229, 1988.

[719] D. J. Montana. The kinematics of contact and grasp. *International Journal of Robotics Research*, 7(3):17–32, 1988.

[720] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings AAAI National Conference on Artificial Intelligence*, 1999.

[721] M. E. Mortenson. *Geometric Modeling, 2nd Ed.* Wiley, New York, 1997.

[722] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, U.K., 1995.

[723] R. Munos. Error bounds for approximate value iteration. In *Proceedings AAAI National Conference on Artificial Intelligence*, 2005.

[724] R. Munos and A. Moore. Barycentric interpolator for continuous space & time reinforcement learning. In *Proceedings Neural Information Processing Systems*, 1998.

[725] R. Munos and A. Moore. Variable resolution discretization in optimal control. *Machine Learning*, 49:291–323, 2001.

[726] T. Muppirala, R. Murrieta-Cid, and S. Hutchinson. Optimal motion strategies based on critical events to maintain visibility of a moving target. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3837–3842, 2005.

[727] T. Murphey. *Control of Multiple Model Systems*. PhD thesis, California Institute of Technology, May 2002.

[728] R. M. Murray, Z. Li, and S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press,

Boca Raton, FL, 1994.

[729] R. M. Murray, M. Rathinam, and W. M. Sluis. Differential flatness of mechanical control systems. In *Proceedings ASME International Congress and Exposition*, 1995.

[730] R. M. Murray and S. Sastry. Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control*, 38(5):700 –716, 1993.

[731] R. Murrieta-Cid, A. Sarmiento, S. Bhattacharya, and S. Hutchinson. Maintaining visibility of a moving target at a fixed distance: The case of observer bounded speed. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 479– 484, 2004.

[732] Y. Nakamura, T. Suzuki, and M. Koinuma. Nonlinear behavior and control of a nonholonomic free-joint manipulator. *IEEE Transactions on Robotics & Automation*, 13(6):853–862, 1997.

[733] J. Nash. Noncooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[734] S. G. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, New York, 1996.

[735] B. K. Natarajan. The complexity of fine motion planning. *International Journal of Robotics Research*, 7(2):36–42, 1988.

[736] B. K. Natarajan. On planning assemblies. In *Proceedings ACM Symposium on Computational Geometry*, pages 299–308, 1988.

[737] New York University. MathMol Library. Scientific Visualization Center. Available from http:// www. nyu. edu/pages/mathmol/library/, 2005.

[738] A. Newell and H. Simon. GPS: A program that simulates human thought. In E. A. Feigenbaum and J. Feldman, editors, *Computers and Thought*. McGraw-Hill, New York, 1963.

[739] W. S. Newman and M. S. Branicky. Real-time configuration space transforms for obstacle avoidance. *International Journal of Robotics Research*, 10(6):650–667, 1991.

[740] A. Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings Conference on Uncertainty in Artificial Intelligence*, 2000.

[741] H. Niederreiter. Random Number Generation and Quasi-Monte-Carlo Methods. *Society for Industrial and Applied Mathematics*, Philadelphia, 1992.

[742] H. Niederreiter and C. P. Xing. Nets, (t,s)-sequences, and algebraic geometry. In P. Hellekalek and G. Larcher, editors, *Random and Quasi-Random Point Sets*, pages 267–302. Springer-Verlag, Berlin, 1998. Lecture Notes in Statistics, 138.

[743] D. Nieuwenhuisen and M. H. Overmars. Useful cycles in probabilistic roadmap graphs. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 446–452, 2004.

[744] H. Nijmeijer and A. J. van der Schaft. *Nonlinear Dynamical Control Systems*. Springer-Verlag, Berlin, 1990.

[745] N. J. Nilsson. A mobile automaton: An application of artificial intelligence techniques. In *1st International Conference on Artificial Intelligence*, pages 509–520, 1969.

[746] N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Company, Wellsboro, PA, 1980.

[747] N. J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann, San Francisco, CA, 1998.

[748] S. Ntafos. Watchman routes under limited visibility. *Computational Geometry: Theory and Applications*, 1:149–170, 1992.

[749] P. A. O'Donnell and T. Lozano-Pérez. Deadlock-free and collision-free coordination of two robot manipulators. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 484 – 489, 1989.

[750] C. O'Dunlaing. Motion planning with inertial constraints. *Algorithmica*, 2(4):431– 475, 1987.

[751] C. O'Dunlaing, M. Sharir, and C. K. Yap. Retraction: A new approach to motion planning. In J .T .Schwartz, M. Sharir, and J. Hopcroft, editors, *Planning, Geometry, and Complexity of Robot Motion*,

pages 193–213. Ablex, Norwood, NJ, 1987.

[752] C. O'Dunlaing and C. K. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111, 1982.

[753] P. ögren. *Formations and Obstacle Avoidance in Mobile Robot Control.* PhD thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, 2003.

[754] J. M. O'Kane. Global localization using odometry. In *Proceedings IEEE International Conference on Robotics and Automation*, 2005.

[755] J. M. O'Kane and S. M. LaValle. Almost-sensorless localization. In *Proceedings IEEE International Conference on Robotics and Automation*, 2005.

[756] B. O'Neill. *Elementary Differential Geometry.* Academic, New York, 1966.

[757] S. Oore, G. E. Hinton, and G. Dudek. A mobile robot that learns its place. *Neural Computation*, 9:683–699, 1997.

[758] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, 1987.

[759] J. O'Rourke. Visibility. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 643–663. Chapman and Hall/CRC Press, New York, 2004.

[760] J. O'Rourke and S. Suri. Polygons. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 583–606. Chapman and Hall/CRC Press, New York, 2004.

[761] M. H. Overmars and J. van Leeuwen. Dynamic multidimensional data structures based on Quad- and K-D trees. *Acta Informatica*, 17:267–285, 1982.

[762] G. Owen. *Game Theory.* Academic, New York, 1982.

[763] B. Paden, A. Mees, and M. Fisher. Path planning using a Jacobian-based freespace generation algorithm. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1732–1737, 1989.

[764] D. K. Pai and L. M. Reissell. Multiresolution rough terrain motion planning. *IEEE Transactions on Robotics & Automation*, 14(5):709–717, 1998.

[765] S. Pancanti, L. Pallottino, D. Salvadorini, and A. Bicchi. Motion planning through symbols and lattices. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3914–3919, 2004.

[766] C. H. Papadimitriou. An algorithm for shortest-path planning in three dimensions. *Information Processing Letters*, 20(5):259-263, 1985.

[767] C. H. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985.

[768] C. H. Papadimitriou and K. J. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice Hall, Englewood Cliffs, NJ, 1982.

[769] C. H. Papadimitriou and J. N. Tsitsiklis. Intractable problems in control theory. *SIAM Journal of Control & Optimization*, 24(4):639– 654, July 1986.

[770] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441– 450, August 1987.

[771] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84:127–150, 1991.

[772] A. Papantonopoulou. *Algebra: Pure and Applied.* Prentice Hall, Englewood Cliffs, NJ, 2002.

[773] S.-M. Park, J.-H. Lee, and K.-Y. Chwa. Visibility-based pursuit-evasion in a polygonal region by a searcher. Technical Report CS/TR-2001-161, Dept. of Computer Science, KAIST, Seoul, South Korea, January 2001.

[774] R. Parr and A. Eliazar. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proceedings International Joint Conference on Artificial Intelligence*, 2003.

[775] R. Parr and S. Russell. Approximating optimal policies for partially observable stochàstic domains. In *Proceedings International Joint Conference on Artificial Intelligence*, 1995.

[776] T. D. Parsons. Pursuit-evasion in a graph. In Y. Alavi and D. R. Lick, editors, *Theory and Application of Graphs*, pages 426–441. Springer-Verlag, Berlin, 1976.

[777] T. Parthasarathy and M. Stern. Markov games: A survey. In *Differential Games and Control Theory II*, pages 1–46. Marcel Dekker, New York, 1977.

[778] R. P. Paul. *Robot Manipulators: Mathematics, Programming, and Control*. MIT Press, Cambridge, MA, 1981.

[779] R. P. Paul and B. Shimano. Compliance and control. In *Proceedings of the Joint American Automatic Control Conference*, pages 1694–1699, 1976.

[780] J. Pearl. *Heuristics*. Addison-Wesley, Reading, MA, 1984.

[781] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988.

[782] B. Peleg and P. Sudlölter. *Introduction to the Theory of Cooperative Games*. Springer-Verlag, Berlin, 2003.

[783] J. Peng and S. Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. In J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, editors, *Algorithmic Foundations of Robotics V (WAFR 2002)*, pages 221–237. Springer-Verlag, Berlin, 2002.

[784] J. Pertin-Troccaz. Grasping: A state of the art. In O. Khatib, J. J. Craig, and T. Lozano-Pérez, editors, *The Robotics Review 1*. MIT Press, Cambridge, MA, 1989.

