

# 硕士学位论文

## 基于 D\* Lite 算法的移动机器人路径规划 研究

### RESEARCH ON MOBILE ROBOT PATH-PLANNING BASED ON D\* LITE ALGORITHM

徐开放

哈尔滨工业大学  
2016 年 12 月

国内图书分类号: TN911.73  
国际图书分类号: 621.3

学校代码: 10213  
密级: 公开

工程硕士学位论文

基于 D\* Lite 算法的移动机器人路径规划  
研究

硕 士 研 究 生: 徐开放

导 师: 张乃通 教授

申 请 学 位: 工程硕士

学 科: 电子与通信工程

所 在 单 位: 深圳研究生院

答 辩 日 期: 2016 年 12 月

授予学位单位: 哈尔滨工业大学

Classified Index: TN911.73

U.D.C: 621.3

Dissertation for the Master's Degree in Engineering

**RESEARCH ON MOBILE ROBOT  
PATH-PLANNING BASED ON D\* LITE  
ALGORITHM**

<b>Candidate:</b>	Xu Kaifang
<b>Supervisor:</b>	Prof. Zhang Naitong
<b>Academic Degree Applied for:</b>	Master's of Engineering
<b>Speciality:</b>	Electronic and Communication Engineering
<b>Affiliation:</b>	Shenzhen Graduate School
<b>Date of Defense:</b>	December , 2016
<b>Degree-Conferring-Institution:</b>	Harbin Institute of Technology

## 摘 要

路径规划作为移动机器人导航系统的核心,在生活服务导航,外星球探索、无人驾驶、水下探索等诸多领域有着不可或缺的作用,解决了在已知起点和终点的情况下“怎么去”的问题。路径规划经历了从环境已知到环境未知、简单环境到复杂环境、小地图到大地图、简单人工智能向高级人工智能的发展。目前路径规划领域中针对未知环境下的移动机器人路径规划的研究尚未形成体系。尤其在大地图和复杂环境下,传统人工智能路径规划方法采用高分辨率地图表示环境,规划存在着盲目性、复杂性以及规划耗时等缺点。本文就如何在保证规划效果相近的前提下,优化环境地图空间表示、提高规划灵活性和高效性以减少路径规划花费时间进行研究。

传统的路径规划方法在解决大地图环境路径规划时,通常采用大地图细粒化的方案,这样带来的弊端就是在未知环境下的重规划次数较多,而且随着栅格增多,规划算法复杂度呈指数级增大,规划十分耗时。除此之外,在环境比较复杂的情况下,传统规划方法因不能根据环境特点做出理性判断而导致其效率低下。本文对经典的路径规划方法进行了详细分析,着重分析各个算法的优缺点以及各自较为适用的特定场景。最终针对本文研究的复杂大地图下路径规划提出了一种融合算法,即将地图分层规划,细化规划方式以做到集合多种算法对于全局采用性能极好的  $D^*$  Lite 算法,对于局部采用具有环境自主学习能力的增强神经网络算法。事实上,采用多种算法融合的方法来解决特定环境下的路径规划具有一定的合理性。它可以集多种算法的优点,针对不同的特定环境下的路径规划做到局部最优,进而达到算法间优势互补的效果。

本文采用微软公司推出的微软机器人开发者平台 (Microsoft Robotics Developer Studio, MRDS) 仿真平台对提出的机器人进行建模,将融合算法封装为一个服务提供给移动机器人,指导其进行路径规划。采用量化分析的方式,从规划路径长度、重规划次数、更新节点数以及规划花费时间等四个维度与传统  $D^*$  Lite 算法下路径规划进行对比分析。实验结果表明融合算法可以大大优化规划花费时间,为大地图复杂环境下路径规划提供了新的可行方案。

**关键词:** 移动机器人; 路径规划; 人工智能; 神经网络

## Abstract

As the core of the mobile robot navigation system, path planning plays an indispensable role in life service navigation, extraterrestrial exploration, unmanned driving, underwater exploration and so on. It solves the problem of "how to go" based on knowing the start position and target position. Path planning experienced the development process from knowing the environment to unknown, simple environment to the complex environment, a small map to the big map, simple artificial intelligence to advanced artificial intelligence. At present, the research of path planning for mobile robots in unknown environment has not yet formed a system. Especially in the large map and complex environment, the traditional artificial intelligence path planning method using high-resolution map representation environment, planning has blindness, complexity and planning time-consuming and other shortcomings. In this paper, how to optimize the environmental map spatial representation and improve the flexibility and efficiency of the planning to reduce the path planning time will be studied.

The traditional path planning method in solving the large-map environment path planning, usually grids the large-map to small, the drawback is that in the unknown environment, the times of re-planning will be much more, and as the grid increases, the complexity of planning algorithm increases exponentially and planning is time-consuming seriously. In addition, as for complex environment, the traditional planning method can not make rational judgments according to the environmental characteristics and that leads to its low efficiency. In this paper, the traditional path planning method is analyzed in detail, and the advantages and disadvantages of each algorithm and the specific scenarios are analyzed as well. Finally, a fusion algorithm is proposed for the path planning under the complicated large-scale map in this paper, that is, using the hierarchical map to plan. Refine the planning method to get a collection of a variety of algorithms so that makes every aspect the best performance. As for the global, the fusion method uses D \* Lite algorithm which has a excellent performance for the global path planning, and as for the local, the fusion method uses enhanced neural network algorithm to have the ability to autonomously learn environment. In fact, it is reasonable to adopt a variety of algorithm fusion methods to solve the path planning under certain circumstances. It can collect the advantages of a variety of algorithms, for different specific path planning to achieve local optimal, and then to complement the advantages of the algorithm between the results.

In this paper, the proposed MRDS simulation platform is used to model the proposed robot, and the fusion algorithm is encapsulated as a service to the mobile robot to guide its path planning. By using the quantitative analysis method, four dimensions include the planning path length, the number of re-planning, the number of updating nodes and the planning time are compared with the traditional D \* Lite algorithm path planning. The experimental results show that the fusion algorithm can greatly optimize the planning time and provide a new feasible scheme for road planning under the complex environment of large map.

**Keywords:** mobile robot, path planning, artificial intelligence, neural network

# 目 录

摘 要 .....	I
ABSTRACT.....	II
第 1 章 绪 论 .....	1
1.1 课题背景及研究的目的和意义 .....	1
1.2 移动机器人及路径规划发展概况.....	2
1.2.1 国内外研究现状.....	2
1.2.2 移动机器人的研究内容及发展趋势 .....	3
1.2.3 路径规划研究内容及趋势.....	5
1.3 本文的主要研究内容 .....	5
第 2 章 路径规划相关理论研究 .....	7
2.1 引言 .....	7
2.2 规划问题表述 .....	7
2.3 状态空间规划 .....	9
2.4 路径规划地图表示 .....	11
2.5 本章小结.....	12
第 3 章 路径规划算法研究 .....	14
3.1 引言 .....	14
3.2 基于图的搜索算法.....	15
3.2.1 A*搜索算法.....	15
3.2.2 D*搜索算法.....	17
3.2.3 LPA*搜索算法 .....	20
3.2.4 D* Lite 搜索算法 .....	23
3.3 局部路径规划方法.....	27
3.3.1 人工势场法.....	27
3.3.2 神经网络路径规划.....	29
3.4 基于采样的规划方法 .....	29
3.4.1 PRM 算法 .....	30
3.4.2 RRT 算法 .....	32
3.5 本章小结.....	34
第 4 章 D* Lite 与神经网络融合路径规划研究 .....	35

4.1 引言 .....	35
4.2 D* Lite 算法的不足 .....	35
4.3 D* Lite 和 BP 神经网络融合路径规划 .....	36
4.4 神经网络的优化与改进 .....	41
4.5 本章小结 .....	42
<b>第 5 章 基于 MRDS 平台的仿真实验 .....</b>	<b>43</b>
5.1 引言 .....	43
5.2 实验平台简介 .....	43
5.3 实验环境搭建 .....	46
5.3.1 移动机器人建模 .....	46
5.3.2 地图设计 .....	47
5.3.3 定位问题的处理 .....	49
5.4 实验结果 .....	49
5.5 性能比较与量化分析 .....	52
5.6 本章小结 .....	54
<b>结 论 .....</b>	<b>55</b>
<b>参考文献 .....</b>	<b>56</b>
<b>攻读硕士期间发表论文及成果 .....</b>	<b>59</b>
<b>哈尔滨工业大学学位论文原创性声明和使用权限 .....</b>	<b>60</b>
<b>致 谢 .....</b>	<b>61</b>



# 第1章 绪 论

## 1.1 课题背景及研究的目的和意义

近些年来, 机器人技术伴随计算机、精密仪器设备制造、人工智能算法等技术呈现出了快速发展的趋势。其应用遍及军事、工业、海洋、交通、太空探索、医疗等诸多领域, 已成为各国争先发展的技术之一。自主移动机器人作为机器人研究领域的重要分支, 逐渐成为各国及相关公司和科研机构研究的重点之一。移动机器人关键在于其具有“移动”的特性, 一般需要具备定位、环境感知、导航等功能。因此, 移动机器人的研究涉及到机械工程、传感器技术、计算机技术、自动化控制、人工智能等诸多方面<sup>[1]</sup>, 是一个涉及面很广泛的交叉学科。移动机器人能够通过自身的传感器感知周边环境以及自身的状态, 并根据获取的信息数据实现躲避障碍物、移动至指定位置、完成一定的要求作业。

目前, 人们对移动机器人的需求越来越多, 各种服务型机器人不断出现, 例如帮助家庭打扫卫生的扫地机器人, 协助医院手术医生的手术机器人以及日本发生地震时使用的搜救机器人等等。面对人们的种种需求, 移动机器人技术的发展有着新的挑战, 运动控制、传感器以及定位精度、路径规划等诸多方面都有着更高的要求。针对于路径规划方面, 移动机器人经历了已知环境地图路径搜索到未知环境地图路径搜索的发展、小场景地图到大场景地图路径规划发展以及单一智能体路径规划以及群体智能体协作路径规划的发展<sup>[2]</sup>。对于已知地图路径规划目前已经有了较为成熟的研究成果, 但对于环境未知的路径规划虽然已有一定的探索研究, 但尚未形成体系<sup>[3]</sup>。这也是日本虽然在移动机器人领域处于较为领先的地位, 但其移动机器人在地震救援时针对未知的场景却不能迅速做出较好的规划决策而无法发挥预期的作用。在环境部分未知或全部未知的情况下做出较好的路径规划依然是目前研究的重要方向之一<sup>[4]</sup>。

已知环境的路径规划大多在实际应用中需要借助外界辅助技术对地图完成精准构建, 如果这一步能够很好完成的话, 那么很多成熟的算法就能够得到很好的应用。然而现实情况较复杂, 首先, 借助辅助技术构建地图时, 地图的空间会受限。其次, 大多情况下移动机器人通过自身航迹推算导航会产生巨大的误差, 转而采用定位系统辅助, 然而室内定位精度问题一直没有很好的解决方案<sup>[5]</sup>, 这导致的后果就是做路径规划时的位置信息存在模糊度, 进而无法实现较为精准的路径规划。这从某种意义上加大了对未知环境路径规划的需求, 借助自身携带的传感器, 例如激光、红外、超声波等对自身周围环境进行良好建模, 然后采用分层

思想, 先进行全局路径规划得出路径规划节点序列, 然后再采取局部路径规划的方法解决相邻序列节点抵达任务。此方法对未知路径规划方法是一种较好的解决方案<sup>[6]</sup>。本论文试图采用此方案对局部路径规划做一些优化, 使得在部分未知的复杂环境的大地图中的路径规划更为有效。

综上所述, 目前移动机器人在大地图以及未知地图方面的路径规划效果尚且不理想, 主要表现在局部小范围精度不易控制, 自主定位与地图建模不够准确, 这些都是目前研究的关键问题。好的路径规划能力能够使移动机器人在太空探索、工业以及人们生活等方面有着很重要的作用。在进入工业 4.0 的进程中, 移动机器人的研究和应用水平是衡量一个国家科技创新和高端工业水平的重要标志<sup>[7]</sup>。本文将在移动机器人路径规划方向做出一定的研究, 为移动机器人技术更好的发展添砖加瓦。

## 1.2 移动机器人及路径规划发展概况

### 1.2.1 国内外研究现状

学术界为机器人的自主移动总结为三大关键问题: **Where am I?** (在哪里?), 自定位问题; **Where am I going?** (到哪里?), 目标规划, 地图表示问题; **How do I get there?** (怎么去?) 导航规划问题<sup>[8]</sup>。自主移动又必须具备四个要素: 环境模型/地图(预先已知或者自主构建), 感知和分析环境(传感器获取), 确定自己在环境中的位置(定位), 规划并执行移动(路径规划)<sup>[9]</sup>。

路径规划作为移动机器人导航系统的主要内容之一, 经历了传统的基于图搜索的算法研究、局部路径规划研究、基于采样的路径规划研究和现代的混合算法路径规划、智能化路径规划的两个大的阶段<sup>[10]</sup>。具有重要历史意义的成果如在静态路网中加入启发函数的 A\* 算法, 它是人工智能中最经典算法之一, 一改以往的盲目搜索导致的巨大复杂度。1994 年 Anthony Stentz 提出了 D\* 算法<sup>[11]</sup>, 它将规划的方向进行逆转, 从目标点向起始点规划, 这样可以处理环境部分未知或全部未知以及动态障碍的一些情况。此算法被作为路径规划的核心算法应用于美国 1996 年发射的火星探测器上, 并获得了很大的成功。Anthony Stentz 又于 1995 年提出了 Focused D\* 算法<sup>[12]</sup>, 重点提高了 D\* 算法的实时性。2005 年 Koenig 和 Likhachev 在 D\* 的基础上提出了 D\* Lite 算法降低了 D\* 路径规划的算法复杂度<sup>[13]</sup>。2006 年 Anthony Stentz 又提出了 Filed D\* 算法, 对多角度路径搜索结果做了进一步优化<sup>[14]</sup>。在局部路径规划方面 1986 年 Khatib 提出的人工势场法是其中最为经典的一个算法<sup>[15]</sup>, 其将障碍点和目标点看做两个力场, 目标点对移动机器人产生引力而障碍点对移动机器人产生斥力。然而人工势场方法容易陷入局部最小值情况而不能到达

目标点。为了解决此问题,2006 年刘义等提出了修改斥力函数的方法<sup>[16]</sup>,哈工大张建英等提出了遇到极小值时为移动机器人添加附加控制力的方法<sup>[17]</sup>,取得了较好的效果。另外神经网络<sup>[18]</sup>、蚁群算法<sup>[19]</sup>、人工免疫<sup>[20]</sup>、遗传<sup>[21]</sup>、退火算法<sup>[22]</sup>等现代智能算法也渐渐被应用的路径规划上来。一些混合算法在路径规划方面可以结合两者优势达到很好的效果,比如王新杰等将遗传算法与图搜索算法相结合<sup>[23]</sup>,郑秀敏等在 2007 年提出的栅格法-模拟退火法<sup>[24]</sup>,黄席樾等提出的基于神经网络模型的遗传算法和模拟退火算法相结合的方法<sup>[25]</sup>,都经试验验证为行之有效的算法。

### 1.2.2 移动机器人的研究内容及发展趋势

移动机器人,顾名思义就是指具有“移动”特性的机器人,其能够在特定环境下进行自主规划的能力。移动机器人的研究主要包含了环境感知的传感器研究、自主定位与地图构建的研究、导航系统研究以及控制系统研究等 4 个方面<sup>[26]</sup>。

移动机器人通常采用传感器对周边环境信息进行捕获,比如采用激光、超声波等传感器捕获障碍,或者采用麦克捕获外界语音与人交互等。在为移动机器人配备传感器时,通常需要考虑移动机器人的使用场景和要实现的目标,另外为了更精准的实现某一功能,比如确定本身的速度或方向信息,通常采用多传感器融合的方式,进行卡尔曼等滤波算法降低数据的误差。

定位一直是移动机器人匹配自己所在位置所必须的功能,定位不好就会走失完全迷失了方向。在移动机器人定位方面,室外通常采用全球定位系统(Global Position System, GPS)等卫星技术以及基站定位技术实现<sup>[27]</sup>,而在室内由于 GPS 信号受到遮挡物影响而使定位失效。目前常用的定位方法有超宽带(Ultra Wide Band, UWB)基站定位<sup>[28]</sup>、ZigBee/蓝牙定位等,但效果不是很理想。由于在定位与地图构建时,精准的定位需要已知的地图,而地图的精准构建又反过来需要精准定位,两者相互依赖互为前提,通常将定位与地图构建放在一起研究,即同时定位与地图构建(Simultaneous Localization and Mapping, SLAM)概念<sup>[29]</sup>。对于 SLAM 的研究主要围绕地图的表示方法、如何获取准确的定位信息、如何实时更新自定位和地图信息等问题<sup>[31]</sup>。在实际设计移动机器人时,通常的采用多传感器数据融合的方法来处理以上问题<sup>[32]</sup>。值得一提的是,Google 最近公布一个开源框架 Cartographer,这是一个机器人操作系统(Robot Operating System, ROS)支持的 2D 和 3D SLAM 库。SLAM 算法结合来自多个传感器(激光传感器、惯性传感器和摄像头等)的数据,同步计算传感器的位置并绘制传感器周围的环境。

导航系统研究是指移动机器人根据自身的位置以及地图信息按照一定的路径规划策略完成人们设定的运动目标。其前提是获取移动机器人当前所在的位置和

要导航的目标点。导航需要识别周边环境存在的障碍和危险状态,保证移动机器人在安全可靠的位置上行走。目前已知环境地图的导航系统已经取得较好的成果,但对于环境地图未知的导航还存在很多问题亟待解决<sup>[33]</sup>。目前常见的导航方式有路标导航即通过识别地图中固定特征的路标来确定运动的情况,视觉导航即根据视觉范围的情景来判断道路障碍等地图信息,声音导航即根据不同方向发出的不同频段的声音判断地图信息等方式。导航的结果是生成一系列的运动序列,其中主要的研究内容有环境地图的表示方式、路径规划策略的设计等。

移动机器人的控制系统指的是移动机器人中各个模块的沟通桥梁,连接着各种传感器、驱动模块、显示模块等。控制系统将导航产生的规划结果,即一系列的行为序列,转换为控制指令落实到硬件执行。控制系统会为各个小的系统提供需求数据,比如将环境感知数据递交给自主定位与地图构建系统、将定位结果和构建的地图交给导航系统等。由于传感器精度的提升以及复杂算法对运算速度的需求以致对控制系统的要求越来越高,驱使着控制系统朝着高速和并行的方向发展。

就目前国内外对移动机器人的研究方向来看,可以将移动机器人研究分为如下五个方面:

(1) 移动机器人机械设计和控制理论研究。机械设计的优良影响着移动机器人运动的灵活性,更精巧的设计使机器人完成更复杂的动作有了可能。机器人控制部分是移动机器人执行指令的核心,目前基于神经网络的控制理论体系已经很成熟,并且随着处理器的并行能力及处理速度的提高,控制理论方向将会有新的突破。

(2) 移动机器人的 SLAM 研究。移动机器人对定位和地图构建有着极高的要求,只有好的定位和地图才能为导航提供精确的依据,而在 SLAM 方面的研究主要是围绕在多传感器数据滤波和融合方法的研究。

(3) 移动机器人并行化处理技术研究。是为满足移动机器人对实时性的要求,主要体现在处理器的性能以及算法的设计上,只有做到数据的并行化处理,才能保证应对突发事件的及时性以及减少顺序处理产生的延时而导致的控制误差。

(4) 优化的路径规划算法研究。一方面是对传统的路径规划的进一步优化和改进,另一方面是对于目前的路径规划算法在处理地图未知或部分未知时存在误差过大从而容易迷路的情况设计较好的路径规划方案。目前采用多种路径规划算法融合的方案是一直较好的解决途径。

(5) 多智能体协作的研究。在一些特定场景中,单个移动机器人的能力有限,而多个移动机器人像人一样能够集体协作完成任务很有必要,因此多智能体的协作策略以及相互间的交互方式都是此类问题的研究方向。

### 1.2.3 路径规划研究内容及趋势

路径规划问题自提出以来已经经过国内外众多学者数十年的深入研究，其中也产生了很多成果，例如地图构建方面的可视图法<sup>[34]</sup>、切线图法<sup>[35]</sup>、Voronoi 图法<sup>[36]</sup>、拓扑法<sup>[37]</sup>、栅格法<sup>[38]</sup>等，搜索算法方面的 A\*、D\*、人工势场、神经网络算法、遗传算法等。