[785] S. Petitjean, D. Kriegman, and J. Ponce. Computing exact aspect graphs of curved objects: algebraic surfaces. *International Journal of Computer Vision*, 9:231–255, Dec 1992.

[786] L. A. Petrosjan. Differential Games of Pursuit. *World Scientific*, Singapore, 1993.

[787] J. Pettré, J.-P. Laumond, and T. Siméon. A 2-stages locomotion planner for digital actors. In *Proceedings Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 258–264, 2003.

[788] F. Pfeiffer and R. Johanni. A concept for manipulator trajectory planning. *IEEE Journal of Robotics & Automation*, RA-3(2):115–123, 1987.

[789] J. M. Phillips, N. Bedrosian, and L. E. Kavraki. Spacecraft rendezvous and docking with real-time randomized optimization. In *Proceedings AIAA Guidance, Navigation and Control Conference*, 2003.

[790] A. Piazzi, M. Romano, and C. G. Lo Bianco. G3 splines for the path planning of wheeled mobile robots. In *Proceedings European Control Conference*, 2003.

[791] L. Piegl. On NURBS: A survey. *IEEE Transactions on Computer Graphics & Applications*, 11(1):55–71, Jan 1991.

[792] D. A. Pierre. *Optimization Theory with Applications*. Dover, New York, 1986.

[793] R. W. Pike. Optimization for Engineering Systems. [Online], 2001. Available at http://www.mpri.lsu.edu/bookindex.html.

[794] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration. In *Proceedings International Joint Conference on Artificial Intelligence*, pages 1025–1032, 2003.

[795] C. Pisula, K. Hoff, M. Lin, and D. Manocha. Randomized path planning for a rigid body based on hardware accelerated Voronoi sampling. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, 2000.

[796] R. Pito. A sensor based solution to the next best view problem. In *International Conference Pattern Recognition*, 1996.

[797] M. Pivtoraiko and A. Kelly. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[798] E. Plaku and L. E. Kavraki. Distributed sampling-based roadmap of trees for large-scale motion planning.

In *Proceedings IEEE International Conference on Robotics & Automation*, 2005.

[799] M. Pocchiola and G. Vegter. The visibility complex. *International Journal Computational Geometry & Applications*, 6(3):279–308, 1996.

[800] I. Pohl. Bi-directional and heuristic search in path problems. Technical report, Stanford Linear Accelerator Center, Stanford, CA, 1969.

[801] I. Pohl. Bi-directional search. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, pages 127–140. Elsevier, New York, 1971.

[802] J. Ponce and B. Faverjon. On computing three-finger force-closure grasps of polygonal objects. *IEEE Transactions on Robotics & Automation*, 11(6):868–881, 1995.

[803] J. Ponce, S. Sullivan, A. Sudsang, J.-D. Boissonnat, and J.-P. Merlet. On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *International Journal of Robotics Research*, 16(1):11–35, February 1997.

[804] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *L.S. Pontryagin Selected Works, Volume 4: The Mathematical Theory of Optimal Processes*. Gordon and Breach, Montreux, Switzerland, 1986.

[805] J. Popovic, S. M. Seitz, M. A. Erdmann, and Z. Popovic A. P.Wiktin. Interactive manipulation of rigid body simulations. In *Proceedings ACM SIGGRAPH*, pages 209–217, 2002.

[806] J. M. Porta, M. T. J. Spaan, and N. Vlassis. Robot planning in partially observable continuous domains. In *Proceedings Robotics: Science and Systems*, 2005.

[807] H. Pottman and J. Wallner. *Computational Line Geometry*. Springer-Verlag, Berlin, 2001.

[808] P. Poupart and C. Boutilier. Value-directed compression of POMDPs. In *Proceedings Neural Information Processing Systems*, 2003.

[809] F. P. Preparata and M. I. Shamos. *Computational Geometry*. Springer-Verlag, Berlin, 1985.

[810] S. Quinlan. Efficient distance computation between nonconvex objects. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3324 –3329, 1994.

[811] M. Rabin. Transaction protection by beacons. *Journal of Computation Systems Science*, 27(2):256–267, 1983.

[812] S. Rajko and S. M. LaValle. A pursuit-evasion bug algorithm. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1954–1960, 2001.

[813] A. Rao and K. Goldberg. Manipulating algebraic parts in the plane. *IEEE Transactions on Robotics & Automation*, 11(4):598–602, 1995.

[814] N. Rao, S. Kareti,W. Shi, and S. Iyenagar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410:1–58, Oak Ridge National Laboratory, July 1993.

[815] S. Ratering and M. Gini. Robot navigation in a known environment with unknown moving obstacles. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 25–30, 1993.

[816] M. Rathinam and R. M. Murray. Configuration flatness of Lagrangian systems underactuated by one control. *SIAM Journal of Control & Optimization*, 36(1):164–179, 1998.

[817] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[818] J. Reif and Z. Sun. On frictional mechanical systems and their computational power. *SIAM Journal on Computing*, 32(6):1449–1474, 2003.

[819] J. Reif and H.Wang. Non-uniform discretization approximations for kinodynamic motion planning. In J.-P. Laumond and M. H. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 97–112. A.K. Peters, Wellesley, MA, 1997.

[820] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 421– 427, 1979.

[821] J. H. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 144 –154, 1985.

[822] J. H. Reif and M. Sharir. Motion planning in the presence of moving obstacles. *Journal of the ACM*, 41:764–790, 1994.

[823] J. H. Reif and Z. Sun. An efficient approximation algorithm for weighted region shortest path problem. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 191–203. A.K. Peters, Wellesley, MA, 2001.

[824] J. H. Reif and S. R. Tate. Continuous alternation: The complexity of pursuit in continuous domains. *Algorithmica*, 10:157–181, 1993.

[825] J. Reimpell, H. Stoll, and J. W. Betzler. The Automotive Chassis: Engineering Principles. *Society of Automotive Engineers*, Troy, MI, 2001.

[826] E. M. Reingold, J. Nievergelt, and N. Deo. *Combinatorial Algorithms.* Prentice Hall, Englewood Cliffs, NJ, 1977.

[827] E. Remolina and B. Kuipers. Towards a general theory of topological maps. *Artificial Intelligence Journal*, 152(1):47–104, 2004.

[828] W. Rencken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2192–2197, 1993.

[829] E. Rimon and J. W. Burdick. Mobility of bodies in contact– I : A 2nd order mobility index for multiple-finger grasps. *IEEE Transactions on Robotics & Automation*, 14(5):696–708, 1998.

[830] E. Rimon and J. W. Burdick. Mobility of bodies in contact– II : How forces are generated by curvature effects. *IEEE Transactions on Robotics & Automation*, 14(5):709–717, 1998.

[831] E. Rimon and J. Canny. Construction of C-space roadmaps using local sensory data-What should the sensors look for? In *Proceedings IEEE International Conference on Robotics & Automation*, pages 117–124, 1994.

[832] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Transactions on Robotics & Automation*, 8(5):501–518, October 1992.

[833] A. A. Rizzi. Hybrid control as a method for robot motion programming. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 832–837, 1998.

[834] C. P. Robert. *The Bayesian Choice, 2nd. Ed.* Springer-Verlag, Berlin, 2001.

[835] H. Rohnert. Shortest paths in the plane with convex polygonal obstacles. *Information Processing Letters*, 23:71–76, 1986.

[836] I. M. Ross and F. Fahroo. Pseudospectral methods for optimal motion planning of differentially flat systems. *IEEE Transactions on Automatic Control*, 49(8):1410–1413, 2004.

[837] J. J. Rotman. *Introduction to Algebraic Topology.* Springer-Verlag, Berlin, 1988.

[838] N. Roy and G. Gordon. Exponential family PCA for belief compression in POMDPs. In *Proceedings Neural Information Processing Systems*, 2003.

[839] H. L. Royden. *Real Analysis.* MacMillan, New York, 1988.

[840] W. Rudin. *Real Analysis.* McGraw-Hill, New York, 1987.

[841] W. Rudin. *Functional Analysis, 2nd Ed.* McGraw-Hill, New York, 1991.

[842] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach, 2nd Edition.* Prentice-Hall, Englewood Cliffs, NJ, 2003.

[843] S. Sachs, S. Rajko, and S. M. LaValle. Visibility-based pursuit-evasion in an unknown planar

environment. *International Journal of Robotics Research*, 23(1):3–26, January 2004.

[844] H. Sagan. *Introduction to the Calculus of Variations*. Dover, New York, 1992.

[845] H. Sagan. *Space-Filling Curves*. Springer-Verlag, Berlin, 1994.

[846] G. Sahar and J. M. Hollerbach. Planning minimum-time trajectories for robot arms. *International Journal of Robotics Research*, 5(3):97–140, 1986.