对于路径规划技术根据规划方法特点可分为三类：基于图搜索算法、基于采样的规划算法、局部规划算法。路径规划研究的重点由初期的面向已知静态地图路径规划向面向未知动态地图的局部路径规划方法转变，由单智能体向多智能体转变，由单一算法向多算法融合算法转变，由小地图向大地图转变。研究的目的是针对传统算法的缺点做改进，比如 A\*算法是对 Dijkstra 算法的改进，D\* Lite 算法是对 D\*算法的改进等、将其他领域的智能算法做路径规划方向的应用，比如遗传算法、神经网络算法等以及对多种路径规划算法进行融合达到性能优化的目的<sup>[39]</sup>。

另外，近些年来机器学习、人工智能、机器视觉等领域的成果已经开始全面提升提升了机器人的能力。例如谷歌无人驾驶汽车的成功试行，其在无人驾驶汽车上采用了丰富的传感器，多传感器数据融合与谷歌地图配合实时 SLAM 进行智能导航，以及其深层次研究的深度学习技术，更是将移动机器人的研究推向了新的高度。在机器学习方面，谷歌也有发布开源项目 TensorFlow<sup>[40]</sup>，其支持异构设备分布式计算，它能够在各个平台上自动运行模型。这也标志着移动机器人在未来一段时间内的研究中将加入更多的机器学习的内容。

## 1.3 本文的主要研究内容

本文主要研究并分析了现有的诸多路径规划方法，深入理解每种算法的适用场景及其优缺点，并依此提出一种行之有效且效率较高的解决复杂大地图环境的融合路径规划算法，即全局采用优化的 D\* Lite 算法，局部采用基于 BP 神经网络的路径规划算法。考虑环境地图较大，融合算法将环境地图进行层次划分，分为粗粒度的栅格地图和直接表征的度量地图。移动机器人依靠激光传感器获取周围 180 度的环境数据输入神经网络系统从而计算出局部的行走方案。本文的研究主要内容可以分为如下四个部分：

(1) 阐述路径规划的相关理论，包括规划的表述、规划解的评价标准、与本文密切相关的状态空间规划问题以及状态规划地图的四种表示方法。

(2) 介绍目前路径规划算法的成果，即已研究出的具有代表意义的路径规划算法，分析其原理及优缺点，并选择部分典型的算法做了仿真演示和对比试验。

此部分为本文提出混合路径规划方案做了必要的准备工作。

(3) 提出了本文最为核心的路径规划方案, 分析了目前路径规划效果较好的 D\* Lite 算法存在的某些问题, 对于局部路径规划采用创新的神经网络进行路径规划, 神经网络的设计则采用了增强学习的方式进行训练, 从而达到移动机器人能够自主探索未知环境的目的。

(4) 采用微软提供的微软机器人开发者平台 (Microsoft Robotics Developer Studio, MRDS) 平台搭建模型<sup>[41]</sup>, 进行 3D 效果仿真, 对算法仿真的结果进行分析, 采用量化的方式对不同算法进行数据对比验证, 进而验证方案的可行性与有效性。

## 第 2 章 路径规划相关理论研究

### 2.1 引言

移动机器人是一个复杂系统，移动机器人的研究涵盖了同时定位与地图构建（SLAM）、运动控制系统、环境感知系统以及目标导航系统。其中环境感知系统通过传感器获取外界的状态更新信息，SLAM 为移动机器人提供较精准的地图位置信息，导航系统根据前两者提供的数据信息采用合理的路径规划算法来规划路径，并交由运动控制系统执行，如图 2-1 所示。目前在实际的移动机器人系统中更多采用的是并行化处理。

移动机器人路径规划是移动机器人导航系统中的运动动作决策行为，其策略的制定以移动机器人本身以及周边环境为依据，针对某一个或者一些条件约束找到抵达目标点的路径解。规划一直以来都是人工智能、控制理论以及移动机器人等学科的探索内容，它是经过一些抽象而清晰的深思熟虑，制定出关于动作的推理过程。

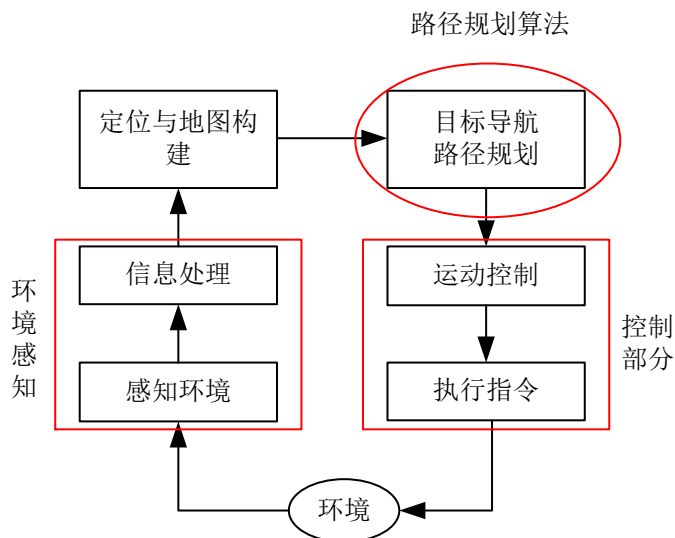


图 2-1 移动机器人体系结构示意图

### 2.2 规划问题表述

对于规划问题的描述可以使用状态空间来描述，状态空间是一个规划对象的状态集合，比如移动机器人在环境中的所有状态集合包括了其所在位置和姿态信

息<sup>[42]</sup>。对规划问题求解则是制定一种规划器，其能够根据初始状态和目标描述作出计划从而使控制器执行相应的动作，即改变了规划对象的状态，执行过程中需要和外界环境进行交互，对外界环境的事件进行观察并进一步分析得出下一步计划，如此反复，直到求出问题解。规划问题的解即为从初始状态转换到目标状态对应于时间轴或者时间序列的状态转换动作集合，也称行为集。行为在离散状态空间中可表现为转换函数，能够使系统从一个状态转换成另一状态。大多情况下，规划问题也是 NP 问题，在研究时需要考虑规划的最优化方法。规划的目标是在状态空间中找到完备有效的行为集，对于规划的评价按照递增的次序可分为可行性与最优性，前者是在有解的情况下能够通过规划找到从初始状态到目标状态的解，后者是指在前者基础上添加某些约束条件并使其达到最优化效果。

在移动机器人路径规划模型中，如图 2-2，规划器即是我们设计的路径规划算法，初始状态即是机器人的起始状态包括初始姿态信息和初始位置信息，而目标状态也同样包含目标姿态信息和目标位置信息。路径规划的过程就是制定一系列行动指令集，供控制器执行，在整个过程中传感器可以不断获取环境信息更新状态空间，并对新的状态空间和新的初始状态重新进行路径规划。

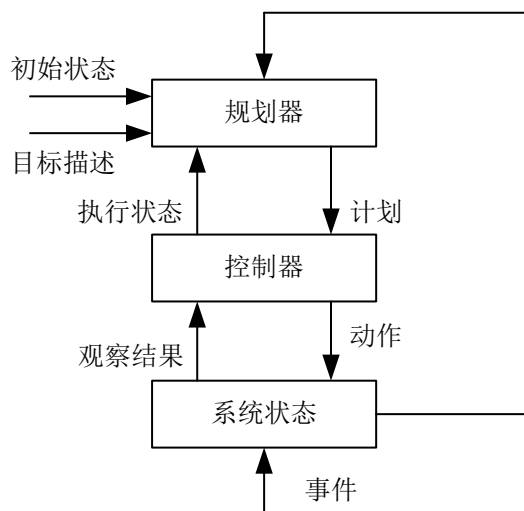


图 2-2 规划模型示意图

为了分析规划问题的判定性和复杂性，通常需要把这些问题进行语言识别表述，给定规划问题的描述  $D$ ：是描述  $P \in D$  的集合， $P$  表示一个可解的规划问题； $PLAN\_EXISTENCE(D)$  是描述  $P \in D$  的集合， $P$  表示一个可解的规划问题，并且这个规划问题有一个长度不超过  $k$  的解。规划研究已取得的成果表明，对于状态变量规划来说， $PLAN\_EXISTENCE(D)$  是可判定的，因为可能的状态数是有限的，因此可以使用穷尽搜索来研究问题是否存在解。如果对状态变量规划加以扩



展, 允许包含符号项, 则  $\text{PLAN\_EXISTENCE}(D)$  半可判定, 而  $\text{PLAN\_LENGTH}(D)$  仍可判定。经典路径规划以及状态变量路径规划的复杂度从常数时间到指数级时间不等, 而受到条件约束, 前提条件以及负效果因素影响, 状态变量规划的  $\text{PLAN\_LENGTH}(D)$  复杂性通常是 NP 完全的。

另外, 规划解的评价标准也很重要, 其主要包含三个方面, 即可靠性、完备性以及可纳性<sup>[43]</sup>。可靠性指的是规划算法在对规划问题求解时返回结果不为失败, 则返回解一定是规划问题的解。如果规划的解不在解空间之中, 则称其不可靠。完备性指当规划算法在对规划问题求解时, 规划解存在则一定返回不为失败结果。可纳性指规划算法在对规划问题求解, 当解存在时能够返回满足约束  $P$  的最优解, 比如规划消耗时间最少、规划占用状态空间最少等做到最优化。规划算法满足可纳则也一定可靠且完备。对于大地图复杂环境下的移动机器人路径规划的评价标准通常有规划的路径长度、规划的花费时间等具体约束条件。

## 2.3 状态空间规划

在经典规划算法当中, 状态空间搜索是在移动机器人路径规划中经常使用的规划方式, 其规划出的节点序列皆为搜索空间的状态, 相邻两个节点之间的连接就是一个搜索空间中状态间的转换。根据规划算法的起始规划点是初始状态或是目标状态可将搜索算法分为正向搜索算法和反向搜索算法, 如图 2-3。

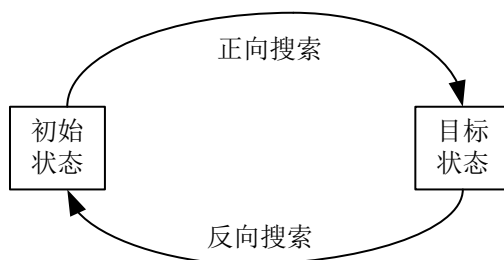


图 2-3 两类状态空间搜索算法

正向搜索算法是一种可靠的搜索算法, 它以规划问题  $p$  的描述  $P = (O, s_0, g)$  为输入, 其中  $O$  是操作集,  $s_0$  是初始状态,  $g$  是目标状态。如果规划问题  $p$  有解, 则返回规划解, 否则返回搜索失败。正向搜索是一种以初始状态为规划到目标状态的解空间的搜索方式, 出发, 步步为营, 不断向目标状态靠近的搜索算法。目前的路径规划算法大多属于此类, 在图论搜索理论当中, 我们学习到的深度优先搜索、广度优先搜索以及 Dijkstra 算法都属于此类, 由于状态空间是有限的, 任何由

正向搜索算法  $Forward-search(O, s_0, g)$  返回的规划解都是规划问题的解，并且该解也一定在规划问题 $p$ 的解空间中。正向搜索算法是完备的，其搜索空间要比需要的空间大，但是可以通过剪枝技术缩小搜索空间。Dijkstra 就是对广度优先算法的搜索空间剪枝，而 A\*算法则是利用启发函数对 Dijkstra 算法的搜索空间剪枝的优化算法。正向搜索的算法的一种实现描述如表 2-1。

表 2-1 正向搜索算法

算法: Forward-search	
输入: 操作集 $O$ ，初始状态 $s_0$ ，目标状态 $g$	
1.	将初始状态 $s_0$ 添加到 $Q$ 队列，标记 $s_0$ 为 visited 状态
2.	while $Q$ 不为空 do
3.	从 $Q$ 中取出一个节点赋给 $x$
4.	if $x == g$
5.	找到目标状态，返回规划解
6.	for all $u$ 在 $U(x)$ 中，
7.	将 $f(x, u)$ 赋给 $x'$
8.	if $x'$ not visited
9.	标记 $x'$ 为 visited 状态
10.	$Q$ 添加 $x'$
11.	else
12.	丢弃 $x'$ 并执行循环
13.	return false

反向搜索顾名思义就是指搜索从目标状态开始，反向查找直到遇到初始状态。类似的，反向搜索亦是完备且可靠的，对于目标  $g$  的前面状态满足下述条件的集合  $s_g$ ： $\Gamma^{-1}(g) = \{s | \text{存在动作 } a \text{ 使得 } r^{-1}(g, a) \text{ 满足 } g\}$ 。反向搜索表面上只是改变了求解的思路，将从初始状态开始规划的固有思维打破，提供了另一种求解规划的算法。也许你会觉得这样做并不能直接带来效率的提高，然而，反向搜索能够解决一些正向搜索解决起来很棘手的问题，比如当状态空间部分未知或动态改变时，此时采用正向搜索的思路是反复的重新规划，此时搜索效率十分低下，而反向搜索在完成一次搜索后便维护一定当前状态到目标状态的解空间。当局部遇到未知状态或者状态动态改变时，其维护的解空间仍可用，此时仅需要做少量的解空间的更改就能重新得到一个新的规划解。此类算法的一个比较好的代表就是 D\*算法。反向搜索算法的一种实现描述如表 2-2。

表 2-2 反向搜索算法

算法: Backward-search

输入: 操作集  $O$ , 初始状态  $s_0$ , 目标状态  $g$

1. 规划解  $\pi$  置空
2. 循环
3.   if  $s_0$  满足  $g$  then
4.     返回  $\pi$
5.   将从集合  $\{a | a \text{ 是与 } g \text{ 相关的操作集的实例}\}$  取出一个 relevant
6.   if relevant 为 null
7.     返回搜索失败
8.   选择一个行为  $a$  满足一定条件
9.   将  $a$  的搜索解赋给  $\pi$
10. 将  $r^{-1}(g, a)$  赋给  $g$

## 2.4 路径规划地图表示

移动机器人的路径规划需要对环境信息有所感知, 环境信息主要通过定位和传感器来获取, 在移动机器人路径规划中表现为环境地图。环境地图的表示方法主要有栅格地图、几何特征地图、拓扑地图以及直接表示法地图四种。

栅格地图法于 1968 年由 W.E.Howden 提出, 它是将环境空间按照一定的分辨率分割成大小相同的栅格单元。每个栅格单元对应与实际环境的某个位置, 具有相应的状态, 比如栅格有无障碍或者部分被障碍占据。对于栅格大小的划分即环境的分辨率是一个需要综合考虑进而寻找平衡点的要素, 直接影响路径规划的性能, 栅格粒度越小, 则环境表示越精准, 但空间存储越大, 算法搜索时间越长。反之, 则会影响路径规划的精度。栅格地图法创建和维护实现起来较为简单, 能够较好的保存环境信息, 便于移动机器人做路径规划。

几何地图表示法是通过外部传感器捕获环境状态数据, 根据这些数据采用线段或者曲线描绘这些物体或者空间的几何特征。由于这些几何特征的提取需要精密的传感器设备, 对噪声比较敏感, 通常用于对结构化物体的建模, 比如在室内对墙壁和桌椅的建模。

拓扑地图表示法于 1994 年由 Kortenkamp D 和 Weymouth T 提出, 拓扑地图主要把握环境中的若干关键节点, 使用直线或者弧线将这些关键节点连接, 直线弧线即代表节点间的通路。移动机器人的规划就是位于某一节点时, 能够寻找那些

弧线的组合可以更有效的抵达目标点。因此可知拓扑表示法具有存储空间小、效率高、灵活性强等优点，但对环境中出现的比较相似的节点难以区分。

直接表征法直接对环境信息进行地图构建，将环境中的障碍物以及自由空间真实的映射为二维或三维的地图中，这种表示方法最为准确的还原了环境中的真实形态，对于复杂精细的规划很有必要。其代价也很明显，那就是精准还原真实环境的形态需要很大的计算量，在映射过程中需要对噪声误差的产生做一定处理。

图 2-4 为四种地图表示方法的示意图。在已知地图中，以上地图建模方法都是适用的，但一般会选择实现较为简单、计算并不复杂的地图表示方式，比如几何和拓扑地图表示法。而对于环境未知，或者较大地图，由于缺乏对环境的认知，不能总体划分自由空间和障碍空间，因而需要采用可控的且随着移动机器人的探索不断填充未知区域地图的方式进行建模，此时栅格地图表示法以及直接表征法比较适合。对于大地图来说一般会引入双层地图进行表示，即低分辨率的高层与高分辨率的低层。在对环境建模时经常会采用混合多种表示法表示地图，但对于复杂一些的环境为了使其能够更精准的还原地图使得移动机器人能够拥有更多的姿态空间，一般采用较低分辨率的栅格法表示或者采用直接表征法表示。

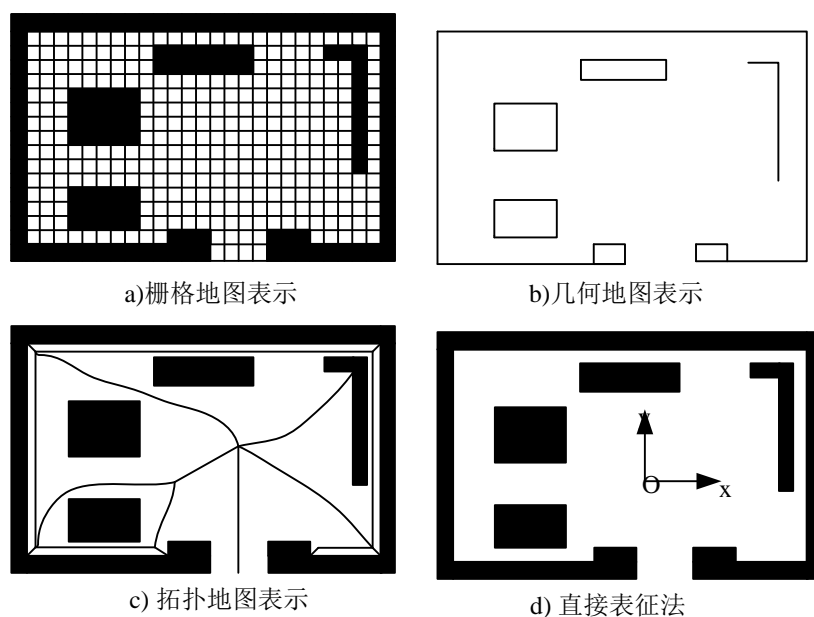


图 2-4 地图表示方法示意图

## 2.5 本章小结

本章首先介绍了规划问题，重点介绍了与本文相关的状态空间描述方法和其评价标准。随后分析了状态空间规划方法，其为移动机器人路径规划的常用方法，

主要有正向搜索算法和反向搜索算法两种，其中反向搜索算法常用于状态空间不确定即路径规划的地图未知或部分未知的情况。最后对目前常用的路径规划地图表示方法进行分析，在针对大地图做路径规划时，常常采用分层混合地图表示法，这样便形成了下一节将介绍的全局路径规划和局部路径规划的概念。

## 第3章 路径规划算法研究

### 3.1 引言

移动机器人的路径规划是集人工智能、控制理论以及机器人学等多门学科的交叉学科，是目前机器人领域的主要研究内容之一。路径规划是以某种条件为约束目标，可以是路径最短、花费时间最少等，规划出一条能够避开障碍物顺利抵达到目标状态的路径。在路径规划研究中，能达到的空间集合成为路径规划的工作空间，能以任意姿态达到的点的集合成为灵活的工作空间。一个机器人所有可能的姿态集合称为姿态空间，不可行的姿态集合称为障碍物空间，在障碍物空间中机器人会与障碍物发生碰撞，可行的姿态集合称为自由空间，在自由空间中机器人将无碰地安全移动。路径规划也就是在自由姿态空间中为机器人寻找一条路径，使其从初始姿态发展到目标姿态，如图 3-1。

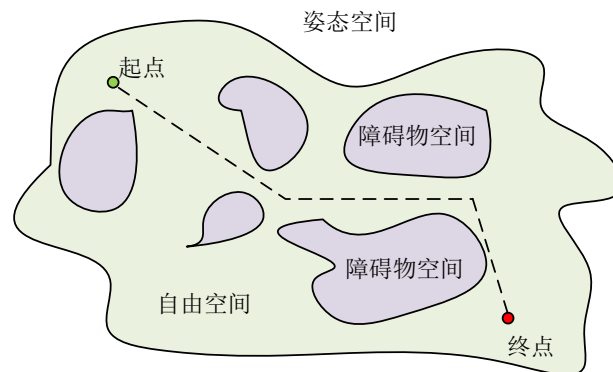


图 3-1 姿态空间示意图

移动机器人路径规划可以分为两种类型：全局路径规划和局部路径规划。通常使用全局路径规划将全局目标分解为一系列小的局部目标，再使用局部路径规划算法实现局部目标。事实上，大部分路径规划方法即可用作全局路径规划也可用作局部路径规划，通常需要基于特定的环境选取特定的算法对该场景下的路径规划做具体优化。此时，路径规划需要解决的问题包括：1）当解存在时，能够在有限的时间内找到解；2）规划的解需能够绕开障碍物；3）规划的解尽可能满足约束条件最优。