[847] G. Sánchez and J.-C. Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Proceedings International Symposium on Robotics Research*, 2001.

[848] S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer-Verlag, Berlin, 1999.

[849] Y. Sawaragi, H. Nakayama, and T. Tanino. Theory of Multiobjective Optimization. *Academic*, New York, 1985.

[850] A. Scheuer and T. Fraichard. Collision-free and continuous-curvature path planning for car-like robots. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 867–873, 1997.

[851] A. Scheuer and C. Laugier. Planning sub-optimal and continuous-curvature paths for car-like robots. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 25–31, 1998.

[852] B. Schiele and J. Crowley. A comparison of position estimation techniques using occupancy grids. In *Proceedings IEEE International Conference on Robotics & Automation*, 1994.

[853] J. T. Schwartz and M. Sharir. On the Piano Movers' Problem: Ⅰ. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[854] J. T. Schwartz and M. Sharir. On the Piano Movers' Problem: Ⅱ. General techniques for computing topological properties of algebraic manifolds. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[855] J. T. Schwartz and M. Sharir. On the Piano Movers' Problem: III. Coordinating the motion of several independent bodies. *International Journal of Robotics Research*, 2(3):97–140, 1983.

[856] J. T. Schwartz, M. Sharir, and J. Hopcroft. *Planning, Geometry, and Complexity of Robot Motion*. Ablex, Norwood, NJ, 1987.

[857] F. Schwarzer, M. Saha, and J.-C. Latombe. Exact collision checking of robot paths. In J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, editors, *Algorithmic Foundations of Robotics V (WAFR 2002)*. Springer-Verlag, Berlin, 2002.

[858] L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. Springer-Verlag, Berlin, 1996.

[859] R. Sedgewick. *Algorithms in C++, 2nd Ed.* Addison-Wesley, Reading, MA, 2002.

[860] S. Sekhavat and J.-P. Laumond. Topological property for collision-free nonholonomic motion planning: The case of sinusoidal inputs for chained-form systems. *IEEE Transactions on Robotics & Automation*, 14(5):671–680, 1998.

[861] S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars. Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *International Journal of Robotics Research*, 17:840–857, 1998.

[862] N. F. Sepetov, V. Krchnak, M. Stankova, S. Wade, K. S. Lam, and M. Lebl. Library of libraries: Approach to synthetic combinatorial library design and screening of "pharmacophore" motifs. *Proceedings of the National Academy of Sciences*, USA, 92:5426–5430, June 1995.

[863] J.-P. Serre. *Lie Algebras and Lie Groups*. Springer-Verlag, Berlin, 1992.

[864] J. A. Sethian. *Level set methods : Evolving interfaces in geometry, fluid mechanics, computer vision, and materials science*. Cambridge University Press, Cambridge, U.K., 1996.

[865] A. A. Shabana. *Computational Dynamics*. Wiley, New York, 2001.

[866] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423,

1948.

[867] M. Sharir. Algorithmic motion planning. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 1037–1064. Chapman and Hall/CRC Press, New York, 2004.

[868] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, Cambridge, U.K., 1995.

[869] R. Sharma. Locally efficient path planning in an uncertain, dynamic environment using a probabilistic model. *IEEE Transactions on Robotics & Automation*, 8(1):105–110, February 1992.

[870] R. Sharma. A probabilistic framework for dynamic motion planning in partially known environments. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2459–2464, Nice, France, May 1992.

[871] R. Sharma, J.-Y. Hervé, and P. Cucka. Dynamic robot manipulation using visual tracking. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1844–1849, 1992.

[872] R. Sharma, S. M. LaValle, and S. A. Hutchinson. Optimizing robot motion strategies for assembly with stochastic models of the assembly process. *IEEE Trans. on Robotics and Automation*, 12(2):160–174, April 1996.

[873] R. Sharma, D. M. Mount, and Y. Aloimonos. Probabilistic analysis of some navigation strategies in a dynamic environment. *IEEE Transactions on Systems, Man, & Cybernetics*, 23(5):1465–1474, September 1993.

[874] R. W. Sharpe. *Differential Geometry*. Springer-Verlag, Berlin, 1997.

[875] H. Shatkay and L. P. Kaelbling. Learning topological maps with weak local odometric information. In *Proceedings International Joint Conference on Artificial Intelligence*, 1997.

[876] T. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, September 1992.

[877] C. L. Shih, T.-T. Lee, and W. A. Gruver. A unified approach for robot motion planning with moving polyhedral obstacles. *IEEE Transactions on Systems, Man, & Cybernetics*, 20:903–915, 1990.

[878] Z. Shiller and S. Dubowsky. On the optimal control of robotic manipulators with actuator and end-effector constraints. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 614 – 620, 1985.

[879] Z. Shiller and S. Dubowsky. On computing time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions on Robotics & Automation*, 7(6), Dec 1991.

[880] Z. Shiller and H.-H. Lu. Computation of path constrained time-optimal motions with dynamic singularities. *Transactions of the ASME, Journal of Dynamical Systems, Measurement, & Control*, 114: 34 – 40, 1992.

[881] K. G. Shin and N. D. McKay. Minimum-time control of robot manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6):531–541, 1985.

[882] K. G. Shin and N. D. McKay. A dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Transactions on Automatic Control*, 31(6):491–500, 1986.

[883] K. G. Shin and Q. Zheng. Minimum-time collision-free trajectory planning for dual-robot systems. *IEEE Transactions on Robotics & Automation*, 8(5):641 – 644, October 1992.

[884] A. M. Shkel and V. J. Lumelsky. Incorporating body dynamics into sensor-based motion planning: The maximum turn strategy. *IEEE Transactions on Robotics & Automation*, 13(6):873–880, December 1997.

[885] R. Sim and G. Dudek. Learning generative models of scene features. *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 920–929, 2001.

[886] T. Siméon, J.-P. Laumond, and C. Nissoux. Visibility based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6), 2000.

[887] T. Siméon, S. Leroy, and J.-P. Laumond. Path coordination for multiple mobile robots: A resolution complete algorithm. *IEEE Transactions on Robotics & Automation*, 18(1), February 2002.

[888] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, and J. O'Sullivan. A layered architecture for office delivery robots. In *Proceedings First International Conference on Autonomous Agents*, Marina del Rey, CA, 1997.

[889] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

[890] B. Simov, S. M. LaValle, and G. Slutzki. A complete pursuit-evasion algorithm for two pursuers using beam detection. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 618–623, 2002.

[891] B. Simov, G. Slutzki, and S. M. LaValle. Pursuit-evasion using beam detection. In *Proceedings IEEE International Conference on Robotics and Automation*, 2000.

[892] M. Sipser. *Introduction to the Theory of Computation*. PWS, Boston, MA, 1997.

[893] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[894] I. H. Sloan and S. Joe. *Lattice Methods for Multiple Integration*. Oxford Science, Englewood Cliffs, NJ, 1994.

[895] J.-J. E. Slotine and H. S. Yang. Improving the efficiency of time-optimal path following algorithms. *IEEE Transactions on Robotics & Automation*, 5(1):118–124, 1989.

[896] W. D. Smart and L. P. Kaelbling. Practical reinforcement learning in continuous spaces. In *Proceedings International Conference on Machine Learning*, 2000.

[897] D. Smith, J. Frank, and A. Jónsson. Bridging the gap between planning and scheduling. *Knowledge Engineering Review*, 15(1):47–83, 2000.

[898] R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, 1986.

[899] S. J. J. Smith, D. S. Nau, and T. Throop. Computer bridge: A big win for AI planning. *AI Magazine*, 19(2):93–105, 1998.

[900] E. J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 9(2):149–168, 1978.

[901] G. Song and N. M. Amato. Using motion planning to study protein folding pathways. *Journal of Computational Biology*, 26(2):282–304, 2002.

[902] E. Sontag. Gradient technique for systems with no drift: A classical idea revisited. In *Proceedings IEEE Conference Decision & Control*, pages 2706–2711, December 1993.

[903] O. J. Sordalen. Conversion of a car with n trailers into a chained form. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1382–1387, 1993.

[904] P. Souères and J.-D. Boissonnat. Optimal trajectories for nonholonomic mobile robots. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 93–169. Springer-Verlag, Berlin, 1998.

[905] P. Souères and J.-P. Laumond. Shortest paths synthesis for a car-like robot. In *IEEE Transactions on Automatic Control*, pages 672–688, 1996.

[906] R. Spence and S. A. Hutchinson. Dealing with unexpected moving obstacles by integrating potential field planning with inverse dynamics control. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1485–1490, 1992.

[907] M. Spivak. *Differential Geometry*. Publish or Perish, Houston, TX, 1979.

[908] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, New York, 2005.

[909] R. L Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica*, 6:579–589,

1991.