本文按照移动机器人路径规划方法特点对路径规划算法进行分类，分别对基于图的搜索算法、局部路径规划算法以及基于采样的规划方法进行研究。分析典

型算法的原理，对其特点、适用场景以及优缺点进行详细描述，并对部分算法做出相应的仿真实验。

## 3.2 基于图的搜索算法

基于图的搜索算法将姿态空间离散化，在离散姿态空间中搜索最优路径，当离散姿态空间中存在解时，能够在有限时间内找到解。

### 3.2.1 A\*搜索算法

A\*算法是根据评估函数在静态连通图中寻找最优路径的经典启发式搜索算法，也是最有效的直接搜索算法。在 A\*算法之前，路径规划通常采用 Dijkstra 算法，Dijkstra 是一种广度优先搜索算法，但其搜索具有盲目性。A\*算法则是利用启发函数对 Dijkstra 的搜索空间进行剪枝的优化算法。两者最大的差别就是启发函数，A\*算法的搜索过程是根据启发函数值的大小向着代价值低的方向进行。即对于所处的节点  $n$ ，算法利用估价函数对其周边节点进行评估，并选择估价值最小的点为下一节点，表达式为  $f(n) = g(n) + h(n)$ 。启发函数  $h(n)$  表示当前节点位置  $n$  到目标点的估计值，而  $g(n)$  表示出发点到当前位置节点  $n$  实际的代价，估计值  $f(n)$  即由这两部分组成。在栅格地图中，启发函数通常采用两点间距离表示，例如曼哈顿距离。

表 3-1 A\*算法

算法：A*
<ol style="list-style-type: none"> <li>1. Openlist.Clear(); ClosedList.Clear();</li> <li>2. currentNode = nil;</li> <li>3. startNode.g(x) = 0;</li> <li>4. Openlist.Push(startNode);</li> <li>5. While currentNode != goalNode</li> <li>6.     currentNode = OpenList.Pop();</li> <li>7.     for each s in currentNode.Children[]</li> <li>8.         s.g(x) = currentNode.g(x) + c(currentNode, s);</li> <li>9.         OpenList.Push(s);</li> <li>10.     end for each</li> <li>11.     ClosedList.Push(currentNode);</li> <li>12. End while</li> </ol>

A\*算法的数据结构有：描述环境的数组或图 Graph、描述数组或图的节点 Node、开启列表 Open List 以及关闭列表 Closed List。每一节点拥有前文讲到的实际代价  $g(n)$ 、启发函数  $h(n)$ 、与之相连的孩子节点数组 Children[]。开启列表 Open List 的每个节点都要被检测，并按照估价值  $f(n)$  的大小进行排序。关闭列表 Closed List 存放被访问过的节点，放入关闭列表的节点的父节点都会被遍历且当目标点被加入到关闭列表时算法结束。算法实现描述如表 3-1，流程图如图 3-2 所示。

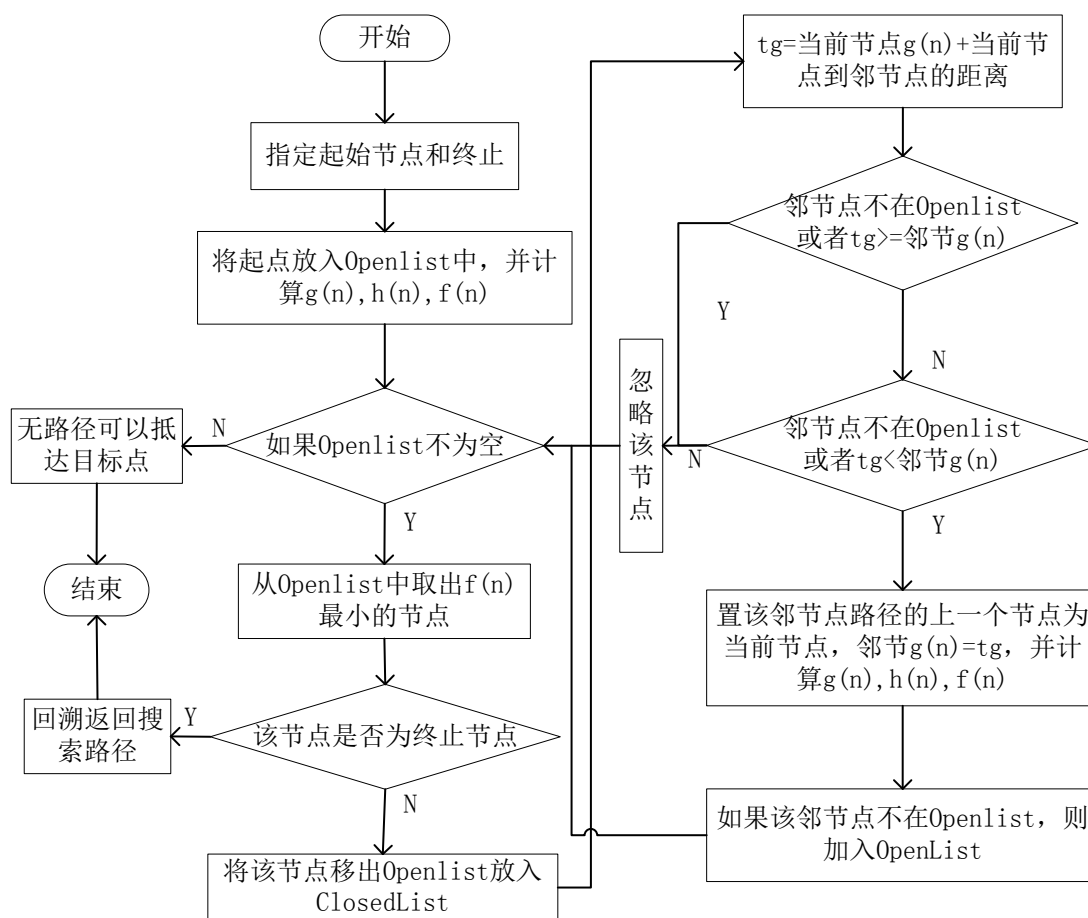


图 3-2 A\*算法流程图

A\*算法的应用场景是静态地图，经常用于游戏中精元的路径规划，因为游戏中的地图是已知的。对 A\*算法的复杂度优化有二叉堆和深度迭代两种方式。图 3-3 是使用二叉堆和深度迭代方法改进的 A\*算法。表 3-2 是在 40\*40 的栅格地图中分别采用改进的 A\*算法与原始 A\*以及 Dijkstra 随机选择起始点和目标点的使用步数以及规划结果的路径长度，他们都能找到最优解，而改进的 A\*比原始 A\*以及 Dijkstra 算法的使用的步数得到了明显优化，由于无法量化算法的复杂度，只能采用简单采样对比来演示。



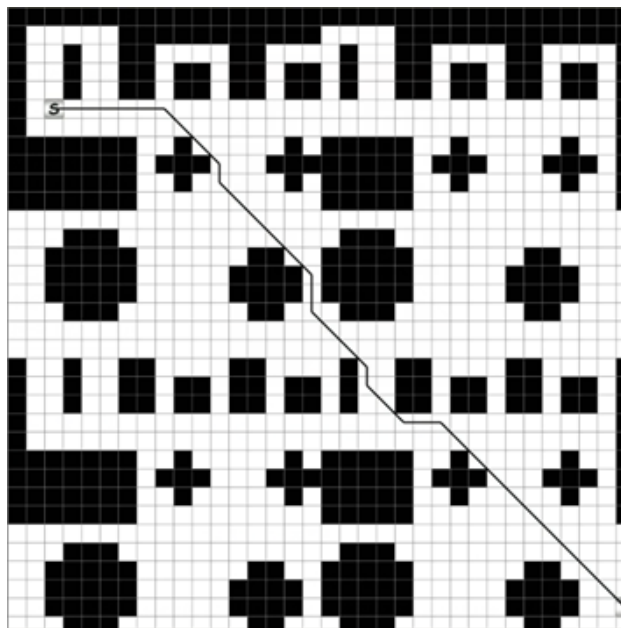


图 3-3 改进 A\*算法仿真图

表 3-2 基于二叉堆与深度迭代的改进 A\*算法对比

算法	使用 步数	路径 长度	使用 步数	路径 长度	使用 步数	路径 长度	使用 步数	路径 长度	使用 步数	路径 长度
Dijkstra	165	17	211	19	59	11	210	19	63	11
原始 A*	73	17	91	19	37	11	85	19	39	11
改进 A*	20	17	20	19	13	11	31	19	13	11

A\*算法在采用较好的启发函数时是最优的静态网路路径搜索算法，然而它也有一些缺点，比如当网络出现动态障碍时，它的规划会立即失效而毫无价值，只能重新进行全局路径规划，并且在规划时容易产生许多弯路而不方便实际移动机器人行走。

### 3.2.2 D\*搜索算法

D\*算法又被称为动态的 A\*算法，曾用在火星探测器的路径规划算法。在未知环境或者有动态障碍出现时，采用以前的 A\*算法需要丢弃运算的当前节点的开启列表和关闭列表以及启发函数的估价值等信息数据，重新进行规划，这样无疑产生了巨大的计算信息损失，在此种情景下 A\*算法无法很好的使用，因此诞生了 D\*算法。D\*算法的核心思想是：先采用 Dijkstra 或者 A\*算法从目标点 G 向起始点进行反向搜索，在搜索的过程中存储了网路中目标点到各个节点的最短路径  $h$  到该节点目标节点的实际路径最短长度  $k$ ，并且搜索出的路径中每个节点包含上一节点

到目标点的最短路径信息。这时其实就是通过反向搜索建立了一个“路径场”，节点路径信息保存在 OPEN 和 CLOSE 中。在遇到动态或临时添加的障碍时，“路径场”信息能够避免不必要的重新路径规划带来的庞大运算量。

当“路径场”建立完成后，机器人便沿着最短路径开始向目标点移动，在移动的过程中，如果下一节点没有变化时（ $k=h$ ），无需重新计算，利用反向搜索时 Dijkstra 算法计算出的路径继续向目标点靠近，当在  $Y$  点发现到下一节点  $X$  状态发生了变化（ $k < h$ ），如  $X$  点被障碍物占据等因素导致从  $Y$  到  $X$  的权值发生变化。此时，机器人需要更新自己从当前位置  $Y$  到目标点  $G$  的实际值  $h(Y)$ 。 $h(Y)=c(X,Y)+h(X)$ ，其中  $c(X,Y)$  代表相邻两点  $X$  与  $Y$  中  $X$  到  $Y$  的新权值， $h(X)$  为  $X$  到目标点的原实际值。其中  $k$  值取  $h$  值变化前后的最小值。

D\*算法首先通过 Dijkstra 或者 A\*算法建立好“路径场”，相较于直接 A\*算法重新路径规划可以节省出路径信息计算量。图 3-4 展示了在 D\*算法动态避障原理，即上文讲述的，遇到障碍时，其实际路径长度和最短路径长度不相等，更新部分规划就可做到重新路径规划，做到了运算数据的重复利用。

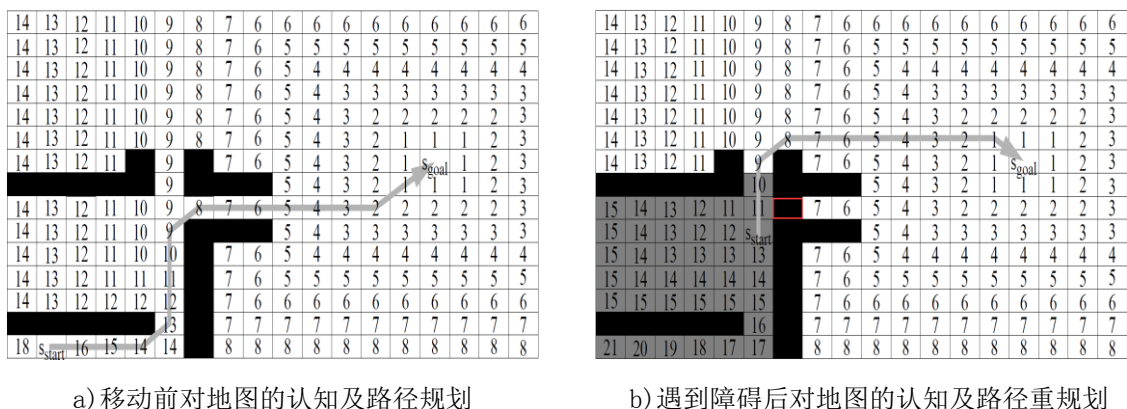


图 3-4 D\*算法路径规划遇障重规划示意图

在移动机器人行走的过程中，如果发现节点的  $k < h$ ，我们称 RAISE 状态，此时可能遇到了障碍物，而如果发现节点的  $k = h$ ，我们称之为 LOWER 状态，路径代价降低，当  $X$  从 OPEN 表中移走时会将更新信息传播到相邻节点。图 3-5 b) 展示了移动机器人根据已知地图反向搜索建立的“路径场”信息，当按照最短路径行移动时发现某一节点被障碍物占据，图 3-5 b)，从而发生了 RAISE，并随着其离开 OPEN 表而更新了相邻节点的路径代价值（图中灰色部分）。此时移动机器人将以当前位置节点为起点重新进行路径规划，由于其反向搜索的特性，起始点不断向目标点靠近，每次重规划目标节点附近的信息可以被重复利用。D\* Lite 算法的描述如表 3-3 所示。

表 3-3 D\*算法

算法: D\*

**Process-state()** 循环执行规划路线

1. 从开启列表中找到估价最小的X节点作为规划序列的下一节点
2. if X为空, 即开启列表为空, then 返回无解
3. X不为空, 计算X节点的k值, 从开启列表移除X, 将X状态设置为CLOSED
4. if  $k_{old} < h(X)$  //X处于RAISE状态
5. 遍历X的邻节点, 对于其中的一个节点Y
6. if 若  $k_{old} \geq h(X)$
7. 且  $h(X) > c(X, Y) + h(Y)$
8. 通过Y可以降低估价值, 将Y设置为X的前节点, 更新h(X)为 $h(Y) + c(Y, X)$
9. if  $k(x) = h(x)$  //处于LOWER状态
10. 遍历X的邻节点, 对于其中的一个节点Y
11. if Y状态为NEW或
12. X为Y的前节点且  $h(Y) \neq c(X, Y) + h(X)$  或
13. X不为Y的前节点且  $h(Y) > c(X, Y) + h(X)$
14. 将X设置为Y的前节点, 将Y加入开启列表
15. else 遍历X的邻节点, 对于其中的一个节点Y
16. if Y状态为 NEW 或
17. X为Y的前节点且  $h(Y) \neq c(X, Y) + h(X)$
18. 将X设置为Y的前节点, 将Y加入开启列表
19. else
20. if X不为Y的前节点且  $h(Y) > c(X, Y) + h(X)$
21. 将X加入开启列表
22. else
23. if X不为Y的前节点且  $h(X) > c(X, Y) + h(Y)$
24. 且Y的状态为CLOSED
25. 且  $h(Y) > k_{old}$
26. 将Y插入开启列表

**Modidy-Cost** (X,Y,cval) 传播开销变化

1. 更改X到Y的开销  $c(X, Y)$  为新值cval
2. if X状态为CLOSED 将X插入开启列表

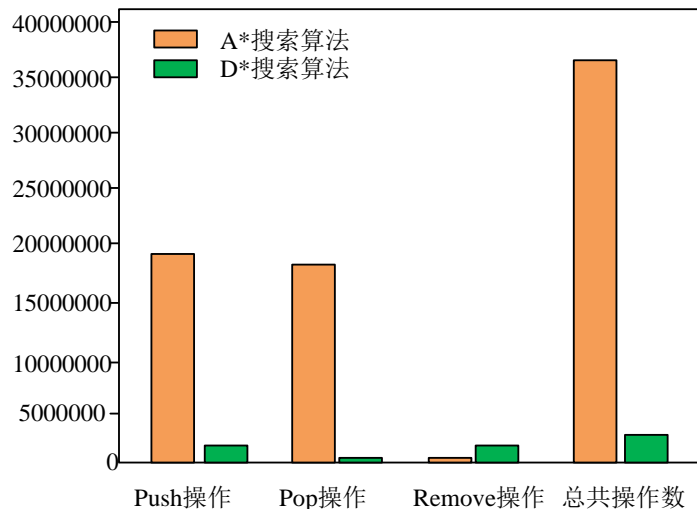


图 3-5 A\*搜索与 D\*搜索的操作数比较

表 3-4 表 A\*重规划与 D\*算法的性能比较

栅格数量	1000	10000	100000	1000000
A*重规划	412 ms	16.52 s	8.5 min	43.53 min
D*算法	250 ms	1.22 s	9.12 s	14.26 s
速度提升倍数	1.648	13.541	55.921	183.156

如图 3-5 显示的是在动态环境地图下使用 A\*不断重规划与使用 D\*算法的操作数的直观比较。分析可以发现，D\*算法的使用场景为环境信息尽量已知，当移动机器人按照规划的路径行走时发现障碍时，可以根据已经建立的“路径场”信息减少重规划的节点数。然而当环境未知时，D\*算法并不能很好的应对，这也就是 D\* Lite 算法的优点。表 3-4 为相同硬件配置相同环境不同栅格地图下 A\*重规划与 D\*算法的性能比较。可以看到，与 A\*重规划相比，栅格地图越大的 D\*算法的效率越突出。

### 3.2.3 LPA\*搜索算法

LPA\* (Lifelong Planning A\*) 算法，是增量式 A\*算法，即在使用 A\*算法进行路径规划时，使用  $rhs$  为每个栅格节点保存父节点的路径规划代价值，一旦此节点遇到障碍还可以利用保存的父节点的规划信息重新规划，而不必做全局的规划。当执行路径规划的路线时遇到了障碍时，只是简单将相应栅格填充为障碍点，重复利用已经计算出的路径规划信息，只对估价值改变的栅格（增量）重新计算，因此需要对已经计算的栅格估价信息做存储，增加了  $rhs(s)$  参数。即 LPA\*算法维护三个参数：从起始点到  $s$  的实际的代价  $g(s)$ 、当前节点位置  $n$  到目标点的估计

图 3-6  $g(s)$  与  $rhs(s)$  关系

如图 3-6 a), 在正常的计算过程中  $rhs(s_2) = g(s_1) + c(s_1, s_2) = A + B = g(s_2)$  即  $g(s_2) = rhs(s_2)$ , 这被称作为本地一致性。当节点  $s_1$  与节点  $s_2$  不可达时, 如图 3-6 b),  $rhs(s_2) = g(s_1) + c(s_1, s_2) = A + \infty > g(s_2)$ , 即  $g(s_2) < rhs(s_2)$ 。当  $g(s) \neq rhs(s)$  时, 称节点  $s$  为本地非一致性。当  $rhs(s) > g(s)$  时, 称低一致性, 一旦发现节点处于本地低一致状态, 则表明该节点的路径花费更大, 一般是遇到了障碍物。发现为低一致性的节点将需要被重置, 路径将完全重计算。当  $rhs(s) < g(s)$  时, 称溢一致性, 当发现一条溢一致性的路径意味着该节点可以降低路径花费, 一般是障碍物节点被清除。

在评价栅格点的估价值时 LPA\*引入  $k(s)$  值用于对开启列表 **OpenList** 中的节点进行比较排序, 其中  $k(s)$  包含两个值  $[k(s_i); k(s_j)]$ , 分别满足以下公式:

$$k_{\gamma}(s) = \min(g(s), rhs(s)) \quad (3-3)$$

路径规划过程就是从当前节点  $s$  的邻节点中选择  $k(s)$  较小或者相等的点作为前进节点, 对于任意相邻节点  $s'$  判断其  $k(s)$  大小的公式为: 且

$$k(s) \leq k(s') \Rightarrow \begin{cases} k_1(s) \leq k_1(s') \\ k_1(s) = k_1(s') \text{ 且 } k_2(s) \leq k_2(s') \end{cases} \quad (3-4)$$

其中  $k_1(s)$  相当于 A\*算法中的估价函数  $f(s)$ ,  $k_2(s)$  相当于 A\*算法中的启发函数  $g(s)$ 。LPA\*算法中节点  $s$  的搜索启发函数包括两部分: 节点  $s$  到节点  $s'$  的代价  $c(s, s')$  以及  $s'$  到目标点  $s_{goal}$  的启发函数值:

对于地图中任一节点  $s$ ，如果满足  $g(s) = rhs(s)$ ，则称节点  $s$  为局部一致的，而在路径规划时，需要使用一个优先对列来维护局部不一致的节点，并采用类似 A\* 中计算最小估价值的节点进行扩展的方式计算最小  $k(s)$  的节点进行扩展。

LPA\* 算法中的数据结构有：地图数组或图 Graph、节点 Node 以及开启列表 OpenList。节点 Node 含有实际代价  $g(s)$ 、启发函数  $h(s)$ 、 $rhs(s)$  函数、 $key$  函数、节点  $s$  可以去往的节点数组 Children[] 以及可以抵达节点  $s$  的节点数组 Parents[]，算法描述如表 3-5 所示。

表 3-5 LPA\* 算法

算法：LPA\*

1. For each  $s$  in Graph
2.      $s.g(x) = rhs(x) = \infty$ ; (本地一致)
3. end for each
4. startNode.rhs = 0; (溢一致)
5. Forever
6. While(OpenList.Top().key < goal.key OR goal is inconsistent)
7.     currentNode = OpenList.Pop();
8.     if(currentNode is overconsistent)
9.         currentNode.g(x) = currentNode.rhs(x); (使其重新一致)
10.     else currentNode.g(x) =  $\infty$ ; (变为溢一致或一致)
11.     for each  $s$  in currentNode.Children[]
12.         update  $s.rhs(x)$ ; (变为溢一致或一致)
13.     end for each
14. End while
15. Wait for changes in Graph
16.     For each connection ( $u, v$ ) with changed cost
17.         Update connection( $u, v$ );
18.         Make  $v$  locally inconsistent;
19.     end for each
20. End Forever

图 3-7 为三种算法仿真对比，在正常的搜索中 Dijkstra、A\*、LPA\*，分别对应于(a)、(b)、(c)，都能搜索到最优的路径，由于 A\* 与 LPA\* 拥有启发函数，减少了不必要节点的访问，此时两者效果一样。然而对地图动态添加 8 个障碍节点后，

Dijkstra 算法和 A\*算法，对应于(d)、(e)对全局重新做了路径规划，第一次路径规划的信息完全被遗弃，而 LPA\*算法，对应于(f)，采用增量式的路径搜索方法，在添加动态节点后仅仅更新了 4 个节点的信息就重新找到了新的最优路径。由此看出 LPA\*的两大优点：启发式搜索和增量式搜索。然而其规划的路径也有许多弯路，事实上在遇到障碍时可以仅更新当前点到目标点的最短路径即可。

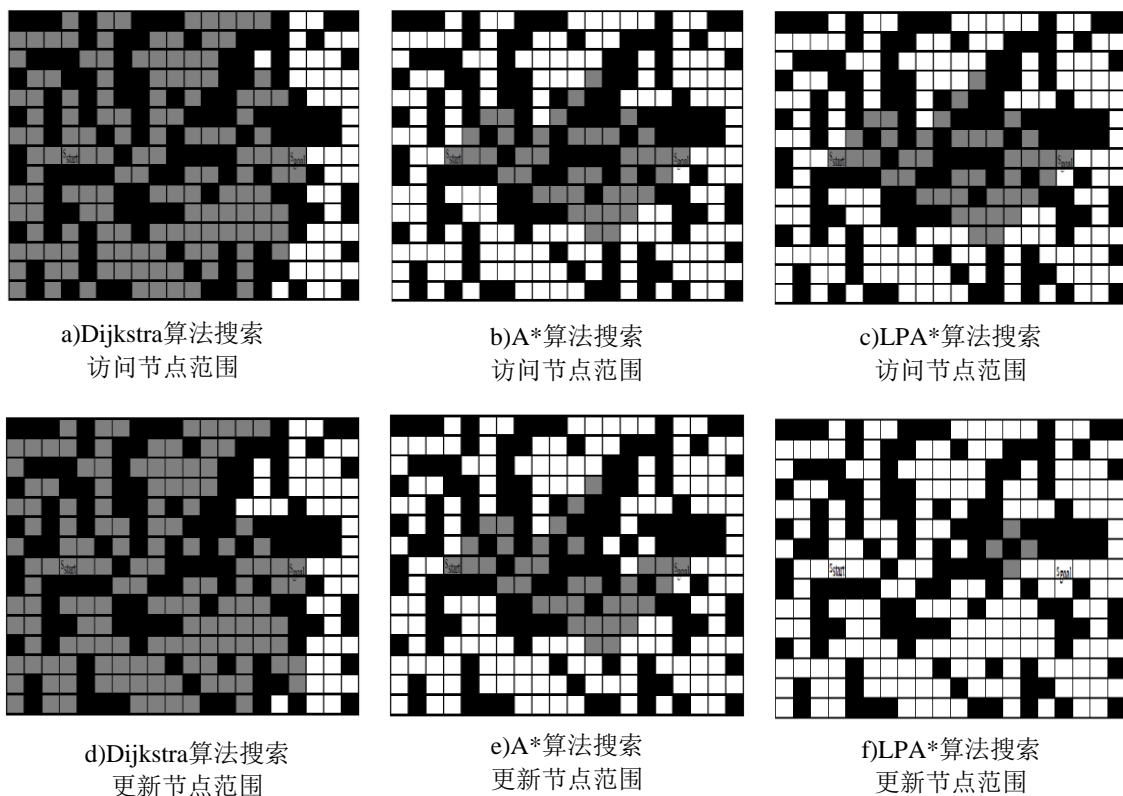


图 3-7 三种算法比较

### 3.2.4 D\* Lite 搜索算法

D\* Lite 算法之于 LPA\*算法犹如 D\*算法之于 A\*算法。与 LPA\*采用的正向搜索算法不同，D\* Lite 采用反向搜索方式，效果与 D\*算法相当。无论是前文提到 LPA\*算法还是 A\*算法都不能满足移动机器人在未知环境中的路径规划需求，因为其在未知地图中需要不断的尝试，与边走边找到最优路径背道而驰。此时反向搜索算法能够很好的处理这种情况，D\*算法虽然可以实现未知环境的路径规划，但效率较低，基于 LPA\*的 D\* Lite 可以很好的应对环境未知的情况，其算法核心在于假设了未知区域都是自由空间，以此为基础，增量式地实现路径规划，通过最小化  $rhs$  值找到目标点到各个节点的最短距离。在移动机器人按照规划的路径进行



前进时其所到的节点即设置为起始节点，因此路经变化或者  $key$  值需要更新时，需要更新从目标点到新起点的启发值以及估计成本。由于移动机器人不断的靠近目标点，节点的启发值将不断减少，且减少至不会超过  $h(startOrg, startNew)$ 。由于我们每次都要减去相同的值，开启列表的顺序并不会改变，因此我们可以不进行这部分的计算，这样便避免了每次路径改变时的队列遍历过程。

若前行过程中发现障碍物则将障碍物所对应环境地图位置设置为障碍物空间，并再以之为起点利用“路径场”信息重新规划出一条路径来。此时不仅更新规划路径的节点数据，也要更新智能体遍历过的节点。其关键在于如何在未知的环境中根据传感器获取的极少周边地图信息来进行最有效的靠近目标点的任务。其实整个靠近的过程一直在扩大已知地图范围，尽可能少的尝试次数来实现完成抵达目标点的任务。如图 3-8 为 D\* Lite 搜索示意图，黑点是在按照反向搜索的路径执行时发现的障碍点，到遇到不能通行的障碍点后便更新地图信息，重新规划出一条新的路径继续前行。

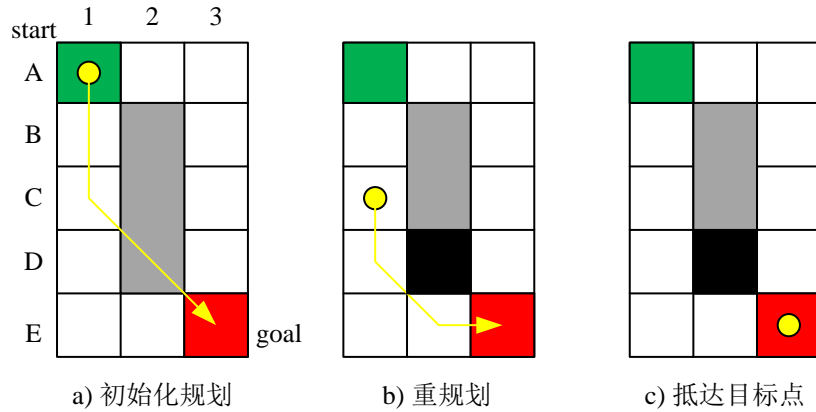


图 3-8 D\* Lite 算法示意图

D\* Lite 算法的原理类似 D\*，起初需要根据已知的环境信息，未知部分视作自由空间，规划出从目标点到起始点的全局最优路径，此时即建立了一个“路径场”信息，为增量靠近目标点提供择优依据。D\* Lite 算法是反向搜索的，因此 LPA\* 里的  $g(s)$ ， $h(s)$  有了新的定义，即分别代表从目标点到当前  $s$  点的代价值以及当前  $s$  点到出发点的启发值。 $rhs(s)$  记录栅格节点  $s$  的父节点的  $g(s)$ ，有公式：

$$rhs(s) = \begin{cases} 0 & s = s_{start} \\ \min_{s' \in Pred(s)} (g(s') + c(s, s')) & otherwise \end{cases} \quad (3-6)$$

在评价栅格点的估价值时 D\* Lite 也引入了  $k(s)$  值进行比较，其中  $k(s)$  包含两个值  $[k(s_1); k(s_2)]$ ，分别满足以下公式：



$$k_1(s) = \min(g(s), rhs(s)) + h(s, s_{goal}) \quad (3-7)$$

$$k_2(s) = \min(g(s), rhs(s)) \quad (3-8)$$

与 LPA\* 算法相对应，可以很容易得出以下公式：

$$h(s, s_{start}) = \begin{cases} 0 & \text{if } s = s_{start} \\ c(s, s') + h(s', s_{goal}) & \text{otherwise} \end{cases} \quad (3-9)$$

D\* Lite 路径规划算法描述如表 3-6。

表 3-6 D\* Lite 算法

算法：D\* Lite

**CalcKey(s)** 计算节点 s 的 key 值

1. 返回  $\min(\min(g(s), rhs(s)) + h(s_{start}, s) + k_m; \min(g(s), rhs(s)))$ ;

**Initialize()** 初始化

2. 开启列表  $U$  设为空， $k_m$  设置为 0
4. 遍历地图节点集  $S$ ，其中一节点  $s$ ，令  $rhs(s) = g(s) = \infty$ ;
5. 对于目标点  $s_{goal}$ ，令  $rhs(s_{goal}) = 0$ ;
6. 计算  $s_{goal}$  的 key 值，并将其加入到开启列表;

**UpdateVertex(u)** 更新节点 u 的信息

7. if ( $u \neq s_{goal}$ ), 令  $rhs(u) = \min_{s_0 \in pred(u)} (c(u, s_0) + g(s_0))$ ;
8. if ( $u$  在开启列表  $U$  中), 从开启列表  $U$  中移除  $u$ ;
9. if ( $g(u) \neq rhs(u)$ ),  $U.Insert(u, CalcKey(u))$ ;

**ComputeShortestPath()** 计算最短路径

10. 当开启列表  $U$  中最小 key 节点小于起始节点 key 值  $CalcKey(s_{start})$   
或  $rhs(s_{start}) \neq g(s_{start})$  循环执行

11. 获取开启列表  $U$  的最小 key 值赋予  $k_{old}$ ;
12. 取开启列表  $U$  顶节点赋予  $u$ ，并从开启列表移除该节点;
13. if ( $k_{old} < CalcKey(u)$ )
14. 将  $u$  加入开启列表  $U$ ;
15. else if ( $g(u) > rhs(u)$ )
16. 更新  $g(u)$ ，令  $g(u) = rhs(u)$ ;

表 3-6 D\* Lite 算法（续）

算法：D\* Lite

17. 遍历节点  $u$  的前向节点集，其中一点  $s$ ，更新节点  $s$  信息  $UpdateVertex(s)$
18. else
19. 令  $g(u) = \infty$  ;//遇到障碍点
20. 遍历节点  $u$  及其  $u$  的前向节点集，其中一点  $s$ ，更新节点  $s$  信息  $UpdateVertex(s)$
- Main()** 主程序
21. 将起始点  $s_{start}$  赋于当前节点  $s_{last}$  ;
22. 执行初始化函数  $Initialize()$  ;
23. 执行计算最短路径函数  $ComputeShortestPath()$  ;
24. 当  $s_{start}$  不为  $s_{goal}$  循环执行
26.  $s_{start} = \arg \min_{s_0 \in succ(s_{start})} (c(s_{start}, s_0) + g(s_0))$  ;
27. 移动至  $s_{start}$  ;
28. 扫描地图，修改节点的开销值;
29. if 所有节点更新完成
30. 令  $k_m = k_m + h(s_{last}, s_{start})$  ,  $s_{last} = s_{start}$  ;
32. 遍历改变开销的有向边  $(u; v)$
33. 更新边开销  $c(u; v)$  ;
34. 更新顶点  $u$  信息  $UpdateVertex(u)$  ;
35. 计算最短路径  $ComputeShortestPath()$  ;

图 3-9 为 D\* Lite 的仿真实现，其中左图为环境地图，右图为维护的路径场信息以及实时规划的路径，R 为起始点，G 为终点。绿色为添加到开启列表被遍历的点。

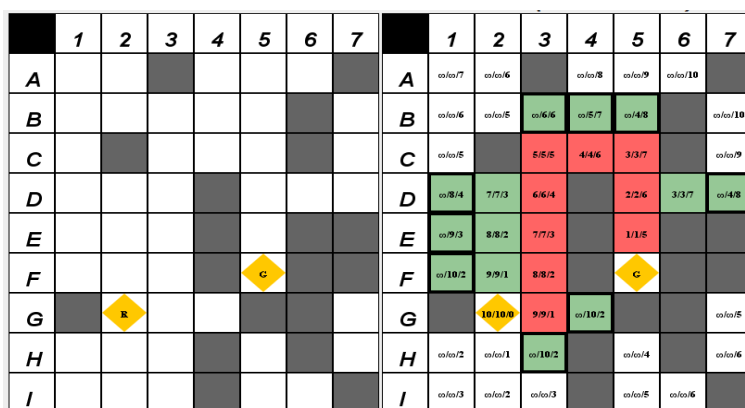


图 3-9 D\* Lite 算法仿真图

### 3.3 局部路径规划方法

#### 3.3.1 人工势场法

人工势场法的基本思想来自物理力学，在对环境地图建模时对障碍物建立斥力场而对目标点建立引力场，环境中目标引力与其他障碍物斥力的所合成的合力构成引导机器人的控制力，如图 3-10。

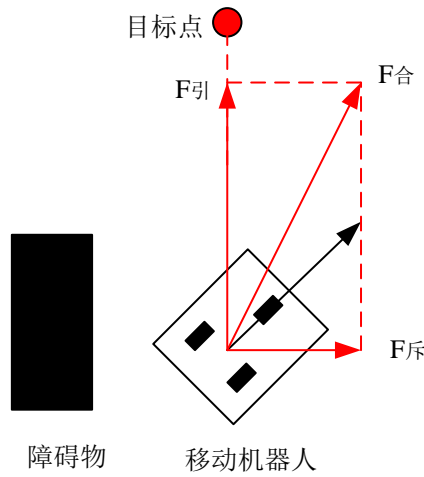


图 3-10 人工势场法原理

人工势场算法：

步骤 1：构建人工势场(Artificial Potential Field)：以目标点为中心建立吸引势场，以障碍物为中心建立推斥势场。

$$U_{rep} = \begin{cases} \frac{1}{2} K_r \left( \frac{1}{\rho} - \frac{1}{\rho_0} \right)^2 & \rho \leq \rho_0 \\ 0 & \rho > \rho_0 \end{cases} \quad (3-10)$$

式中  $\rho$  被评估点和障碍物点之间的距离， $\rho_0$  预定义距离阈值。

步骤 2：根据人工势场计算，对势场求偏导数。

$$F_{att}(x) = -\nabla U_{att}(x) = \begin{cases} -2K_a(x - x_d) & |x - x_d| \leq d_a \\ -2K_a d_a \frac{x - x_d}{|x - x_d|} & |x - x_d| > d_a \end{cases} \quad (3-11)$$

$$F_{rep}(x) = -\nabla U_{rep}(x) = \begin{cases} -2K_r \left( \frac{1}{\rho} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2} \frac{\partial \rho}{\partial x} & \rho \leq \rho_0 \\ 0 & \rho > \rho_0 \end{cases} \quad (3-12)$$

$$\frac{\partial \rho}{\partial x} = \left( \frac{\partial \rho}{\partial x} \quad \frac{\partial \rho}{\partial y} \right)^T = \frac{x - x_0}{\rho} \quad (3-13)$$

$x_0$  为最近障碍物的坐标向量。

步骤 3: 计算合力, 并进而由力计算得到控制律

$$F(x) = -\nabla U(x) = -\nabla U_{att}(x) - \nabla U_{rep}(x) = F_{att}(x) + F_{rep}(x) \quad (3-14)$$

机器人是受人工势场影响的一个点, 沿着势场方向就可以避开障碍物达到目标点, 图 3-11 为人工势场法仿真示意图。

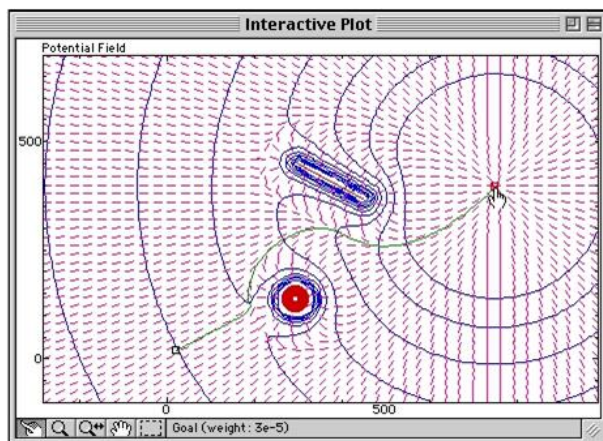


图 3-11 人工势场法仿真示意图

人工势场不仅是一种路径规划方法, 所构建的势场也构成了机器人的控制律, 能够较好地适应目标的变化和环境中的动态障碍物, 其有一个很大的缺点, 那就是: 存在局部最小, 容易产生振荡和死锁。针对这一问题, 目前比较可行的解决方案有: 设计更优的势场函数使其产生局部极小值的概率降低; 结合其他算法, 在遇到极小值点时切换算法解决, 如图 3-12 所示。

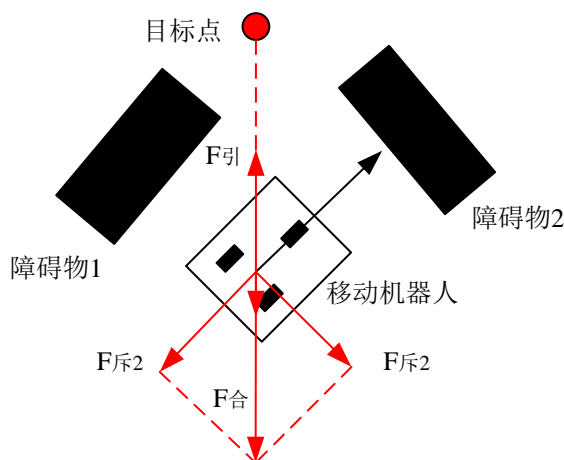


图 3-12 人工势场局部最小值情况

人工势场方法经常用于局部路径规划，由于局部环境信息不易建模和控制，其可以有效避免在小范围内碰到障碍物。

### 3.3.2 神经网络路径规划

神经网络是由神经元组成的并行分布式系统，模拟人脑结构，具有学习和存储知识的能力，通常用于控制领域。神经网络用于解决路径规划问题也是一个很有前景的研究方向，它具有强大的非线性关系处理能力，并在学习过程中不断存储知识，最后达到较好的行为表现。在局部路径规划时，经常要处理障碍躲避的任务，而神经网络能够很好解决这一问题，将神经网络用于局部路径规划具有更好的灵活性，并能很好的解决人工势场局部极小值的问题。