[910] W. Stadler. Fundamentals of multicriteria optimization. In W. Stadler, editor, *Multicriteria Optimization in Engineering and in the Sciences*, pages 1–25. Plenum Press, New York, 1988.

[911] H. Stark and J. W. Woods. *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice-Hall, Englewood Cliffs, NJ, 1986.

[912] L. A. Steen and J. A. Seebach Jr. *Counterexamples in Topology*. Dover, New York, 1996.

[913] R. F. Stengel. *Optimal Control and Estimation*. Dover, New York, 1994.

[914] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3310–3317, 1994.

[915] D. Stewart. A platform with six degrees of freedom. In *Institution of Mechanical Engineers, Proceedings 1965-66, 180 Part 1*, pages 371–386, 1966.

[916] M. Stilman and J. J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. In *Proceedings 2004 IEEE International Conference on Humanoid Robotics (Humanoids '04)*, 2004.

[917] D. Stipanovic, I. Hwang, and C. J. Tomlin. Computation of an over approximation of the backward reachable set using subsystem level set functions, dynamics of continuous, discrete, and impulsive systems. *Series A: Mathematical Analysis*, 11:399– 411, 2004.

[918] P. D. Straffin. Game Theory and Strategy. *Mathematical Association of America*, Washington, DC, 1993.

[919] M. Strandberg. Augmenting RRT-planners with local trees. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3258–3262, 2004.

[920] M. Strandberg. *Robot Path Planning: An Object-Oriented Approach*. PhD thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, 2004.

[921] A. Sudsang, J. Ponce, and N. Srinivasa. Grasping and in-hand manipulation: Geometry and algorithms. *Algorithmica*, 26:466–493, 2000.

[922] A. Sudsang, F. Rothganger, and J. Ponce. Motion planning for disc-shaped robots pushing a polygonal object in the plane. *IEEE Transactions on Robotics & Automation*, 18(4):550–562, 2002.

[923] A. G. Sukharev. Optimal strategies of the search for an extremum. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 11(4), 1971. Translated from Russian, *Zh. Vychisl. Mat. i Mat. Fiz.*, 11, 4 910-924, 1971.

[924] H. Sussmann and G. Tang. Shortest paths for the Reeds-Shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control. Technical Report SYNCON 91-10, Dept. of Mathematics, Rutgers University, Piscataway, NJ, 1991.

[925] H. J. Sussmann. A sufficient condition for local controllability. *SIAM Journal on Control & Optimization*, 16(5):790–802, 1978.

[926] H. J. Sussmann. A general theorem on local controllability. *SIAM Journal on Control & Optimization*, 25(1):158–194, 1987.

[927] H. J. Sussmann. A continuation method for nonholonomic path-finding problems. In *Proceedings IEEE Conference Decision & Control*, pages 2717–2723, December 1993.

[928] H. J. Sussmann. A very non-smooth maximum principle with state constraints. In *Proceedings IEEE Conference Decision & Control*, pages 917–922, December 2005.

[929] K. Sutner and W. Maass. Motion planning among time dependent obstacles. *Acta Informatica*, 26:93–122, 1988.

[930] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[931] I. Suzuki, Y. Tazoe, M. Yamashita, and T. Kameda. Searching a polygonal region from the boundary.

International Journal on Computational Geometry & Applications, 11(5):529–553, 2001.

[932] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, 21(5):863–888, October 1992.

[933] I. Suzuki, M. Yamashita, H. Umemoto, and T. Kameda. Bushiness and a tight worst-case upper bound on the search number of a simple polygon. *Information Processing Letters*, 66:49–52, 1998.

[934] P. Svestka and M. H. Overmars. Coordinated motion planning for multiple carlike robots using probabilistic roadmaps. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1631–1636, 1995.

[935] H. Takeda, C. Facchinetti, and J.-C. Latombe. Planning the motions of a mobile robot in a sensory uncertainty field. *IEEE Transactions Pattern Analysis Machine Intelligence*, 16(10):1002–1017, October 1994.

[936] H. Takeda and J.-C. Latombe. Sensory uncertainty field for mobile robot navigation. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2465–2472, Nice, France, May 1992.

[937] S. Tezuka. *Uniform Random Numbers: Theory and Practice*. Kluwer, Boston, MA, 1995.

[938] S. Tezuka. Quasi-Monte Carlo: The discrepancy between theory and practice. In K.-T. Fang, F. J. Hickernell, and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 124–140. Springer-Verlag, Berlin, 2002.

[939] M. Thorup. Undirected single source shortest paths in linear time. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 12–21, 1997.

[940] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.

[941] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31(5):1–25, April 1998.

[942] S. Thrun,W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[943] B. Tovar, L. Guilamo, and S. M. LaValle. Gap navigation trees: A minimal representation for visibility-based tasks. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, pages 11–26, 2004.

[944] B. Tovar, S. M. LaValle, and R. Murrieta. Locally-optimal navigation in multiply connected environments without geometric maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[945] B. Tovar, S. M. LaValle, and R. Murrieta. Optimal navigation and object finding without geometric maps or localization. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 464– 470, 2003.

[946] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, September 1995.

[947] S. Udupa. *Collision Detection and Avoidance in Computer Controlled Manipulators*. PhD thesis, Dept. of Electrical Engineering, California Institute of Technology, 1977.

[948] University of North Carolina. PQP: A proximity query package. GAMMA Research Group, Available from http://www.cs.unc.edu/~geom/SSV/, 2005.

[949] C. Urmson and R. Simmons. Approaches for heuristically biasing RRT growth. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[950] J. van den Berg and M. Overmars. Roadmap-based motion planing in dynamic environments. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1598–1605, 2004.

[951] J. van den Berg and M. Overmars. Prioritized motion planning for multiple robots. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2217–2222, 2005.

[952] J. G. van der Corput. Verteilungsfunktionen I. *Akademie van Wetenschappen*, 38:813–821, 1935.

[953] D. Vanderpooten. Multiobjective programming: Basic concepts and approaches. In R. Slowinski and J. Teghem, editors, *Stochastic vs. Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty*, pages 7–22. Kluwer, Boston, MA, 1990.

[954] G. Vegter. Computational topology. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 719–742. Chapman and Hall/CRC Press, New York, 2004.

[955] M. Vendittelli and J.-P. Laumond. Visible positions for a car-like robot amidst obstacles. In J.-P. Laumond and M. H. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 213–228. A.K. Peters, Wellesley, MA, 1997.

[956] J. von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.

[957] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, 1944.

[958] G. Walker and D. Walker. *The Official Rock Paper Scissors Strategy Guide*. Fireside, 2004.

[959] X. Wang and F. J. Hickernell. An historical overview of lattice point sets. In K.-T. Fang, F. J. Hickernell, and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 158–167. Springer-Verlag, Berlin, 2002.

[960] F.W.Warner. *Foundations of Differentiable Manifolds and Lie Groups*. Springer-Verlag, Berlin, 1983.

[961] D. S. Watkins. *Fundamentals of Matrix Computations, 2nd Ed.* Wiley, New York, 2002.

[962] G. Weiss, C. Wetzler, and E. von Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 595–601, 1994.

[963] D. Weld. Recent advances in AI planning. *AI Magazine*, 20(2), 1999.

[964] J. T. Wen. Control of nonholonomic systems. In W. S. Levine, editor, *The Control Handbook*, pages 1359–1368. CRC Press, Boca Raton, FL, 1996.

[965] H. Weyl. Über die Gleichverteilung von Zahlen mod Eins. *Mathematische An-nalen*, 77:313–352, 1916.

[966] D. Whitney. Force feedback control of manipulator fine motions. *Transactions of the ASME, Journal of Dynamical Systems*, Measurement, & Control, 99:91–97, 1977.

[967] D. E. Whitney. Real robots don't need jigs. In *Proceedings IEEE International Conference on Robotics & Automation*, 1986.

[968] H. Whitney. Local properties of analytic varieties. In S. Cairns, editor, *Differential and Combinatorial Topology*, pages 205–244. Princeton University Press, Princeton, NJ, 1965.

[969] J. Wiegley, K. Goldberg, M. Peshkin, and M. Brokowski. A complete algorithm for designing passive fences to orient parts. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1133–1139, 1996.

[970] O. Wijk. *Triangulation-Based Fusion of Sonar Data with Application in Mobile Robot Mapping and Localization*. PhD thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, 2001.

[971] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1024–1031, 1999.

[972] R. Wilson, L. Kavraki, J.-C. Latombe, and T. Lozano-Pérez. Two-handed assembly sequencing. *International Journal of Robotics Research*, 14(4):335–350, 1995.

[973] R. H.Wilson. *On Geometric Assembly Planning*. PhD thesis, Stanford University, Stanford, CA, March 1992.