此前已有大量研究人员在将神经网络引入路径规划方面做了很多开拓性工作，其中美国 Carnegie Mellon 大学研究出的基于多层感知机 MLP 神经网络的道路跟踪系统 ALVINN 仍是目前最好的研究成果。ALVINN 全称 Autonomous Land Vehicle in a Neural Network，它的设计思路是 ALVINN 系统在驾驶员驾驶过程中学习不同情况下的操作行为，根据累计学习到的经验使得自己具备自动驾驶的能力。具体实现原理为：ALVINN 系统采用了反向传播作为神经网络的学习方法，在训练阶段，驾驶员在该神经网络模型上进行驾驶训练，ALVINN 采用单隐层的多层神经网络模型，如图 3-13 所示，其输入层为摄像机采集的前方图片信息处理后得到  $32 \times 30$  像素的低分辨率图像，即 960 个输入单元，隐藏层设计为 5 个单元，输出为代表前行的方向 30 个单元，每个输出单元代表一定的角度范围，这样以来就可以很好的细化转向的级别，使得实际操作更加可靠。

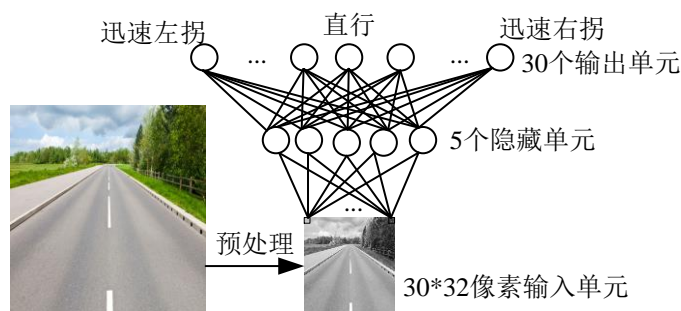


图 3-13 ALVINN 系统神经网络模型

## 3.4 基于采样的规划方法

概率完备路径规划方法主要有两种：概率路标算法 (Probabilistic Roadmap, PRM)<sup>[44]</sup>、快速探索随机树 (Rapidly Exploring Random Tree, RRT)<sup>[45]</sup>。

### 3.4.1 PRM 算法

PRM 的基本思路是采用图  $G(V, E)$  表示 Roadmap,  $V$  为节点,  $E$  为边, 表示节点之间的连通性, 节点  $q$  为机器人的姿态, 必须在空闲姿态空间中边  $(q_1, q_2)$  表示两个机器人姿态之间存在一条无碰路径, 节点之间的度量采用欧氏距离表示, 采用粗采样法采样节点, 精细采样法采样边, 输出在空闲姿态空间中的 roadmap。算法描述如表 3-7 所示。

表 3-7 PRM 算法

算法: Roadmap 构建算法

输入:  $n$ : 需要放入 Roadmap 的节点数  $k$ : 检查最近相邻的节点配置的个数

输出: 构建好的 Roadmap  $G(V, E)$

1.  $V$  置空,  $E$  置空
2. while  $|V| < n$  do
3.   repeat
4.     从姿态空间随机获取一个配置节点  $q$
5.     until  $q$  是无碰撞点 (自由空间点)
6.     添加  $q$  到  $V$
7.   end while
8. for all  $q \in V$  do
9.   从  $V$  空间将节点  $q$  的  $K$  个相邻节点赋给  $N_q$
10.   for all  $q' \in N_q$  do
11.     if  $(q, q') \in E$  and  $\Delta(q, q') \neq null$  then
12.       将  $(q, q')$  添加到  $E$
13.     end if
14.   end for
15. end for

PRM 算法示意图如图 3-14, 首先需要在随机姿态随机取点, 此时需要均匀随机采样。然后对采样的点进行碰撞检测, 去掉障碍点, 然后将每个节点与相邻节点连接。连接相邻节点时在姿态空间寻找最近邻点采用穷举搜索成本为  $O(n)$ , 成本过大, 可采用 K-D 树搜索, 当搜索空间为二维时, 其成本为  $O(\sqrt{n})$ 。接着对节点间连接进行无碰检查, 保留无碰连接。对于无碰检查有增量式检查和二分检查两种, 如图 3-15, 其中二分检查较优。最后加入起始点和终点, 找出一条通路即完成路径规划。

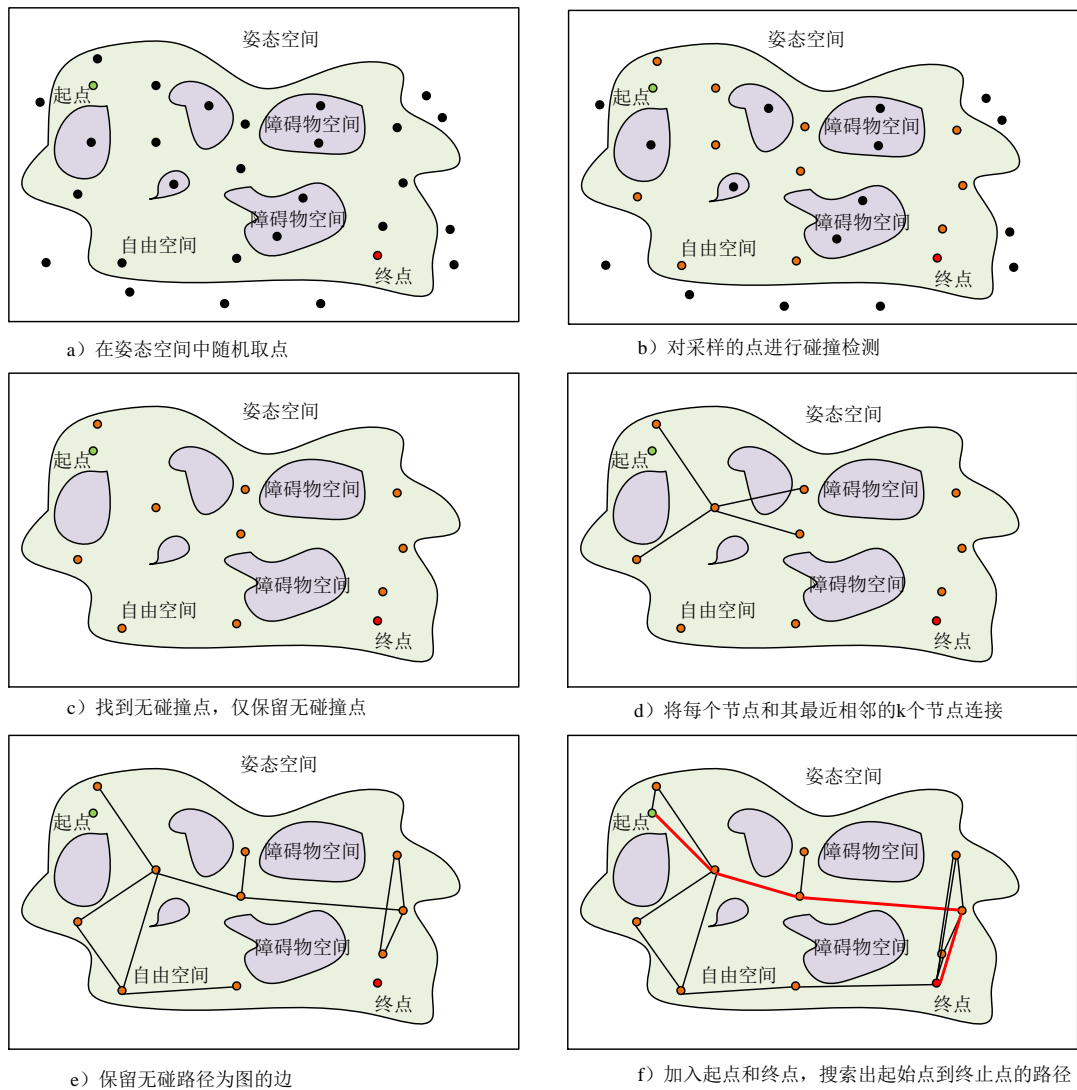


图 3-14 PRM 算法示意图

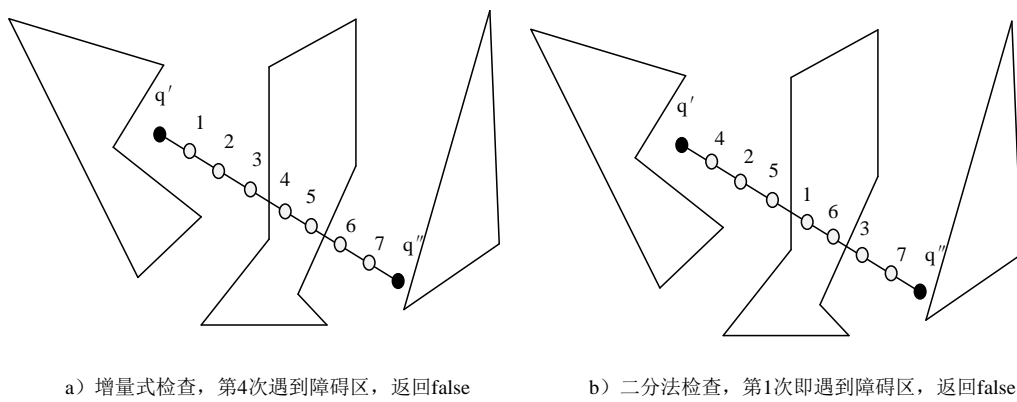


图 3-15 PRM 无碰撞检测的两种方法比较

PRM 后处理路径优化的方式有两种：贪婪优化法以及随机优化法。前者是从目标点开始依此连接非相邻姿态，检查是否存在无碰路径，找到最远的无碰连接姿态，依次向前迭代搜索，后者随机选择非相邻姿态。

### 3.4.2 RRT 算法

与 PRM 的在连接图之前首先对整个配置空间进行采样不同，RRT 从单个点向外生长搜索。从起始节点开始维护一棵可达配置树。在扩展阶段期间，来自配置空间的随机配置  $q_{target}$  被选中。计算树上最近的节点（通常由欧几里德距离测量） $q_{nearest}$ ，并且尝试通过将新的节点  $q_{node}$  放置在从  $q_{nearest}$  指向  $q_{target}$  的方向上将树向目标点生长长度为  $d$  的距离。碰撞检测功能用于检查新节点是否满足全局约束，并且如果朝向  $q_{target}$  延伸需要经过障碍物，则不发生扩展。重复该过程，直到树达到目标位置可达范围内的阈值距离，如图 3-16。

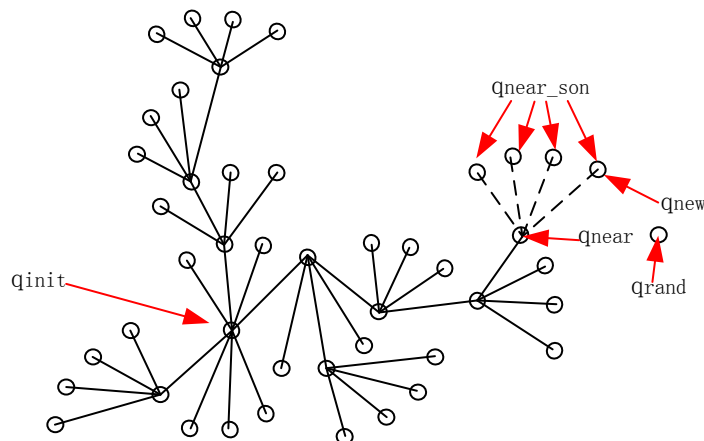


图 3-16 RRT 原理示意图

RRT 连接算法增量地构建以开始点和目标点配置为根的两个随机树。树向着彼此生长，直到最后一个分支添加到某一棵树且成为了另一棵树进行扩展的  $q_{target}$ 。现场机器人环境中许多工作都是基于此来应用 RRT 的。树是通过测量候选节点的“障碍属性”并且将其与将搜索的易于导航的可遍历地形阈值进行比较来扩展的。该阈值向上修正，直到找到一条路径。Urmson 和 Simmons 引入了一个测量因子，即将路径的成本纳入 RRT 算法，目的是在复杂的户外地形中也可以应用 RRT 进行导航。类似地，Ferguson 和 Stentz 在非均匀成本环境中融合 RRT 与 Anytime 算法以产生响应于初始查询来生成一系列 RRT 的系统。每棵树重复使用先前搜索的结果以产生越来越好的路径。RRT 的算法描述如表 3-8，仿真图为 3-17。



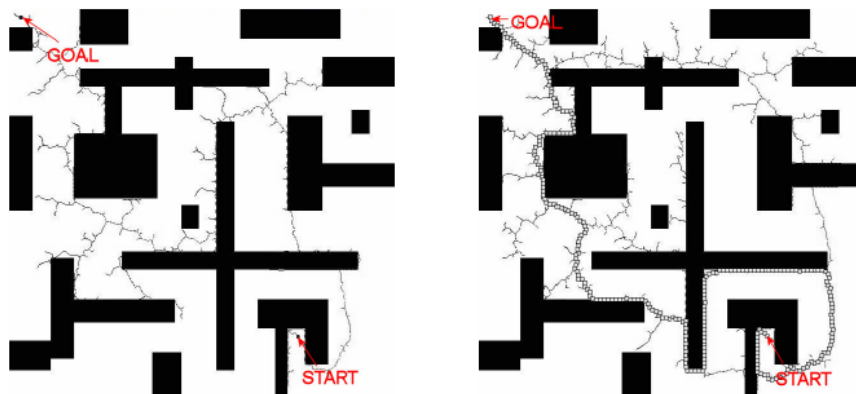


图 3-17 RRT 仿真示意图

表 3-8 RRT 算法

算法: RRT\*

1. 将  $\{x_{init}\}$  赋予  $V$ ,  $E$  置空
2. for  $i = 1, \dots, n$  do
3.     将  $\text{SampleFree}_i$  赋予  $x_{rand}$
4.     将  $\text{Nearest}(G(V, E), x_{rand})$  赋予  $x_{new}$
5.     if  $\text{ObstacleFree}(x_{nearest}, x_{new})$  then
6.          $V$  添加  $x_{new}$ ,  $E$  添加  $(x_{nearest}, x_{new})$
7.     end if
8. end for
9. return  $G(V, E)$

影响规划收敛速度的三个步骤：随机状态的采样、在搜索树中查找与随机状态距离最近的节点以及新生成节点的防碰检测。RRT 改进算法有双向搜索 Bidirectional-RRT 和 RRT\*。其中 RRT\* 为实现渐近最优，考虑路径成本，增加最近点领域内路径最短的节点到新节点的路径，将新节点的父节点更改为最近点领域内路径最短的节点。

随机抽样方法的缺点是，它们规划出来的路径也是随机的，该类算法的变体倾向于产生复杂的锯齿状路径，具有许多不必要的路径段，在它们被送到机器人的运动控制模块之前仍需要某种形式的后处理。因此，基于网格地图的规划技术仍然是主要的移动机器人路径规划方式。

### 3.5 本章小结

本章节对各种路径规划算法做了详细的研究，包括基于图搜索算法、局部路径规划算法以及基于采样规划算法。本章节从算法演变的角度分析了路径规划的产生原因，重点描述了他们的使用场景。并对典型的算法重现其推理过程进行实验仿真和对比，分析出其中内在的机理。每种算法都有其优缺点，在实际应用中应观察具体场景，综合考量多种路径规划算法的特点来为下一节提出的融合算法设计提供依据，从而设计出特定场景下效率高且灵活易用的路径规划方案。

## 第 4 章 D\* Lite 与神经网络融合路径规划研究

### 4.1 引言

前文讲到路径规划算法各有优缺点，更接近真实的场景是移动机器人在环境未知的大地图中进行路径规划，前文讲到的 D\* Lite 算法能够很好的适用于未知环境做路径规划，由于其增量规划的思想，它可以做到较少重规划次数以及较少的重规划影响节点数。对于实际情况两个栅格节点的能否抵达通常需要根据局部信息做局部的路径规划，前文有介绍人工势场法以及神经网络算法，前者存在有局部极小点的情况，而此时可以通过调整势场函数来降低局部极小点出现的概率但却带来了复杂的计算公式，并且由于其缺少对一些常识判断来规划，依然具有一定的盲目性。而能够很好处理非线性映射关系的神经网络可以很好的处理这一问题，因此在局部采用神经网络结合全局的 D\* Lite 路径规划算法是一个很好的融合路径规划算法。

### 4.2 D\* Lite 算法的不足

当状态空间比较大，也就是环境地图比较大的时候，采用的 D\* Lite 路径规划算法的反向搜索过程需要维护的栅格节点数急剧增加，增加了搜索的时间复杂度。除此之外，D\* Lite 路径规划不能应对环境复杂的情况，即局部环境的精细规划。对于大环境下的路径规划，D\* Lite 算法的做法是将环境地图进行更细粒度的栅格化，虽然在足够细粒化的环境地图中可以实现较优的路径解，但也会带来更多的规划序列导致执行次数以及重规划次数增多，进而路径规划执行花费时间也会变得更长。比较好的解决方案是采用粗细结合的方案，即维护两种分辨率地图，上层地图采用粗粒度的高分辨栅格地图表示，下层即栅格单元采用细粒度的地分辨栅格分解。首先进行上层地图的 D\* Lite 路径规划，在执行规划序列时必要的情况下加载下层栅格地图进行精细 D\* Lite 路径规划。这样能够将规划问题进行分解，分为高层粗粒度路径规划与底层细粒度路径规划两部分，两层的路径规划的维护的地图栅格数有所降低，会带来一定程度的时间复杂度的降低。但是总的栅格节点仍然是那么多，并不能带来存储空间的优化，而且其又带来了结构复杂的问题。并且依然并不能解决局部环境下的精准细密的规划问题，即在复杂环境下不能根据环境特征进行慎思规划。

由于复杂的大环境下的路径规划需要考虑复杂的环境情况，需要对局部路径

规划进行相应的优化。可以看到基于 D\* Lite 算法的路径规划在局部规划中是一种盲目性的试错路径规划，如图 4-1。在复杂环境中会带来更多的时间效率损耗，而如果移动机器人能够像人一样可以分析周边环境并根据目标点情况并作出合理的方向判断无疑可以大大提高局部路径规划的准确性，从而大大提供规划的效率。

分析可以发现移动机器人的状态、周边的环境信息以及目标点位置信息与“理性”的判断之间是非线性关系，需要移动机器人适应环境并对环境进行预知学习才能拥有足够的信息去做合理的判断，而这正是神经网络的强项。对于 D\* Lite 算法在局部路径规划处理复杂环境的不足，本文提出了一种融合型路径规划算法，即将环境分全局和局部两层，使用 D\* Lite 算法解决全局路径规划，使用神经网络解决局部路径规划。具体的实现策略将在下一节详细描述。

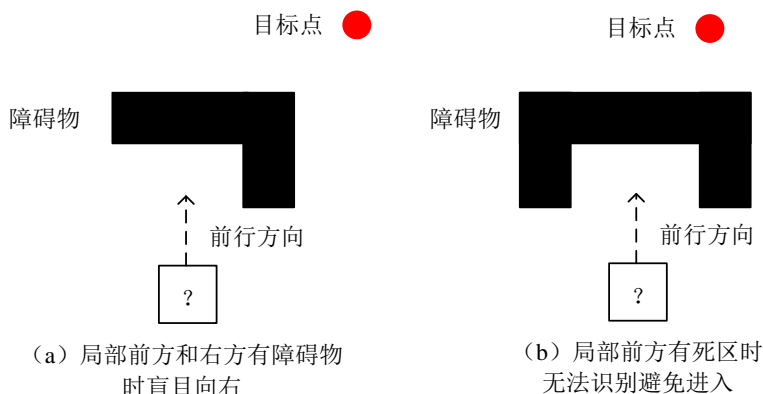


图 4-1 D\* Lite 不能处理的情况

### 4.3 D\* Lite 和 BP 神经网络融合路径规划

根据上一小节的分析，我们要实现的策略是：全局采用低分辨率地图减少状态空间，使用 D\* Lite 算法进行路径规划，得出一组路径节点序列，然后移动机器人开始按照节点序列依次前行。在整个行走过程中，移动机器人的激光传感器可以不断的扩大已知地图的信息，移动机器人可以根据这些信息在局部路径规划中采用神经网络进行规划，通过增强学习算法的训练使得移动机器人能够很好的处理全局路径规划出的相邻两个节点的抵达问题。

融合算法的具体实现需要做一些合理的先决条件假设：首先假设移动机器人任何时候都能知道自己所在全局栅格的节点位置以及局部的度量地图的坐标位置，其次移动机器人能够使用激光传感器准确地获取环境中障碍物的距离。很明显，前者是对定位的假定，后者是对传感器的假定。