[974] R. H. Wilson and J.-C. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence Journal*, 71(2):371–396, 1994.

[975] P. H. Winston. *Artificial Intelligence*. Addison-Wesley, Reading, MA, 1992.

[976] S. C. Wong, L. Middleton, and B. A. MacDonald. Performance metrics for robot coverage tasks. In *Proceedings Australasian Conference on Robotics and Automation*, 2002.

[977] C. F. J. Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

[978] L. Xu and M. I. Jordan. On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, 8:129–151, 1996.

[979] J. Yakey, S. M. LaValle, and L. E. Kavraki. Randomized path planning for linkages with closed kinematic chains. *IEEE Transactions on Robotics and Automation*, 17(6):951–958, December 2001.

[980] K. Yamane, J. J. Kuffner, and J. K. Hodgins. Synthesizing animations of human manipulation tasks. In *Proceedings ACM SIGGRAPH*, 2004.

[981] M. Yamashita, H. Umemoto, I. Suzuki, and T. Kameda. Searching for a mobile intruder in a polygonal region by a group of mobile searchers. *Algorithmica*, 31:208–236, 2001.

[982] B. Yamauchi, A. Schultz, and W. Adams. Mobile robot exploration and map building with continuous localization. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3715–3720, 2002.

[983] L. Yang and S. M. LaValle. The sampling-based neighborhood graph: A framework for planning and executing feedback motion strategies. *IEEE Transactions on Robotics and Automation*, 20(3):419–432, June 2004.

[984] Q. Yang. *Intelligent Planning*. Springer-Verlag, Berlin, 1997.

[985] Y. Yavin and M. Pachter. *Pursuit-Evasion Differential* Games. Pergamon, Oxford, U.K., 1987.

[986] A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *Proceedings IEEE International Conference on Robotics and Automation*, 2005.

[987] A. Yershova and S. M. LaValle. Deterministic sampling methods for spheres and SO(3). In *Proceedings IEEE International Conference on Robotics and Automation*, 2004.

[988] A. Yershova, B. Tovar, R. Ghrist, and S. M. LaValle. Bitbots: Simple robots solving complex tasks. In *AAAI National Conference on Artificial Intelligence*, 2005.

[989] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.

[990] M. Yim. *Locomotion with a Unit-Modular Reconfigurable Robot*. PhD thesis, Stanford University, Stanford, CA, December 1994. Stanford Technical Report STAN-CS-94-1536.

[991] J. Yong. On differential evasion games. *SIAM Journal on Control & Optimization*, 26(1):1–22, January 1988.

[992] J. Yong. On differential pursuit games. *SIAM Journal on Control & Optimization*, 26(2):478–495, March 1988.

[993] J. Yong. A zero-sum differential game in a finite duration with switching strategies. *SIAM Journal on Control & Optimization*, 28(5):1234–1250, September 1990.

[994] T. Yoshikawa. *Foundations of Robotics: Analysis and Control*. MIT Press, Cambridge, MA, 1990.

[995] Y. Yu and K. Gupta. On sensor-based roadmap: A framework for motion planning for a manipulator arm in unknown environments. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1919–1924, 1998.

[996] J. Zabczyk. Some comments on stabilizability. *Applied Mathematics and Optimization*, 19(1):1–9, 1989.

[997] L. S. Zaremba. Differential games reducible to optimal control problems. In *Proceedings IEEE Conference Decision & Control*, pages 2449–2450, Tampa, FL, December 1989.

[998] M. Zefran and J. Burdick. Stabilization of systems with changing dynamics by means of switching. In

Proceedings IEEE International Conference on Robotics & Automation, pages 1090–1095, 1998.

[999] M. Zefran, J. Desai, and V. Kumar. Continuous motion plans for robotic systems with changing dynamic behavior. In *Proceedings IEEE International Conference on Robotics & Automation*, 1996.

[1000] M. I. Zelikin and V. F. Borisov. *Theory of Chattering Control*. Birkhäuser, Boston, MA, 1994.

[1001] M. Zhang, R. A. White, L. Wang, R. N. Goldman, L. E. Kavraki, and B. Hassett. Improving conformational searches by geometric screening. *Bioinformatics*, 21(5):624 – 630, 2005.

[1002] N. L. Zhang and W. Zhang. Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 14:29–51, 2001.

[1003] R. Zhou and E. A. Hansen. An improved grid-based approximation algorithm for POMDPs. In *Proceedings International Joint Conference on Artificial Intelligence*, 2001.

[1004] Y. Zhou and G. S. Chirikjian. Probabilistic models of dead-reckoning error in nonholonomic mobile robots. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1594–1599, 2003.

[1005] S. Zionts. Multiple criteria mathematical programming: An overview and several approaches. In P. *Serafini, editor, Mathematics of Multi-Objective Optimization*, pages 227–273. Springer-Verlag, Berlin, 1985.

英汉词汇对照表

C_0 function——C_0 函数
C_∞ function——C_∞ 函数
C_∞ manifold——C_∞ 流形
C_∞ structure——C_∞ 结构
C_k function——C_k 函数
L_1 metric——L_1 度量
L_2 metric——L_2 度量
L_∞ metric——L_∞ 度量
L_p metric——L_p 度量
L_p norm——L_p 范数
k-cell——k -胞腔
k -neighborhood k-邻域
(t,m,s) -nets——(t,m,s) -要点
(t,s) -sequences——(t,s) -顺序
1-complex——1-复形
1-neighborhood——1-邻域
2-neighborhood——2-邻域
3D triangles 三维三角形

A

A^*algorithm——A^*算法
Abelian group——Abelian 群
acceleration vector 加速度向量
acceleration-based control 基于加速度的控制
accelerometer 加速度计
accessibility (of a roadmap)（路线图的）可达性
accessible system 可达系统
accumulation point 聚点
action history 行动历史
action sequence 行动序列
action trajectory 行动轨迹
active localization problem 主动定位问题
active-passive decomposition 主动-被动分解
actuators 执行机构
Adams methods——Adams 方法
adjoint transition equation 伴随转移方程
adjoint variables 伴随变量
admissible configurations 可容许的位形
affine space 仿射空间
affine-in-control system 仿射控制系统
AGV(automated guided vehicle) 自动导引车辆
airport terminal 机场终端
algebraic primitive 代数本原
algebraic Riccati equation 代数 Riccati 方程
algebraic set 代数集
algebraic variety 代数簇
Allen wrench 通用板手
Alpha Puzzle——α 难题
Alphabet 字母表
ambient isotopy 合伦
ambient space 空间
analytic function 解析函数
angular momentum 角动量
angular velocity 角速度
annihilator 零化子
antipodal points 对径点
approximate cell decomposition 近似胞腔剖分
approximate cover 近似覆盖
approximate optimal motion planning 近似最优运动规划
Ariadne's Clew algorithm——Ariadne 线索算法
asymptotic convergence to a goal 渐近收敛到目标
asymptotic solution plan 渐近解规划
asymptotic stability 渐近稳定性
atlas 图册
automated farming 自动化耕作

automated guided vehicles 自动导引车辆
autonomous differential equations 自治微分方程
average cost-per-stage model 每阶段平均代价模型
average dispersion 平均离散度
averaging methods 平均方法
axioms of rationality 合理性公理
axis-aligned bounding box 轴向包围盒

B

B-splines——B-样条
Backprojection 后向射影
backward action space 后向行动空间
backward reachable set 后向可达集
backward search 后向搜索
backward state transition equation 后向状态转移方程
backward system simulator 后向系统模拟器
backward value iteration 后向值迭代
bang-bang approach——bang-bang 方法
Barraquand-Latombe nonholonomic planner——Barraquand-Latombener 非完整规划器
base point (on a manifold)（流形上的）基点
base point of a path 路径的基点
Bayes' rule 贝叶斯准则
Bayesian classifier 贝叶斯分类器
behavioral strategies 行为策略
best-first search 最好优先搜索
bidirectional search 双向搜索
bijective sensor 双射传感器
bilinear programming 双线性规划
binding constraints 绑定约束
bitangent line 双切线
bitangent ray 双切射线
bitmap 位图
black-box simulators 黑箱模拟器
body density 体密度
body frame 体坐标
bond angle 键角
bond length 键长
Borel sets 波雷尔集合
boundary grid point 边界方格点
boundary point 边界点
boundary representation 边界表示
boundary sensors 边界传感器
bounded set 有界集
bounded-acceleration model 有界加速度模型
bounded-velocity model 有界速度模型
Brachistochrone curve 最速降线曲线
breadth-first search 广度优先搜索
Brockett's condition——Brockett 条件
Brockett's system——Brockett 系统
bug algorithms——bug 算法
bug trap——bug 陷阱
Bug1 strategy——Bug1 策略
Bug2 strategy——Bug2 策略
BVP 两点边值问题