根据上文描述，融合算法路径规划对环境地图建模采用的是第 2 章所介绍的度量-拓扑分层结构的混合地图表示法。其中全局为栅格地图，首先确定地图的大

小，按照一定的分辨率将环境地图分为  $m*n$  个栅格，总数每个栅格点记录本栅格地图的障碍占据密度以及本栅格节点到其他 8 个相邻节点代价值。这些数据在执行路径规划时统一初始化为可抵达的定值，在随着移动机器人的不断前进过程中，通过传感器获取并更新。对于全局栅格地图中任意栅格节点  $X$ ，其包含的信息有：当前栅格是否被障碍物占据、其相邻的 8 个栅格节点是否可达及可达的状态转换代价  $c(X,Y)$ ，其中  $Y$  指  $X$  相邻节点  $Y$ 。对于局部的度量层需要移动机器人传感去建模，以每个栅格为单元，为局部做度量地图建模。因此，算法需要维护两个地图：全局栅格地图  $Gmap$  以及局部度量地图  $Lmap$ ，前进时交替执行两种路径规划算法：全局为  $D^*$  Lite，局部为神经网络。对于地图中任一点  $X$  其坐标数据表示为：全局坐标  $G:int X, int Y$ ，局部坐标  $L:float x, float y$ ，如图 4-2 所示。

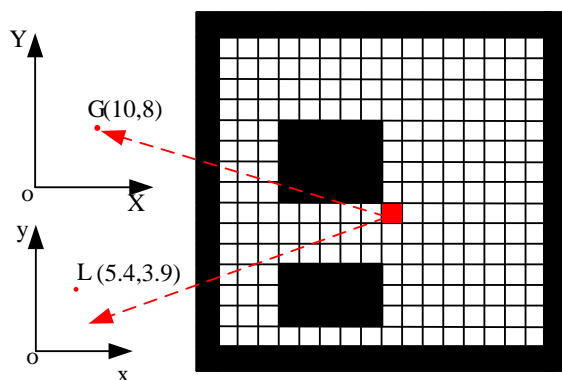


图 4-2 两层坐标表示

接下来将对神经网络解决度量移动机器人在地图中的路径规划进行详细讲解。当栅格点  $Y$  是栅格点  $X$  的 8 个相邻节点的一个节点时，从栅格点  $X$  到栅格点  $Y$  的可达性表示有两种，如果栅格  $X$  与栅格  $Y$  共享一条边， $X$  可以通过共享边抵达  $Y$  则为  $(X,Y)$  可达，另一种情况是  $XY$  只共享一个顶点，即  $X$  与  $Y$  对角， $X$  借助相邻节点能够抵达  $Y$  则为  $(X,Y)$  可达，如图 4-3。

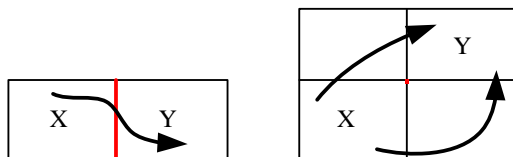


图 4-3  $X$  到相邻节点  $Y$  的可达性定义

本论文设计的神经网络采用的  $BP$  网络算法，节点激活函数选取标准的 *Sigmoid* 型函数，并参考了前文提到的 *ALVINN* 系统的单隐藏层神经网络结构，对于位于  $X_i$

位置的移动机器人，在全局路径规划出的栅格序列中可以了解到其下一个局部目标为  $X_{i+1}$ ，映射在局部的度量地图中为  $X_i$  到  $X_{i+1}$  最近的可达坐标。本论文设计的神经网络系统采用  $X_i$  到  $X_{i+1}$  的角度和距离作为其中 2 维输入，另外还有来自移动机器人激光传感器采集到前方 180 度范围的障碍物距离信息，这部分数据有 181 个，因此输入层为 183 维的输入向量。隐藏层内节点的设置对于网络的性能影响很大，且不是唯一的，目前对隐藏层节点数的选择大多数情况下还是靠经验，隐藏层阶数太少可能会导致局部极少增多，网络训练不出来。隐藏层节点太多又会导致训练学习时间太长，经过实际训练，考量收敛速率、规划准确性以及反应时间等多个因素本论文对隐藏层设置为 6 个单元。输出层与 ALVINN 系统类似采用 36 个输出单元分别代表不同方向角度区间的推荐值，如图 4-4 所示。

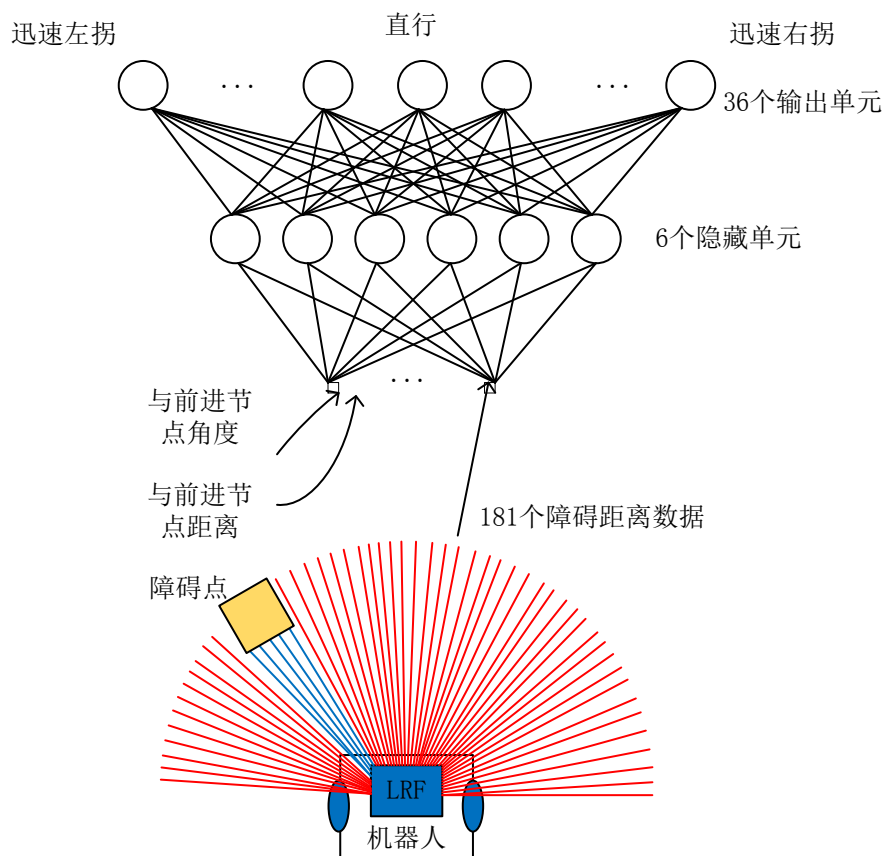


图 4-4 神经网络结构

为了使移动机器人能够自主的探索未知地图，神经网络在训练时需要采用增强学习方法的非监督神经网络。增强学习采用奖赏或称为回报函数对每次操作进行结果分析反馈给学习系统，在这个过程中能够实现神经网络的自学习。这正是我们想要达到的目标：移动机器人通过自学习完成对环境的认知，并能迅速对未知环境下的局部路径规划有着经验和理性认知。如图 4-5 是增强学习状态图。

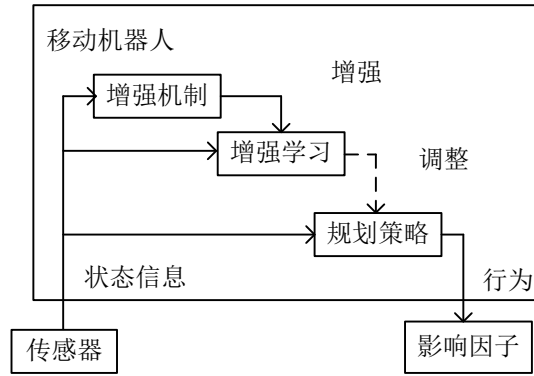


图 4-5 增强学习状态图

传感器从环境中获取环境状态数据，移动机器人的行为会对环境状态产生影响称为影响因子，规划策略制定新的为移动机器人行为集，而增强机制为每一次行为执行生成一个增强信号，增强学习模块对环境状态信息以及增强信号产生反馈，进行学习进而调整规划策略。

局部路径规划的结果是激活一个角度输出，表示移动机器人沿该角度方向前行距离  $s$ 。设一次局部路径规划的目标为  $s_g$ ，则在实际规划时， $s_g$  是一个可达的全局规划路径中下一个栅格内的一点。又假设移动机器人一次局部路径规划的起始点为  $s_t$ ，则增强学习神经的回报函数可以表示为：

$$reward(s_{jt}) = r_t + \sum_{i=1}^m (\gamma^i \times r_{t+i}) + p_t \quad (4-1)$$

$$r = c_1 \cos(\max(\overrightarrow{O_k}, \overrightarrow{O_k}) + (\overrightarrow{s_t s_g} - \overrightarrow{s_{t+1} s_g})) \quad (4-2)$$

$$r_{t+i} = \begin{cases} rwd & \text{if } (s_{t+1} = s_g) \\ \gamma r_{t+i+1} & \text{otherwise} \end{cases} \quad (4-3)$$

公式 (4-1) 中  $s_{jt}$  指移动机器人在  $s$  位置  $t$  时刻时选择第  $j$  个神经元所表示的方向，回报值包括瞬时回报部分  $r_t$  和累计回报部分  $r_{t+i}$  两部分。回报衰减率由  $\gamma$  表示，调节两部分回报值的权重。 $p_t$  则是表示移动机器人执行运动后，其与障碍物的距离的最小值低于了预设的阈值  $\theta$  所获得的惩罚值。瞬时回报部分  $r_t$  由远离障碍物程度与靠近目标点程度两部分组成，前者表示为前行方向与避开障碍物方向的夹角余弦值，后者指靠近目标点的增量距离， $c_1$  为调节两者权重的系数。累计回报部分  $r_{t+i}$  分为两种情况，当执行结果抵达目标点时，一次给以  $rwd$  的奖励，否则按照  $\gamma$  的比率来衰减回报值。

$$reward(s_{kt}) = \frac{1}{|j-k|} reward(s_{kt}), (j - \left\lfloor \frac{m}{2} \right\rfloor \leq k \leq j + \left\lfloor \frac{m}{2} \right\rfloor, j \neq k) \quad (4-4)$$

式(4-4)为移动机器人在 $s$ 位置 $t$ 时刻时选择第 $j$ 个神经元所表示的方向后对其周边第 $k$ 个神经元所表示的方向的回报值影响,根据公式可以看出,其影响程度随距离而减弱。

在使用BP神经网络进行局部路径规划中涉及的参数有:地图栅格数 $n$ ,一次完整局部路径规划的最多行动次数 $t$ ,也表示局部路径规划的频繁程度,每次移动的距离 $s$ ,增强学习中累计回报衰减率 $\gamma$ ,瞬时回报中远离障碍物程度与靠近目标点程度的权重系数 $c_1$ ,一次执行 $reward$ 值受影响的输出神经元个数,用于调整误差,还有算法的学习率 $\eta$ 。这7个参数可以构成一个元祖 $(n, t, s, \gamma, c_1, m, \eta)$ ,这个元祖决定了神经网络模型。采用梯度下降和链式求偏导的方法实现信息前向传播,误差反向传播的非监督神经网络。

本文将提出的融合路径规划算法,其主要步骤及解释如表4-1所示。

表4-1 融合路径规划算法

输入:	$s_g$ : 路径规划的目标点	$g_m$ : 全局栅格地图	$l_m$ : 局部度量地图
输出:	$s_{out}$ : 路径规划序列		
1.	加载神经网络模型		
2.	执行基于D* Lite算法的全局路径规划,生成路径规划序列,序列的数据结构为双向链表,从头结点到尾节点对应起始节点到目标点的节点序列 $(x_1, x_2, \dots, x_N)$		
3.	for i=1 to N-1		
4.	if(链表头结点 $x_h$ 不为目标点 $x_N$ )		
5.	取出链表头结点 $x_h$ 与序列中相邻的下一节点 $x_p$		
6.	while(移动机器人所在位置不在节点 $x_p$ 的区域)		
7.	更新局部地图		
8.	对于 $(x_h, x_p)$ 两节点,用神经网络模型根据局部地图进行局部路径规划		
9.	if( $x_p$ 不可达)		
10.	计算代价训练模型		
11.	将当前节点作为新的全局起始点		
12.	返回2执行replanning,		
13.	else		
14.	if(执行局部规划次数 $count$ 小于最大可执行次数 $t$ )		
15.	执行局部路径规划的结果,即向激活的一个角度方向移动 $s$ 长度		
16.	else		
17.	计算代价训练模型		
18.	将当前节点作为新的全局起始点		



表 4-2 融合路径规划算法（续）

输入：  $s_g$ ：路径规划的目标点  $g_m$ ：全局栅格地图  $l_m$ ：局部度量地图

输出：  $s_{out}$ ：路径规划序列

19. 将当前节点作为新的全局起始点
20. 返回2执行replanning,
21. end while
22. else 抵达目标点
23. 结束本次路径规划

上文我们已经得出了 BP 神经网络的模型以及增强学习方法，接下来最为关键的就是如何训练模型，使之学习环境知识，最终满足我们的局部路径规划需求。起初我们将为神经网络的各个神经元赋以较合理的随机值，并指定一次路径规划的最大可执行次数  $t$ 。如果在  $t$  次操作内不能完成全局规划好的当前节点移动到下一个节点的移动任务则使用 D\* Lite 对全局进行重规划。每次路径规划都会根据传感器得到的地图数据信息以及学习算法不断计算每个输出方向节点的回报值，并以瞬时回报和累计回报之和作为目标函数，在成功抵达下一节点时，回报值会增加，并与前面最多  $t$  次操作构成了训练样本集。因此，训练可以是离线也可以是实时的。

#### 4.4 神经网络的优化与改进

事实上，BP 算法存在不少问题，例如存在不少局部极小值、隐含层节点个数选择困难以及算法的收敛速度很慢等。前两者需要根据实际的情况按照经验合理的设置和调节来解决，在目前没有统一而完整的指导理论。对于最后一项，可以在神经网络中加入动量项使学习速率自适应来解决。带有动量加权的调节公式为：

$$\Delta\omega(t+1) = -\eta * \frac{\partial E}{\partial \omega} + a * \Delta\omega(t) \quad (4-5)$$

动量项的引入可以使得  $\eta$  值可以随学习过程而变化，进而可以加快网络的收敛速度，使系统尽快脱离误差饱和区而迅速进入平坦区，加快学习的步伐。然而参数  $\eta$  和  $a$  却只能靠实验来确定。第二种优化方案是采用学习速率渐小法，即网络初始化时给学习速率一个较大的值  $\eta(0)$ ，训练期间使用如下公式使其逐渐变小。

$$\eta(t) = \eta(0) / (1 + \frac{t}{R}) \quad (4-6)$$

式中  $t$  为迭代次数， $R$  为常量参数，主要用于调节训练周期，学习  $R$  步之后，

学习速率减半。

## 4.5 本章小结

本章首先分析了 D\* Lite 算法在未知环境条件下的不足之处,将路径规划以及环境地图分别分层的概念进行处理,前者全局采用 D\* Lite 局部采用 BP 神经网络算法,后者全局采用低分辨率的栅格地图局部采用度量地图,实现了一种融合型路径规划策略。为了实现移动机器人能够实时在线训练,神经网络采用增强学习算法进行训练。章节中给出了具体的实现方案以及优化策略,具体的实验仿真及性能分析将在下一节讲述。

## 第 5 章 基于 MRDS 平台的仿真实验

### 5.1 引言

MRDS 平台是微软推出的机器软件搭建及仿真平台, 已为很多机器人提供了仿真模型和编程接口, 机器人开发人员可以很方便的使用 Visual Studio 开发工具在 MRDS 上开发软件下载到本地机器人进行调试以及使用 MRDS 提供的仿真模型对软件进行仿真。借助 MRDS 提供的可视化仿真环境, 可以使用户快速搭建自己的移动机器人模型而不必关注于购买机器人模块与模块的驱动等非研究内容上面。在验证本文设计的算法实现上, 使用 MRDS 提出的服务概念可以很方便的实现, 即将算法封装为一个服务, 读取外界信息输出规划指令。本章详细介绍了仿真平台的搭建、仿真实验的实施以及融合算法的验证, 采用量化的方式分析了算法的实际效果, 并验证了算法的正确性与可行性。

### 5.2 实验平台简介

MRDS 是微软发布的一款用于开发机器人程序的开发环境, 一经推出就受到业余爱好者, 学术和商业开发人员的青睐。MRDS 为机器人应用开发提供了一个高性能多任务开发环境以及一个由它所支持的许多真实机器人的三维模拟环境。简而言之, MRDS 是一个具有轻量级 REST 风格的、面向服务、可视化创作的机器人模拟工具, 具体包含了并发与协调运行时 (Concurrency and Coordination Runtime, CCR)、分散式软件服务 (Decentralized Software Services, DSS)、可视化编程语言 (Visual Programming Language, VPL)、可视化仿真环境等组件 (Visual Simulation Environment, VSE)。

CCR 负责为 MRDS 提供多任务处理机制, 是一个完全基于 .NET 通用语言运行时 (Common Language Runtime, CLR) 构建的非常有效的一个轻量级的运行时。它的作用就解决编程时各个组件的通信、同步以及信息一致性问题。很好的理解 CCR 对于你使用 MRDS 编程很重要。如图 5-1 所示, CCR 的基本结构如下所示: CCR 主要包括三部分: 首先是 Port 和 PortSet, CCR 通过 Port 来传递消息 (Messages), 而 PortSet 是可以传递多种不同的 Port 的集合类, 消息在 Port 中是以队列 (Queue) 的方式来存储的。当一个消息丢给某个 Port 后, 一般通过接收器 (Arbiter.Receive) 来接收, 然后启动该接收器的 Handler。可以将 Port 和 PortSet 简单的理解为 CCR 的消息存储读取接口。其次是 Dispatcher, DispatcherQueue 以及

Task, 类似于 Thread, CCR 使用 ITask 来实现多线程的, DispatcherQueue 是一个存放多个 Task 的队列, 而 Dispatcher 是执行 Task 的线程池。最后是 Arbiter, 它是 CCR 提供的组件协同工作的工具类, 里面提供了很多用于处理组件协同工作的工具方法。其中 Arbiter.Recive 用于产生一个 Receive Task, 即当 Port 收到消息时, 执行其中定义的 Handler 处理函数。Arbiter.Activate 使用某个 DispatcherQueue 执行一系列的 Tasks。Arbiter.Choice 有类似于 switch case, 用于处理 Task 分支流程, 它会生成一个 Choice Task, 选择 Handlers 中的一个执行。Arbiter.FromHandler, Arbiter.FromIteratorHandler 将 delegate function 转成 Task, 供 DispatcherQueue 使用。Arbiter.Interleave 是 CCR 的锁机制, 用于处理并发资源同步问题。Arbiter.JoinedReceive 等候两个 Port 都分别收到消息时作相应处理。Arbiter.MultipleItemReceive 一定数量的消息送到一个或多个 Port 的时候就会执行相应 Handler。

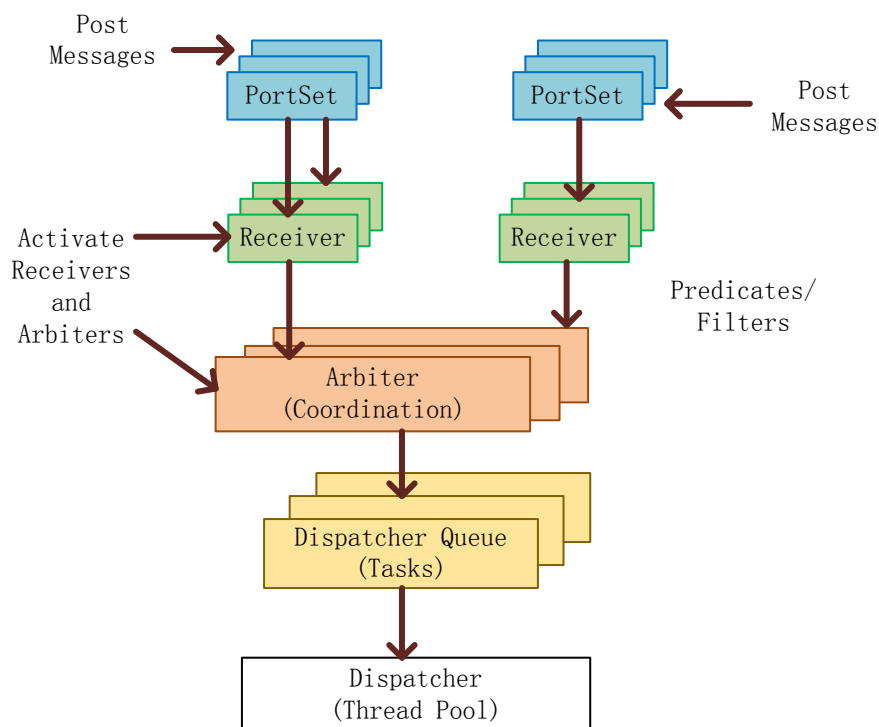


图 5-1 CCR 基本结构

DSS 提供了一个允许网络上多个服务间交换消息的运行环境, 如图 5-2。它内部使用了 CCR 来实现, 但与 CCR 不同的是它超越了 CCR 的单个进程内并行运行传递消息的限制而扩展至进程间甚至多个机器人间的通信。一个 DSS Service 需要有两个名称来进行辨识区分, 一个是 Service Identifier, 表示该服务所存在的统一资源定位器 (Uniform Resource Locator, URL) 位置, 另一个是 Contact Identifier

用于服务间通信时区别不同服务。使用 DSS 构建的应用程序由并行运行的多个独立服务组成。每个服务又包含与其相关联的状态和它接收的操作消息的类型。当服务接收到消息时，它可以改变其状态，然后向其他服务发送消息和通知。开发者可以通过向服务发送 Get 消息来获取服务的状态，或者使用 web 浏览器读取和显示服务的状态。服务可以订阅其他服务的状态改变或其他事件发生的通知来与其他服务协作。开发者需要以不同的思维来基于 CCR 和 DSS 编写程序。对于 DSS 需要考虑异步处理消息的独立代码块，而不是函数调用和线程。DSS 采用服务导向架构（SOA），可以很方便地访问自带或自己设计的服务（Services），并可以使用 Web 浏览器或窗口应用监控机器人的运动状态。

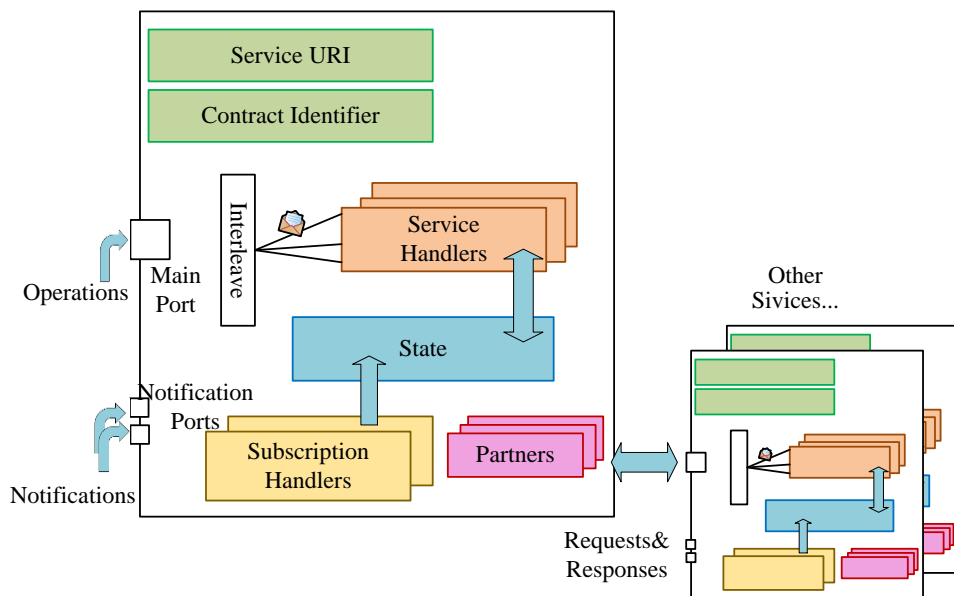


图 5-2 DSS 基本结构

VPL 是 MRDS 提供的可视化编程语言，可以在不编写单行代码的情况下实现服务编排，通过在可视编程语言环境中使用服务连线的方式实现在服务之间的传输数据。在 MRDS 应用程序中，通常是几个服务在运行，其中一些是直接和硬件接口的低级服务，另外一些是直接连接到模拟引擎的模拟服务。一个 MRDS 应用通常是一个或多个顶级服务控制一个或多个机器人的行为。这些顶级服务通常仅与其他服务进行接口通信，又称业务流程服务。VPL 可以直观的表现服务间的消息传递流程，给机器人应用开发带来了很大的便捷。

VSE 是 MRDS 为机器人应用开发提供的 3D 仿真环境及物理仿真，仿真环境包含室内和室外场景，以及各种模拟机器人。它是可扩展的，开发者可以添加自定义的场景对象以及机器人，它有一个内置的编辑器，可以在模拟器运行时添加和移动对象。该模拟器使得在实际的机器人硬件上运行新的机器人算法之前可以

通过仿真排错，避免硬件损耗。目前模拟环境已经包含几个机器人的模型，例如乐高 NXT、iRobot、Pioneer 3DX 和 KUKA LBR3 机器人手臂等。还提供了诸如相机，天空，地面和各种其他对象的模拟实体。

本论文采用的是 MRDS 中的 Pioneer 3DX (P3DX) 进行实验的，P3DX 是一种先进的科研移动机器人，可以板载 PC，具有一系列传感器（包括激光测距仪），并可以通过 WiFi 进行通信。

## 5.3 实验环境搭建

### 5.3.1 移动机器人建模

实验的智能体选用的是 Pioneer P3DX，如图所示，它具有差分驱动配置，这可以使机器人能够移动到任意位置，激光测距传感器，如图 5-3，图 5-4，可以测量前方 180 度的障碍物距离数据，压力传感器检测车身是否接触障碍物。该机器人开发工作室提供了一种称为 ArcosDrive 的服务实现抽象驱动的协调，具体操作有：单独设置的左，右轮速度，旋转指定的角度，行驶指定的距离。其状态更改时，包括车轮速度和驱动是否已启用，驱动器服务都会发送相应的通知。

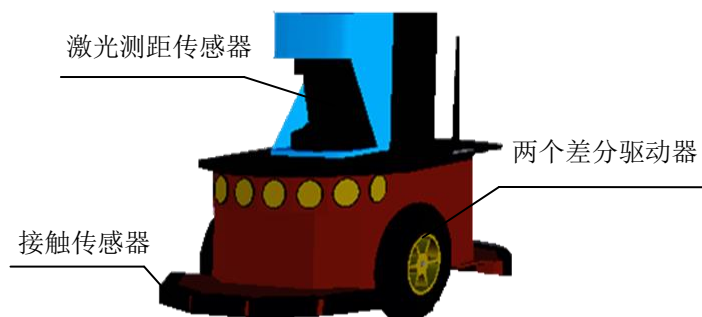


图 5-3 Pioneer 3DX (P3DX) 机器人模型

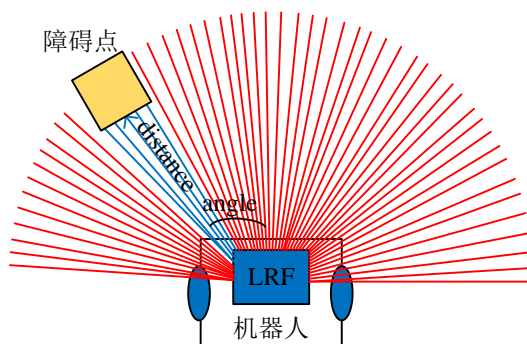


图 5-4 SICK 激光测距仪

P3DX 的路径规划算法封装在 Explorer 服务中, Explorer 服务实时收集激光数据、小车速度、角度以及位置数据, 并根据内部计算出的路径规划指令序列不断驱动机器人靠近规划目标。按照前文的算法的实现描述, 在本文研究的实验中移动机器人实体并没有使用压力传感器, 本文设计的移动机器人导航系统中主要组件有激光测距传感器 Sick LRF 与差分驱动器 Differential Drive 两个。Sick LRF 是移动机器人的“眼睛”, 移动机器人起步时需要旋转一圈, 利用 Sick LRF 对周边局部环境有初步认知, 整个过程一直维护着一个不断更新的地图。当 Explorer 服务规划出全局路径序列后, 接着将根据当前获取的局部地图信息进行局部路径规划, 局部路径规划的结果是一组角度序列。然后移动机器人差分驱动将依次按照规划的角度序列前行固定长距离。如果局部路径规划时发现前方遇到的障碍物致使不能完成局部路径规划任务则通知系统, 对全局地图进行重新规划。移动机器人导航系统结构如图 5-5。

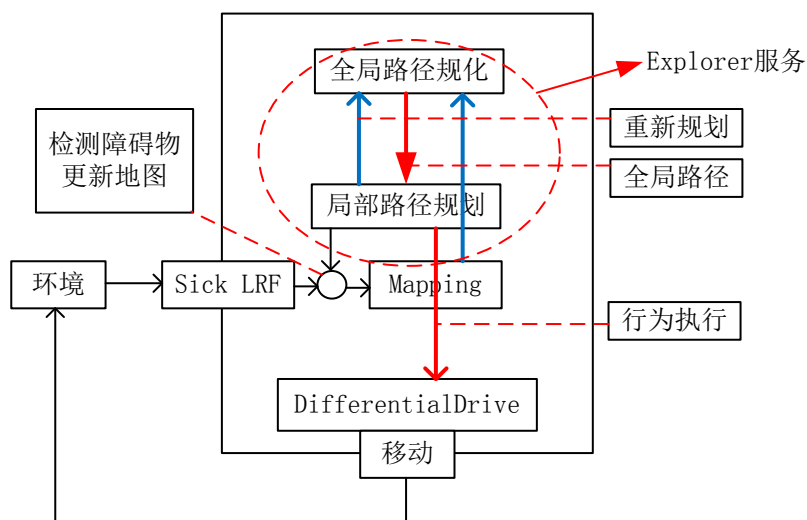
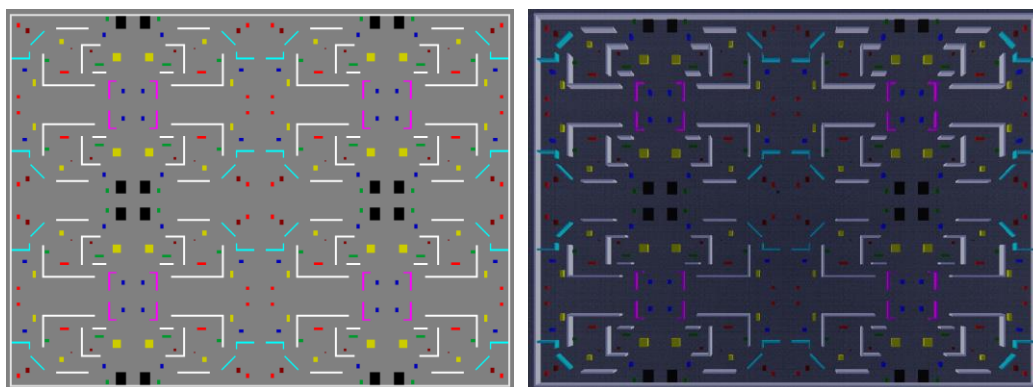


图 5-5 导航系统体系结构

### 5.3.2 地图设计

正如上一节介绍, MRDS 提供了可视化仿真环境工具, 可以用其来设计移动机器人的模拟环境地图。本实验采用了较为巧妙的方式实现了地图的搭建, 即采用位图向环境地图的映射, 采用不同颜色搭建不同高度和样式的障碍物。本文采用手绘的 800 像素\*600 像素的位图, 将不同颜色值进行编码映射为模拟环境中不同的障碍物, 位图中的 1 个像素点对应于模拟环境的 10 cm<sup>2</sup> (映射的比例可以通过配置文件进行调整)。不同的颜色对应仿真环境地图中不同的障碍物高度, 以此来区别障碍物的不同, 设计出较为复杂的环境地图, 实现效果如图 5-6。





a) 绘制的像素位图

b) 映射的环境地图

图 5-6 模拟环境设计

左图为手绘的位图，右图为映射后得出的模拟环境地图。除了模拟的环境地图，还有一个小车构建的内部地图，其是对模拟环境的描述，在移动机器人行进的过程中需要通过激光传感器不断的更新和完善此地图，并根据已获取的地图信息进行全局和局部的路径规划。内部地图的构建比较复杂，采用了窗口界面予以展示。首先创建一个  $800 \times 600$  的地图矩阵对应于窗口图片的像素单元，每个像素点对应于模拟环境地图的  $10 \text{ cm}^2$ 。采用了基于概率的贝叶斯规则，基于新信息的准确性的信任来更新相应矩阵位置障碍物的概率值。如图 5-7，其中黑色代表单元格被占据，白色表示其为自由空间，灰色为默认颜色，表示单元格是否被占用还未知。

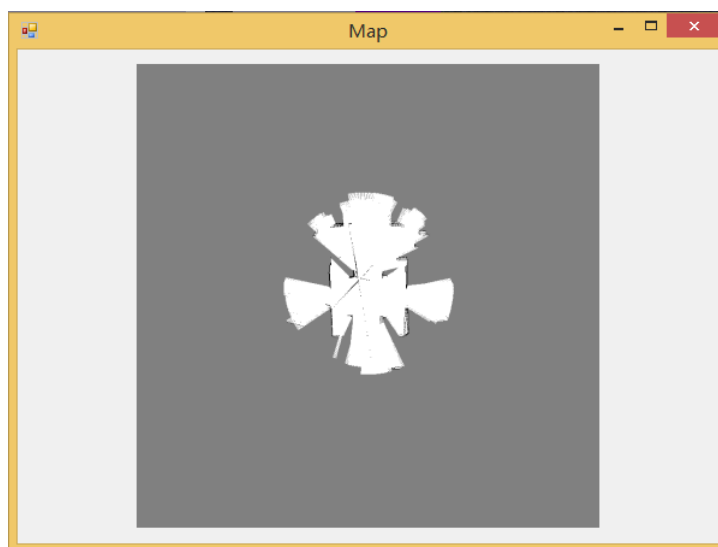


图 5-7 内部维护的位置地图



### 5.3.3 定位问题的处理

前两节讲述了实验机器人的体系结构与环境地图的设计，除此之外，另外还有一些路径规划相关问题需要解决。最重要的就是移动机器人的位置信息如何获取？在第一章有所描述，移动机器人的导航需要建立在准确的定位技术之上，而本文研究的主要对象是移动机器人的路径规划算法，研究之初便对移动机器人做了相应合理的假设以保证路径规划研究的顺利进行。那么本文假设在移动机器人移动的任何时候都知道目前所在位置，为了实现这一目标，需要对移动机器人做一些手脚，即改造差分驱动服务（Differential Drive），为了使移动机器人能够实时获取当前位置，为差分驱动器增加一个所在位置的操作，当移动机器人行走时，差分驱动服务会不断更新自身位置信息到所在位置的属性中，供路径规划使用。其中所在位置的位置属性包括 4 个数据参数，即上一章提到的移动机器人所在的全局栅格节点坐标  $G(x, y)$  以及局部度量坐标  $L(x, y)$ ，如图 5-8 所示。

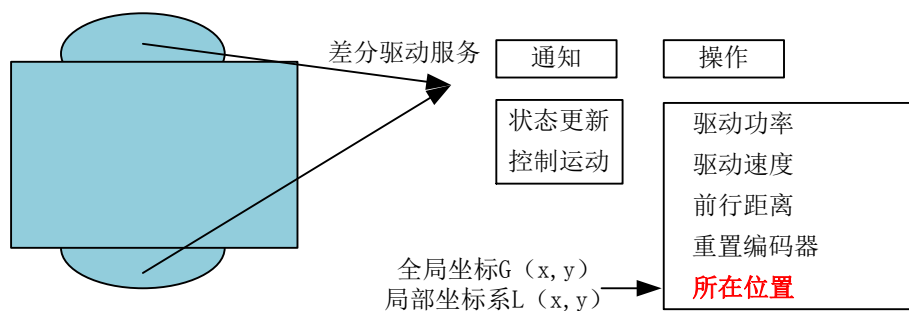


图 5-8 定位处理示意图

## 5.4 实验结果

建模的环境地图为  $8000\text{ cm} \times 6000\text{ cm}$ ，全局坐标对应于  $(-40, -30) \sim (40, 30)$  共  $80 \times 60$  个栅格，一个栅格构成的局部度量地图大小为  $100\text{ cm} \times 100\text{ cm}$ ，坐标为  $(0.0, 0.0) \sim (10.0, 10.0)$ 。移动机器人的大小为  $60\text{ cm} \times 40\text{ cm} \times 50\text{ cm}$  局部步进为  $20\text{ cm}$ ，单位可转向角度为  $180/36=5$  度，速度最高为  $50\text{ cm/s}$ ，激光最远扫描距离为  $800\text{ cm}$ 。路径规划过程为，每次全局路径重规划做一次原地转圈动作，以获取更多信息增加规划的可行度。根据增强学习算法让移动机器人在环境中进行自主移动，自主训练学习。这一部分需要为移动机器人提供一个策略使其不断的有新的路径规划目标并能通过不断试错，强化学习并最终收敛完成任务。因此为其制定了如下策略：

（1）移动机器人具有强制避障模块，即在距离障碍物很近的时候（设定值为  $3\text{ cm}$ ）拒绝继续靠近，以保证训练阶段移动机器人的安全性。此时需要根据具体

情况判断，若是规划路径中的栅格被障碍物填充则认为栅格节点不可达，以当前位置为起点执行全局重规划。若是全局目标点被障碍填充则放弃本次任务，制定新的任务。

(2) 训练时，移动机器人的任务设置以近距离目标抵达为主，这样局部路径规划的成功和失败都很快，能够快速训练神经网络。

移动机器人经过在地图中 100 分钟的训练，移动机器人能够拥有很好收敛效果的神经网络，对于局部路径规划能够很快的完成栅格节点间的抵达任务。如图 5-9 为移动机器人自主训练学习探索地图的过程。

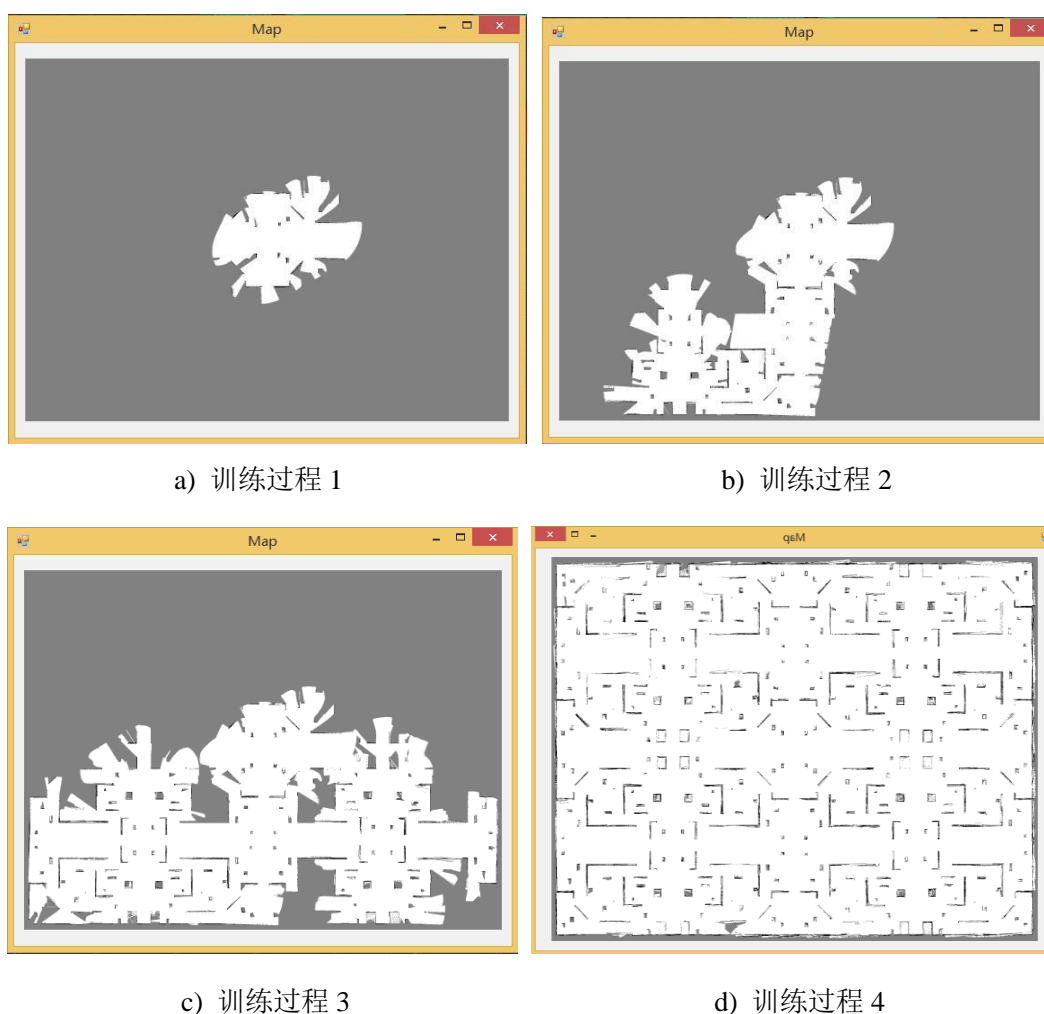


图 5-9 训练过程中位置地图的探索

如图 5-10 为训练好的移动机器人从起始栅格节点  $(-38, 28)$  (对应位置地图的左下角) 到目标栅格节点  $(38, -28)$  (对应位置地图的右上角) 的路径规划过程效果图。图 5-11 展示了移动机器人采用融合算法路径规划的行走轨迹，可以看到规划的路线较为柔和，前行角度灵活多变。为了考量本设计融合路径规划算法的