C

calculus of variations 变分法
candidate Lyapunov function 候选 Lyapunov 函数
Canny's roadmap algorithm——Canny 路线图算法
car pulling trailers 牵引拖车的汽车
Cartesian product 笛卡儿积
carton folding 纸盒折叠
causal links 因果联系
cell decomposition 胞腔剖分
center of mass 质心
Central Limit Theorem 中心极限定理
chain of integrators 积分器的链
chained-form system 链式系统
Chen-Fliess series——Chen-Fliess 级数
chi-square test 卡方检验
classification rule 分类法则
classifier 分类器
closed kinematic chains 闭运动链

closed set 闭集
closed system (in mechanics)（力学中的）封闭系统
closed-loop 闭环
closure of a set 集合的闭包
closure space 闭包空间
Collins decomposition——Collins 剖分
collision-detection 碰撞检测
combinatorial motion planning 组合运动规划
combinatorial roadmaps 组合路线图
commutative group 交换群
commutative ring 可换环
commutator 换向器
compass 罗盘
compatible coordinate neighborhoods 相容坐标邻域
competitive ratio 竞争率
completely integrable 完全可积的
completeness 完备性、完整性
complex 复形
complexity of motion planning 运动规划的复杂度
compliant motions 适应性运动
compressed mode 压缩模式
computational algebraic geometry 计算代数几何
Conchoid of Nicomedes——Nicomedes 螺旋线
conditional Bayes' risk 条件贝叶斯风险
conditional Bayes' rule 条件贝叶斯定理
conditional expectation 条件均值
conditional independence 条件独立
conditional probability 条件概率
configuration space 位形空间、构形空间
connectivity-preserving roadmap 保护连通性的路线图
conservative approximations 保守近似
conservative system 保守系统
constant vector field 常数向量场
constant-sum game 常数和对策
contaminated region 受污染区域
continuous function 连续函数
continuous-steering car 连续转向汽车
contractible space 可缩空间
control system 控制系统
control-affine system 仿射控制系统
controllability matrix 可控矩阵
controlled Markov process 可控的马尔可夫过程
convex hull 凸壳、凸包
convex polygon 凸多边形
convex set 凸集合
convolution 卷积
cooperative game theory 合作对策理论
coordinate neighborhood 坐标邻域
coordination space 坐标空间
Coriolis matrix 科里奥利矩阵
cost-to-come 已付代价
cost-to-go 尚需代价
Coulomb friction 库仑摩擦
covariance matrix 协方差矩阵
cover of a set 集合的覆盖
critical curves 临界曲线
critical gap events 临界间隙事件
critical point of a function 函数的临界点
cube complex 立方复形
cubical partition 立方分拆
cycloid function 摆线函数
cylindrical algebraic decomposition 圆柱代数剖分
cylindrical decomposition 圆柱剖分
cylindrical joint 圆柱关节

D

D^*algorithm—D^*算法
Davis-Putnam procedure—Davis-Putnam 过程
decision maker 决策者
decision problem 决策问题
decision theory 决策论
decision vertex (in a game tree)（对策树中的）决策顶点
decoupled planning 解耦规划
decoupling vector fields 解耦向量场
deformation retract 形变收缩核

degrees of freedom 自由度
delayed-observation sensor 延迟观测传感器
dense sequence 稠密序列
dense set 稠密集合
depth-first search 深度优先搜索
depth-mapping sensors 深度映射传感器
derived information space 衍生信息空间
derived information transition equation 衍生信息转移方程
deterministic finite automaton 确定的有限自动机
dielectric constant 介电常数
differentiable manifold 微分流形
differentiable structure 微分结构
differential constraints 微分约束
differential drive 差动机构
differential game 微分对策
differential inclusion 微分包含
differential models 微分模型
Dirichlet boundary condition—Dirichlet 有界条件
discount factor 折扣因子
discounted cost model 折扣代价模型
discrepancy 偏差
discrete feasible planning 离散可行规划
discrete-time model 离散时间模型
dispersion 离散度
distance function 距离函数
distribution (of vector fields)（向量场的）分布
disturbed odd/even sensor 受扰的奇偶传感器
disturbed sign sensor 受扰的符号传感器
domain of attraction 吸引域
drift 漂移
driftless system 无漂移的系统
dynamic constraints 动态约束
dynamic programming 动态规划

E

embedding of a manifold 流形的嵌入
energy function 能量函数
Euclidean metric 欧几里得度量
Euclidean motion model 欧几里得运动模型
Euclidean norm 欧几里得范数
Euclidean shortest paths 欧几里得最短路径
Euler angles 欧拉角
Euler integration 欧拉积分
Euler-Lagrange equation 欧拉-拉格朗日方程
expansive-space planner 扩展空间的规划器
expectation of a random variable 随机变量的期望
exponentially stable system 指数稳定系统
extended Kalman filter 扩展卡尔曼滤波器
extended system 扩展系统
exterior point 外点
extremal function 极值函数

F

fast Fourier transforms 快速傅里叶变换
feasible planning 可行规划
feasible space 可行空间
feature space 特征空间
feedback control law 反馈控制率
feedback motion planning 反馈运动规划
feedback planning 反馈规划
feedback policy 反馈策略
feedback stabilization 反馈稳定
finite state machine 有限状态机
first-order controllable systems 一阶可控系统
fixed point of a vector field 向量场的不动点
fixed-path coordination 固定路径坐标
fixed-roadmap coordination 固定路线图坐标
flexible materials 柔性材料
folding problems 折叠问题
force sensor 力传感器
formal Lie algebra 李代数
forward projection 前向射影
differential 微分的
nondeterministic 非确定的
probabilistic 概率的

forward search 前向搜索
A^*algorithm—A^*算法
best first 最好优先
breadth-first 广度优先
depth-first 深度优先
forward value iteration 前向值迭代
four-bar mechanism 四杆机构
free space 自由空间
free variables 自由变量
frequentist 频率论者
frequentist risk 频率论的风险
friction cone 摩擦锥
frontier set 边界集
fully actuated system 全驱动系统
function space 函数空间
functional 泛函
fundamental group 基本群

G

gain constant 增益常数
game 对策、博弈
game against nature 与大自然的对策
game graph 对策图
game theory 对策论
game tree 对策树
gap navigation tree 间隙导航树
gap sensor 间隙传感器
gap theorems 间隙定理
Gaussian sampling 高斯采样
Geiger counter sensor 盖格计数器传感器
general linear group 一般线性群
general position 广义位置
generalized coordinates 广义坐标
generalized cylinder 广义圆柱体
generalized damper model 广义阻尼模型
generalized forces 广义力
generalized momentum 广义动量
generalized Voronoi diagram 广义Voronoi 图
geodesics 大地测量学
globally asymptotically stable 全局渐近稳定
globally positive definite 全局正定
globally randomized plan 全局随机化的规划
goal recognizability 目标可辨识性
goal sensor 目标传感器
gradient descent 梯度下降
grid point 方格点
grid resolution 方格分辨率
group 群
group axioms 群公理
group of n-dimensional rotation matrices—n 维旋转矩阵的群
gyroscope 陀螺

H

Haar measure—Haar 测度
half-edge 半边
half-plane 半平面
half-space 半空间
Hamilton's equations 哈密顿方程
Hamiltonian function 哈密顿函数
Hamiltonian mechanics 哈密顿力学
Hammersley point set—Hammersley 点集
harmonic potential function 调和势函数
Hausdorff axiom—Hausdorff 公理
Hausdorff space—Hausdorff 空间
Heisenberg system—Heisenberg 系统
hierarchical planning 分层规划
higher order controllability 高阶可控性
Hilbert space—Hilbert 空间
hill function—hill 函数
history information space 历史信息空间
history information state 历史信息状态
history-based sensor mapping 基于历史的传感器映射
holonomic 完整的

homeomorphic spaces 完整空间
homeomorphism 同胚
homogeneous transformation matrix 相似变换矩阵
homology 同调
homotopic paths 同伦路径
homotopy group 同伦群
hybrid state space 组合状态空间
hybrid system 混合系统

I

I-map—I-映射
I-space—I-空间
I-state—I-状态
ideal distance function 理想距离函数
implicit function theorem 隐函数定理
implicit velocity constraints 隐含速度约束
incremental distance computation 递增距离计算
incremental sampling and searching 递增采样和搜索
inertia tensor 惯性张量
inertial coordinate frame 惯性坐标系
Infimum 下确界
infinite-horizon problem 无限范围问题
information mapping 信息映射
information space 信息空间
information transition equation 信息转移方程
information transition function 信息转移函数
information-conservative property 信息守恒特性
information-feedback plan 信息反馈规划
initial condition space 初始条件空间
integrable system 可积系统
integral curve 积分曲线
integral manifold 可积流形
interior of a set 集合的内部
interior point 内点
interpolation neighbors 插值邻域
interpolation region (for value iteration)（值迭代的）插值区域
interval homeomorphisms 区间同胚
inverse control problem 逆控制问题
inverse kinematics problem 逆运动学问题
involution 对合
isomorphic graphs 同构图
isomorphic groups 同构群
isomorphism 同构

J

Jacobi identity 雅可比恒等式
Jacobian 雅可比
jerk (third time derivative)对时间的三阶导数

K

Kalman filter 卡尔曼滤波器
kinematic chain 运动链
kinematic constraints 运动约束
kinematic singularities 运动奇异性
kinematically controllable 运动学上可控的
kinetic energy 动能
Klein bottle—Klein 瓶
Knot 节
Kutzbach criterion—Kutzbach 准则

L

Lagrange multiplier 拉格朗日乘子
Lagrangian function 拉格朗日函数
Lagrangian mechanics 拉格朗日力学
landmark region detector 地标区域监测器
landmark sensors 地标传感器
layered graph 分层图
layered plan 分层规划
Lebesgue integral—Lebesgue 积分
Lebesgue measure—Lebesgue 测度
left-invariant vector field—左不变向量场
Legendre transformation—Legendre 变换
Lens spaces 透镜空间
level-set method 水平集方法

Lie algebra 李代数
Lie algebra rank condition 李代数秩条件
Lie bracket 李括号
Lie derivative 李导数
Lie group 李群
limit curve 极限曲线
limit cycle 极限环
limit point of a set 集合的极限点
line-segment robot 线段机器人
linear combination 线性组合
linear differential game 线性微分对策
linear interpolation 线性插值
linear momentum 线动量
linear programming 线性规划
linear sensing models 线性感测模型
linear space 线性空间
linear system 线性系统
linear transformations 线性变换
linear-Gaussian system 线性高斯系统
linear-quadratic problems 线性二次型问题
linear-quadratic-Gaussian (LQG) system 线性二次型高斯系统
link 连杆
linkage 连杆组
linkage graph 连杆组的图
Lipschitz condition—Lipschitz 条件
Lipschitz constant—Lipschitz 常数
local operator 局部算子
local planning method 局部规划方法
local visibility sensor 局部可视传感器
localization 定位
locally positive definite 局部正定
locally randomized plan 局部随机规划
logic-based planning 逻辑规划
logical predicate 局部预测
low-discrepancy sampling 低离散度采样
low-dispersion sampling 低偏差采样
lower value of a game 对策的下值
lunar lander 月球车
Lyapunov function—Lyapunov 函数
Lyapunov stability—Lyapunov 稳定

M

Mahalanobis metric—Mahalanobis 度量
maneuver 机动
maneuver automaton 机动自动机
Manhattan metric—Manhattan 度量
Manhattan motion model—Manhattan 运动模型
manifold 流形
manipulation graph 操作图
manipulation planning 操作规划
nonprehensile 非紧握的
manipulator 操作臂
Markov chain 马尔可夫链
Markov decision process 马尔可夫决策过程
Markov game 马尔可夫对策
Markov process 马尔可夫过程
mass matrix 质量矩阵
matrix game 矩阵对策
matrix groups 矩阵群
matrix subgroup 矩阵子群
maximum-clearance navigation function 最大间隙导航函数
maximum-clearance roadmap 最大间隙路线图
maze searching 迷宫搜索
measurable function 可测函数
measurable sets 可测集
measure axioms 测度公理
measure space 测度空间
measure theory 测度论
measure zero 零测度
mechanics 力学
    Hamiltonian 哈密顿
    Lagrangian 拉格朗日
    Newton-Euler 牛顿-欧拉
metric space 度量空间
metric tensor 度量张量

minimum turning radius 最小转弯半径
Minkowski difference—Minkowski 差分
Minkowski sum—Minkowski 和
mixed Nash equilibrium 混合 Nash 平衡点
mode space 模式空间
mode transition function 模式转换函数
mode-dependent dynamics 依赖模式的动力学
moment of a density 密度的矩
moment of force 力矩
moment of inertia 转动惯量
moment of momentum 动量矩
moment-based approximations 基于矩的近似
momentum 动量
monomial 单项式的
monotone polygon 单调多边形
Monte-Carlo localization 蒙特卡罗定位
Morse function 摩尔斯函数
Morse theory 摩尔斯理论
motion command 运动命令
motion planning 运动规划
motion primitive 运动本原
multi-body dynamics 多体动力学
multi-chained-form systems 多链式系统
multi-level approach 多层方法
multi-linear interpolation 多线性插值
multi-resolution grid 多分辨率方格
multiobjective optimization 多目标优化
multiple query 多疑问
multiple-robot motion planning 多机器人运动规划
multiple-robot optimality 多机器人优化
multiply connected 多连通的
mutex condition 互斥条件
mutex relation 互斥关系

N

Nash equilibrium 纳什平衡点
Nonuniqueness 非唯一性
Nature 大自然
nature action space 大自然行动空间
nature observation action 大自然观测行动
nature sensing action 大自然感测行动
navigation function 导航函数
    continuous space 连续空间
navigation problem 导航问题
negative literal 负文字
neighborhood function 邻域函数
neighborhood of a cover 一个覆盖的邻域
Neumann boundary condition—Neumann 边界条件
neuro-dynamic programming 神经动力学规划
Newton's laws 牛顿定律
Newton-Euler mechanics 牛顿-欧拉力学
nilpotent 幂零
nilpotent system 幂零系统
nonconservative forces 非保守力
nonconvex 非凸的
noncooperative game 非合作对策
noncritical regions 非临界区域
nondeterministic finite automaton 非确定的有限自动机
nondeterministic information space 非确定的信息空间
nondeterministic Turing machine 非确定的图灵机
nondeterministic uncertainty 非确定的不确定性
nondirectional backprojections 无方向的后向射影
nondominated 非优超的
nonholonomic 非完整的
nonholonomic constraints 非完整约束
nonholonomic integrator 非完整积分器
nonholonomic metric 非完整度量
nonholonomic planning 非完整规划
nonholonomic system 非完整系统
nonholonomic system theory 非完整系统理论
noninformative prior 非信息的先验信息
nonintegrable 不可积的
nonlinear optimization 非线性优化
nonlinear programming 非线性规划
nonlinear system 非线性系统

affine-in-control 控制中的仿射
nonparametric methods 非参数法
nonpositively curved space 非正曲线空间
nonrigid transformations 非刚体变换
nonzero-sum game 非零和对策
NP (complexity class) NP（复杂度类型）
null sensor 零化传感器
numerical continuation 数值连续统
numerical integration 数值积分
Euler 欧拉
multistep methods 多步法
Runge-Kutta 龙格-库塔
single-step methods 单步法

O

observability 可观性
observation space 观测空间
obstacle region 障碍区域
odd/even sensor 奇偶传感器
odometric coordinates 里程表坐标
odometry sensors 里程表传感器
on-line algorithm 在线算法
open set 开集合
open-loop 开环
operator 算子
optical character recognition 光学字符识别
optimal motion planning 最优运动规划
optimal planning 最优规划
optimization 优化
orientation sensor 方位传感器
orienteering problem 越野识图问题
origami 手工折纸
orthogonal group 正交群

P

parallel manipulator 并行操作臂
parameter estimation 参数估计
parameterization 参数化
Pareto optimal—Pareto 最优
part configuration space 部分位形空间
partial grid 部分方格
partial plan 部分规划
partially observable Markov decision process 部分可观测的马尔可夫过程
particle filtering 粒子滤波
path 路径
path connected 连通路径
path tuning 路径调节
path-constrained phase space 路径受限的相空间
path-directed subdivision tree 有向路径细分树
pattern classification 模式分类
pendulum 钟摆
permissible action trajectories 允许的行动轨迹
pharmacophore 药效团
phase constraints 相约束
phase space 相空间
phase transition equation 相空间转移方程
phase vector 相向量
Philip Hall basis—Philip Hall 基
Piano Mover's Problem 钢琴搬运工问题
piecewise-linear obstacle motion 分段线性的障碍物运动
pitch rotation 俯仰旋转
plan-and-transform method 规划-变换方法
plan-based state transition graph 基于规划的状态转移图
planar joint 平面关节
plane-sweep principle 平面扫描原理
radial sweep 径向扫描
planetary navigation 星际导航
planner 规划器
planning graph 规划图
planning under sensing uncertainty 感测不确定下的规划
point robot 点机器人
point-location problem 点定位问题

policy iteration 策略迭代
polygonal model 多边形模型
polyhedral model 多面体模型
polynomial 多项式
Pontryagin's minimum principle 庞特里亚金最小值原理
position sensor 位置传感器
positive definite function 正定函数
positive literal 正文字
possibilistic uncertainty 概率不确定性
potential energy 势能
potential function 势函数
    attractive term 吸引项
    repulsive term 排斥项
preimage of a function 函数的原像
preimage of a motion command 运动命令的原像
preimage of an observation 一个观测的原像
preimage planning 原像规划
principle of least action 最小作用原理
principle of optimality 最优性原理
prior distribution 先验分布
prioritized planning 优先规划
prismatic joint 棱柱形关节
probabilistic completeness 概率完备性
probabilistic information space 概率信息空间
probabilistic roadmap 随机路线图
probabilistic uncertainty 概率不确定性
probability function 概率函数
probability measure 概率测度
probability space 概率空间
probability theory 概率论
product of inertia 惯量积
projective geometry 射影几何
projective space 射影空间
protein cavity 蛋白质腔
proximity sensor 接近传感器
pseudometric 伪度量
pseudorandom number generation 伪随机数发生器
pursuit-evasion game 追逃对策

Q

Q-factor—Q 因子
Q-learning—Q 学习
quadratic cost functional 二次型代价泛函
quadratic potential function 二次型势函数
quantified variables 量化变量
quantifier 量词
quantifier-elimination problem 量词消除问题
quantifier-free formula 无量词公式
quasi-static 准静态
quaternion 四元数
quotient topology 商拓扑

R

radial sweep 径向扫描
random loop generator 随机环路发生器
random sampling 随机采样
random variable 随机变量
random-walk planner 随机游走规划器
randomized algorithm 随机化算法
randomized lower value 随机化的下值
randomized plan 随机化的规划
randomized potential field 随机化的势场
randomized saddle point 随机化的鞍点
randomized security plan 随机化的安全规划
randomized strategy 随机化的策略
randomized upper value 随机化的上值
range scanner 扫描测距仪
rapidly exploring dense tree 快速探索稠密树
rapidly exploring random tree 快速探索随机树
reachability graph 可达性的图
reachability tree 可达性的树
reachable set 可达集
reactive plan 反应规划
real algebraic numbers 实代数数
recognizability 可辨识性
recontamination 再污染

Reeds-Shepp car—Reeds-Shepp 汽车
Reeds-Shepp curves—Reeds-Shepp 曲线
refinement of a plan 规划的细化
reflex vertex 反射顶点
reinforcement learning 强化学习
repulsive vertex 排斥性的顶点
resolution 分辨率
resolution completeness 分辨率完备性
resultant 合向量
revolute joint 旋转关节
reward 收益
reward function 收益函数
reward functional 收益泛函
reward space 收益空间
Riemannian manifold 黎曼流形
Riemannian metric 黎曼度量
Riemannian tensor 黎曼张量
rigid-body dynamics 刚体动力学
rigid-body transformations 刚体变换
robot displacement metric 机器人位移度量
robot-robot collisions 机器人-机器人碰撞
roll rotation 滚动旋转

S

saddle point 鞍点
sample sequence 采样序列
sample set 采样集合
sample space (of a probability space) 概率空间的采样空间
sampling-based neighborhood graph 基于采样的邻域图
sampling-based planning 基于采样的规划
sampling-based roadmap 基于采样的路线图
Sard's Theorem—Sard 定理
second-order controllable systems 二阶可控系统
second-order differential drive 二阶差动机构
second-order unicycle 二阶独轮车
security plan 安全规划
security strategy 安全策略
semi-algebraic decomposition 半代数剖分
semi-algebraic model 半代数模型
semi-algebraic set 半代数集合
sensing history 感测历史
sensor feedback 传感器反馈
sensor mapping 传感器映射
sensor observation 传感器观测
sensor-based planning 基于传感器的规划
sensorless manipulation 无传感器的操作臂
sensorless planning 无传感器的规划
shooting methods 打靶法
shortest-path functional 最短路径泛函
shortest-path roadmap 最短路径路线图
sigma algebra—sigma 代数
sign sensor 符号传感器
sign-invariant region 符号不变区域
silhouette curves 轮廓曲线
simple polygon 简单多边形
simple-car model 简易汽车模型
simple-unicycle model 简易独轮车模型
simply connected space 单连通空间
simulation-based dynamic programming 基于模拟的动态规划
simulation-based methods 基于模拟的方法
simulation-based planning 基于模拟的规划
singular 0-simplex 奇异 0-单形
singular 1-simplex 奇异 1-单形
singular k-simplex 奇异 k-单形
singular arcs 奇异弧线
singular complex 奇异复形
singular distribution 奇异分布
singular matrix 奇异矩阵
singular point of a distribution 一个分布的奇异点
singular simplex 奇异单形
singular value decomposition (SVD)奇异值分解
skew symmetry 反对称
sliding-mode control 滑模控制
small-time local controllability 短时局部可控性

smooth differential drive 平滑差动机构
smooth distribution 平滑分布
smooth function 平滑函数
smooth manifold 平滑流形
smooth structure 平滑结构
smoothness of a function 函数的平滑性
solution trajectory 解轨迹
spanning tree 生成树
spanning-tree covering 生成树覆盖
spatial constraints 空间约束
special Euclidean group 特殊欧几里得群
special orthogonal group 特殊正交群
spherical coordinates 球面坐标
spherical joint 球形关节
spherical linear interpolation 球形线性插值
spiral search 螺旋搜索
stable configuration space 稳定位形空间
standard grid 标准方格
star-shaped regions 星形区域
state estimation 状态估计
state history 状态历史
state mapping 状态映射
state space 状态空间
state trajectory 状态轨迹
state transition equation 状态转移方程
state transition function 状态转移函数
state transition graph 状态转移图
state transition matrix 状态转移矩阵
state-nature mapping 状态-大自然映射
state-sensor mapping 状态-传感器映射
state-space discretization 状态空间离散化
stationary cost-to-go function 平稳尚需代价函数
statistical decision theory 统计决策理论
steering methods 操控方法
stereographic projection 球极投影
stochastic control theory 随机控制理论
stochastic differential equation 随机微分方程
stochastic fractal 随机分形
stochastic iterative algorithm 随机迭代算法
stochastic shortest-path problem 随机最短路径问题
strong backprojection 强后向射影
sub-Riemannian metric 子黎曼度量
subjective probabilities 主观概率，满射概率
subspace topology 子空间拓扑
sufficient information mapping 充分信息映射
Superquadric 超二次曲面的
supremum 上确界
swath 刈幅
switching boundary 切换边界
switching time 切换时间
symmetric systems 对称系统
symmetric Turing machine 对称图灵机
symmetry class 对称类
symplectic manifold 辛流形

T

tangent bundle 切丛
tangent point 切点
tangent space 切空间
Taylor series 泰勒级数
temporal difference 时间差分
temporal logic 时间逻辑
termination action 终止行动
theory of computation 计算理论
time scaling 时标
time-invariant 时不变的
time-limited reachable set 时间受限可达集
time-monotonic path 时间单调路径
time-optimal trajectory planning 最优时间轨迹规划
time-varying motion planning 时变运动规划
topological complexity 拓扑复杂性
topological graph 拓扑图
topological manifold 拓扑流形
topological property 拓扑特性
topological space 拓扑空间
torque 转矩
trajectory optimization 轨迹优化

trajectory planning 轨迹规划
transformations 变换
transition configurations 位形转换
trapezoidal decomposition 梯形剖分
Traveling Salesman Problem 旅行商问题
triangle inequality 三角不等式
triangulation 三角剖分
trim trajectory 平衡轨迹
trivial operator 平凡算子
trivial topology 平凡拓扑
Turing machine 图灵机
two-point boundary value problem 两点边值问题

U

underactuated system 欠驱动系统
unique point 单点
unit complex number 单位复数
unit quaternions 单位四元数
upper value of a game 对策的上值
utility function 效用函数
utility theory 效用理论

V

value iteration 值迭代
Van der Corput sequence—Van der Corput 序列
variation of a function 函数的变分
vector field 向量场
vector space 向量空间
velocity field 速度场
velocity-tuning method 速度调节方法
vertex selection method 顶点选择方法
vertical decomposition 垂直剖分
violation-free state 无冲突的状态
virtual human 虚拟人
visibility graph 可视图
visibility polygon 可视多边形
visibility region 可视区域
visibility roadmap 可视路线图
visibility sensor 可视传感器
visibility-based pursuit-evasion 基于可视性的追逃
Voronoi diagram—Voronoi 图
Voronoi region—Voronoi 区域
Voronoi vertex—Voronoi 顶点

W

wavefront 波前
wavefront propagation 波前传播
wavelet 小波
weak backprojection 弱后向射影
weighted-region problem 区域加权问题
Weiner process 维纳过程
Whitney's embedding theorem—Whitney 嵌入定理
with probability one 以概率 1
word (sequence of motion primitives)字（运动本原序列）
worst-case analysis 最坏情况分析

Y

yaw rotation 航向旋转

Z

zero-sum game 零和对策