优缺点，将与 D\* Lite 路径规划的结果进行量化对比。

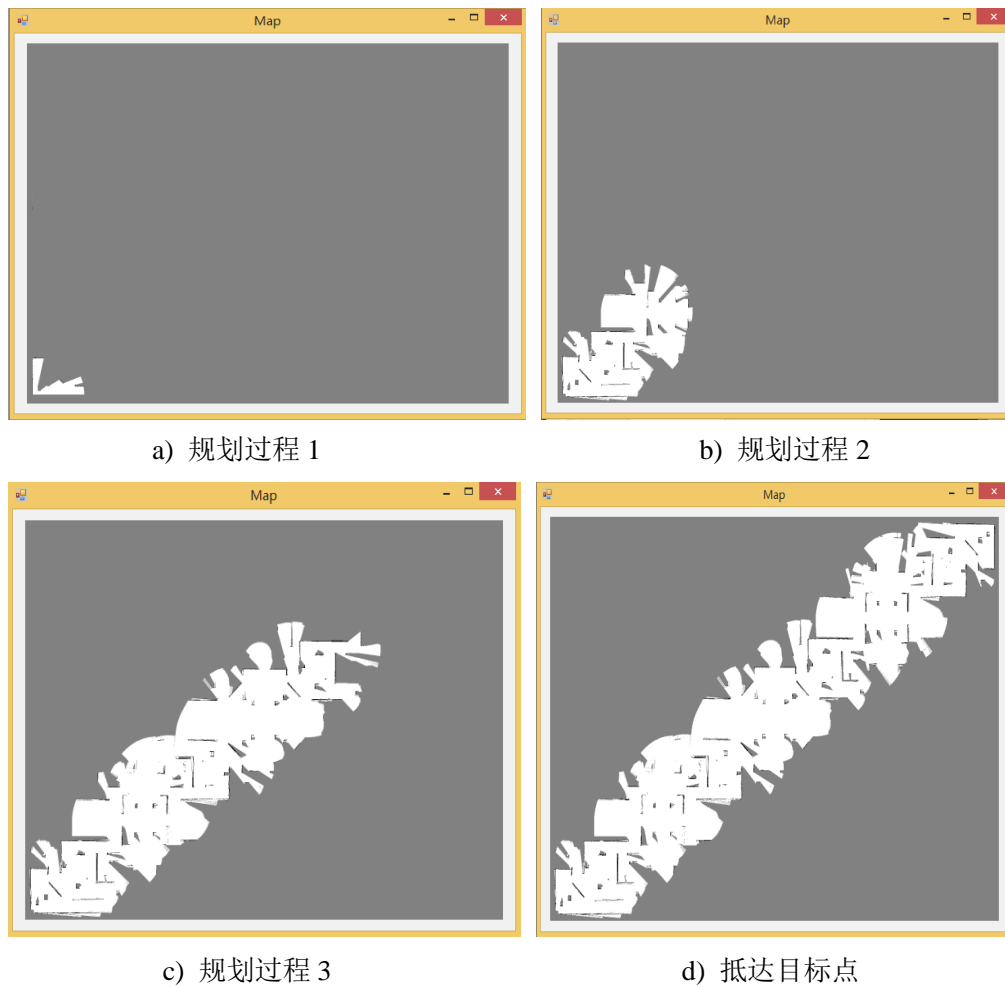


图 5-10 采用融合路径规划算法行走过程

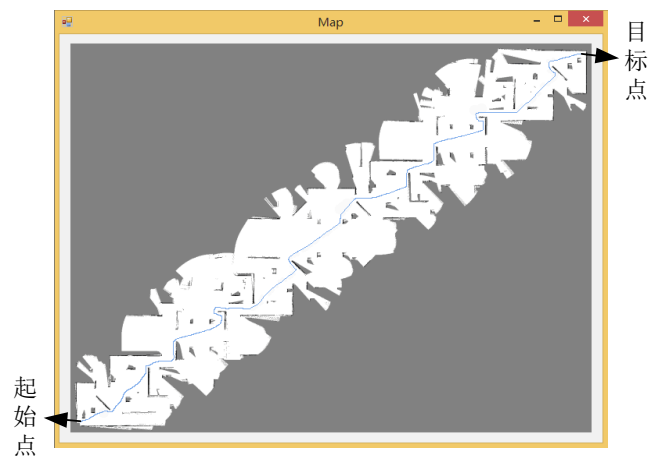


图 5-11 融合算法路径规划行走轨迹

## 5.5 性能比较与量化分析

本节将对融合路径规划算法的结果与 D\* Lite 算法实现的路径规划结果进行比较。并对结果进行分析总结。

对比试验分两组，第一组为 4000 cm\*3000 cm 的较小地图，D\* Lite 算法实现的路径规划分两种节点设置，一种为采用低分辨率 40\*30 栅格地图，每个栅格大小为 100 cm\*100 cm，规划内容为从起始栅格节点 (19, -14) 到目标栅格节点 (-19, 14)。第二种采用高分辨率 400\*300 栅格地图，每个栅格大小为 10 cm\*10 cm，规划内容为从起始栅格节点 (190, -140) 到目标栅格节点 (-190, 140)。对于融合算法则采用前文提到的方法进行测试。测试的内容包括算法规划出的路径长度、累计重规划次数、累计更新节点数以及规划完成花费时间四个维度。为了保证每次路径规划可能由于地图误差带来的结果差异，我们采用多次测试求平均值的方式进行比较，结果见表 5-1。

表 5-1 D\* Lite 与融合算法路径规划实验对比 1

策略	路径长度 (m)	重规划次数	更新节点数	规划花费时间 (min)
低分辨率 D* Lite	空	空	空	$\infty$
高分辨率 D* Lite	57.65	132	12653	5.04
融合算法	59.69	21	1193	1.12
两点距离	47.20	/	/	/

第二组为前文设定的 8000 cm\*6000 cm 的较大地图，类似第一组的设置，对于低分辨率采用 80\*60 栅格地图，规划内容为从起始栅格节点 (38, -28) 到目标栅格节点 (-38, 28)，高分辨率采用 800\*600 栅格地图，规划内容为从起始栅格节点 (380, -280) 到目标栅格节点 (-380, 280)。测试内容以及方法一样。得出的结果见表 5-2。

表 5-2 D\* Lite 与融合算法路径规划实验对比 2

策略	路径长度 (m)	重规划次数	更新节点数	规划花费时间 (min)
低分辨率 D* Lite	空	空	空	$\infty$
高分辨率 D* Lite	115.47	245	59761	24.42
融合算法	118.53	46	4635	2.26
两点距离	94.40	/	/	/

在实验中由于 D\* Lite 算法只能进行特定角度的路径规划（只能是  $\pi/4$  的倍

数), 在环境复杂的地图中使用低分辨率  $D^*$  Lite 算法并不能顺利执行路径规划任务, 其实在做地图栅格化时不能进行更细的确定栅格障碍占用密度, 且为了防止膨胀需要对栅格进行膨胀处理, 导致本可以抵达的路径变得走不通。而高分辨率  $D^*$  Lite 算法的更新节点数远高于融合算法, 这是分辨率较高栅格数指数增加所导致的结果。在路径规划长度方面, 融合算法稍高于高分辨率  $D^*$  Lite 算法, 原因是因为融合算法虽然可以有更多的角度选择, 但受训练时间的约束以及栅格的粗粒度导致未能达到最优, 理论上可以进行参数调整优化神经网络。并且在规划路径长度上面两者相差极小且神经网络的曲线更加柔和比较适合真实环境下的路径规划执行。

两种算法的路径重规划数与更新栅格数的表现与预期一致, 即反应的都是  $D^*$  Lite 算法特点, 本质上是一样的。但由于  $D^*$  Lite 算法路径规划的细粒度栅格地图导致了他重规划数以及维护的栅格数均相关于栅格总数量。

最重要的一项比较就是整个规划所耗费的时间, 这里不包括融合算法中神经网络训练的时间, 以分钟为单位, 在地图面积扩大四倍的情况下, 融合算法的时间增加随距离长度呈线性趋势, 而  $D^*$  Lite 算法则呈指数趋势, 这有着明显的差距, 更显示出了融合算法在大地图中的优势。这主要是因为  $D^*$  Lite 算法的要维护的栅格节点在呈指数增加。这种推论可以在更新节点数上看出来, 但是这只是其中的一个原因, 因为融合算法需要维护的更新节点也是呈指数增加的。这就需要单独考虑同一个地图下两者的差异,  $D^*$  Lite 算法也要比融合算法高数倍, 这是因为除了行走需要耗费时间外, 每次重规划的停顿以及两个规划序列间的停顿都需要花费一定的时间, 而由于融合算法在局部是连续的, 可以减少局部路径规划的停顿时间,  $D^*$  Lite 算法由于全部规划均为离散的, 导致了虽然行走的路程较短但耗费的时间却更多。

总的来讲就是融合算法在全局中采用粗粒度的栅格地图, 使得路径规划更快, 需要更新的栅格更少, 且重规划次数也较少。而局部的路径规划采用增强学习训练的 BP 神经网络进行路径规划, 其可选择的前行方向理论上可以找到更短的全局路径规划。局部地图采用度量地图使得规划的路径连续且更圆滑, 规划的距离增加不会影响局部路径规划的效率, 因此耗费的时间与规划的路径长度呈线性相关特性。正是这种线性相关特性使得融合算法的路径规划更可控。而传统的  $D^*$  Lite 算法虽然相较于  $D^*$  算法减少了路径规划时栅格节点的更新量, 但在大地图复杂环境中, 采用低分辨率栅格地图会导致路径规划失败, 采用高分辨率栅格地图又会增大路径规划的时间复杂度, 导致时间随路径规划长度呈指数增长趋势。因此融合算法相比传统  $D^*$  Lite 算法的优势在于能够快速地完成在大地图复杂环境下的路径规划。

## 5.6 本章小结

本章首先讲解了实现上一节介绍的算法的仿真平台 **MRDS**，介绍了其并行协同处理以及基于 **SOAP** 的 **DSS** 服务设计架构。随后讲解了实验环境的搭建，包括移动机器人的建模、地图的设计以及实时定位问题的巧妙处理。上一章所设计的融合算法被设计成 **MRDS** 平台下的一个服务，并应用到仿真移动机器人模型上。最终完成了本文提出的融合算法在移动机器人上的模拟仿真，并对复杂环境进行了实验测试，对比传统 **D\* Lite** 算法，分析出了其在未知的大地图复杂环境中的优势，验证了其可行性与有效性。

## 结 论

在移动机器人路径规划中, 已知环境地图的路径规划已经取得了丰硕的研究成果, 日渐成熟, 而对于部分未知或者完全未知环境的路径规划还未形成完整的理论体系, 成为了目前主要研究方向之一。本文针对复杂大地图这种特殊的环境下的路径规划做了相关研究, 并提出了一种适应性强、效率高的融合算法路径规划方案。即结合全局路径规划效果极佳的  $D^*$  Lite 算法和使局部路径规划更理性、精细的 BP 神经网络算法而形成的一种融合型算法。综合来说, 本文的主要贡献在于:

(1) 结合路径规划问题阐述了规划问题的规划方法、评价准则以及状态空间表示方式, 并确定了对大地图环境采用分层建模的表示方案。

(2) 对移动机器人常用的路径规划算法进行了理论分析、优缺点总结以及算法间对比。对于比较经典的算法, 本文给出了算法描述、改进策略并通过仿真实验分析出其较为适合的使用场景。为本文研究的特定环境下路径规划的算法选择提供依据, 最终确定采用  $D^*$  Lite 算法做全局路径规划以及神经网络算法做局部路径规划这一方案。

(3) 分析复杂大地图环境下传统路径规划的不足之处, 提出了  $D^*$  Lite 算法与 BP 神经网络融合的解决方案。文中详细描述了地图建模以及融合算法的实现原理。为使移动机器人具备自主探索能力, 神经网络采用增强学习算法进行训练。

(4) 提出一种路径规划算法的验证方案, 即采用微软的 MRDS 平台, 将算法封装成服务应用到移动机器人上进行测试验证。采用量化分析的方式与传统  $D^*$  Lite 算法下路径规划进行对比分析, 得出融合算法的有效性与可行性, 为大地图复杂环境下路径规划提供了新的可行方案。

本课题研究过程中, 默认了准确定位这一前提, 而实际移动机器人并不能提供非常精准定位, 因此要将本文提出的融合算法应用在实际移动机器人上仍有大量的工作需要完成。另外, 如何精准地维护环境地图信息, 如何优化神经网络学习方法使之更快速地收敛都是今后可以深入研究的内容。

## 参考文献

- [1] Hossain M A, Ferdous I. Autonomous Robot Path Planning in Dynamic Environment Using a New Optimization Technique Inspired by Bacterial Foraging Technique[J]. Robotics & Autonomous Systems, 2015, 64(C):137-141.
- [2] Raja P, Pugazhenth S. Path Planning for Mobile Robots in Dynamic Environments Using Particle Swarm Optimization[C]//Artcom 2009, International Conference on Advances in Recent Technologies in Communication and Computing, Kottayam, Kerala, India, 27-28 October. 2009:401-405.
- [3] Garrido S, Moreno L, Abderrahim M, et al. Path Planning for Mobile Robot Navigation Using Voronoi Diagram and Fast Marching[C]//IEEE International Conference on Intelligent Robots and Systems. IEEE, 2006:2376-2381.
- [4] Hossain M A, Ferdous I. Autonomous Robot Path Planning in Dynamic Environment using A New Optimization Technique Inspired by Bacterial Foraging Technique[J]. Robotics & Autonomous Systems, 2015, 64(C):137-141.
- [5] Zhang Y, Gong D W, Zhang J H. Robot Path Planning in Uncertain Environment Using Multi-objective Particle Swarm Optimization[J]. Neurocomputing, 2013, 103(2):172-185.
- [6] Raja P, Pugazhenth S, Raja P, et al. Path planning for A Mobile Robot in Dynamic Environments[J]. International Journal of Physical Sciences, 2011(20):4721-4731.
- [7] 廖绍辉, 张连东. 机器人路径规划技术的现状与发展趋势[J]. 机械工程师, 2007(7):13-16.
- [8] 李群明, 熊蓉, 褚健. 室内自主移动机器人定位方法研究综述[J]. 机器人, 2003, 25(6):560-567.
- [9] Xie S R, Luo J, Rao J J. Computer Vision-based Navigation and Predefined Track Following Control of A Small Robotic Airship[J]. 自动化学报, 2007, 33(3):286-291.
- [10] 贺伟, 梁昔明. 未知环境中移动机器人 SLAM 问题的研究进展[J]. 微计算机信息, 2005(3):179-180.
- [11] Stentz A. Optimal and Efficient Path Planning for Partially-known Environments[C]//IEEE International Conference on Robotics and Automation, 1994. Proceedings. 1994:3310 - 3317.
- [12] Stentz A. The Focussed D\* Algorithm for Real-Time Replanning[C]// International Joint Conference on Artificial Intelligence. 2000:3310-3317.
- [13] Koenig S, Likhachev M. D\*lite[C]// Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial



- Intelligence, July 28 - August 1, 2002, Edmonton, Alberta, Canada. 2002:476-483.
- [14] Ferguson D, Stentz A. Field D\*: An Interpolation-Based Path Planner and Replanner[C]// Robotics Research: Results of the, International Symposium, Isrr 2005, October 12-15, 2005, San Francisco, Ca, USA. 2005:239-253.
- [15] Tang L, Dian S, Gu G, et al. A Novel Potential Field Method for Obstacle Avoidance and Path Planning of Mobile Robot[C]//IEEE International Conference on Computer Science and Information Technology. 2010:633-637.
- [16] 刘义, 张宇. 基于改进人工势场法的移动机器人局部路径规划的研究[J]. 现代机械, 2006(6):48-49.
- [17] 张建英, 赵志萍, 刘瞰. 基于人工势场法的机器人路径规划[J]. 哈尔滨工业大学学报, 2006, 38(8):1306-1309.
- [18] 禹建丽, V.K roumov, 孙增圻,等. 一种快速神经网络路径规划算法[J]. 机器人, 2001, 23(3):201-205.
- [19] Guan-Zheng, Huan, SLOMAN,等. Ant Colony System Algorithm for Real-Time Globally Optimal Path Planning of Mobile Robots[J]. 自动化学报, 2007, 33(3):279-285.
- [20] 胡选子, 谢存禧. 基于人工免疫网络的机器人局部路径规划[J]. 山东大学学报(理学版), 2010, 45(7):122-126.
- [21] Qu H, Xing K, Alexander T. An Improved Genetic Algorithm with Co-evolutionary Strategy for Global Path Planning of Multiple Mobile Robots [J]. Neurocomputing, 2013, 120(10):509-517.
- [22] 秦永法, 刘红军, 赵明扬,等. 基于模拟退火算法的移动机器人路径规划[J]. 信息与控制, 2002, 31(5607):768-768.
- [23] 王新杰, 武秋俊, 王建军,等. 基于改进遗传算法的移动机器人路径规划[J]. 煤矿机械, 2008, 29(4):28-30.
- [24] 郑秀敏, 顾大鹏, 刘相术. 基于栅格法-模拟退火法的机器人路径规划[J]. 微计算机信息, 2007, 23(5):247-248.
- [25] 黄席樾, 蒋卓强. 基于遗传模拟退火算法的静态路径规划研究[J]. 重庆工学院学报:自然科学版, 2007, 21(11):53-57.
- [26] František Duchoň, Andrej Babinec, Martin Kajan, et al. Path Planning with Modified a Star Algorithm for a Mobile Robot [J]. Procedia Engineering, 2014, 96(96):59-69.
- [27] 赵建伟, 王洪燕, 唐兵,等. 基于移动机器人的 GPS 轨迹生成及定位研究[J]. 工矿自动化, 2016, 42(1):17-19.
- [28] 唐世, 叶永. 基于 UWB 无线定位技术的空管设备备件管理系统[J]. 空中交通, 2015(9):81-82.

- [29] 董艳侠, 张红英, 陈姣. 基于 Wi-Fi 室内定位的位置指纹算法[J]. 工业控制计算机, 2015(1):72-74.
- [30] 贺伟, 梁昔明. 未知环境中移动机器人 SLAM 问题的研究进展[J]. 微计算机信息, 2005(3):179-180.
- [31] Mcfetridge L, Ibrahim M Y. A New Methodology of Mobile Robot Navigation: The Agoraphilic Algorithm[J]. Robotics and Computer-Integrated Manufacturing, 2009, 25(3):545-551.
- [32] 董从建, 王可. 基于卡尔曼滤波定位跟踪的现状与发展[J]. 科技研究, 2014.
- [33] Garcia M A P, Montiel O, Castillo O, et al. Path Planning for Autonomous Mobile Robot Navigation with Ant Colony Optimization and Fuzzy Cost Function Evaluation[J]. Applied Soft Computing, 2009, 9(3):1102-1110.
- [34] 许斯军, 曹奇英. 基于可视图的移动机器人路径规划[J]. 计算机应用与软件, 2011, 28(3):220-222.
- [35] 吴峰光, 奚宏生. 一种新的基于切线的路径规划方法[J]. 机器人, 2004, 26(3):193-197.
- [36] 许松清, 吴海彬, 林宜,等. 基于 Voronoi 图法的移动机器人路径规划[J]. 中国工程机械学报, 2005, 3(3):336-340.
- [37] 刘康, 段滢滢, 张恒才. 基于路网拓扑层次性表达的驾车路径规划方法[J]. 地球信息科学学报, 2015, 17(9):1039-1046.
- [38] 于红斌, 李孝安. 基于栅格法的机器人快速路径规划[J]. 微电子学与计算机, 2005, 22(6):98-100.
- [39] 杨宪泽. 基于图搜索算法的探讨[J]. 西南民族大学学报(自然科学版), 1998(2):117-122.
- [40] Rampasek L, Goldenberg A. TensorFlow: Biology's Gateway to Deep Learning?[J]. Cell Systems, 2016, 2(1):12-14.
- [41] Yin J B. Robot Programming and Simulation Based on MSRDS Autonomous[J]. Science & Technology Information, 2013:60-58.
- [42] Weimer S. Michael E. Bratman, Shared Agency: A Planning Theory of Acting Together[J]. The Journal of Value Inquiry, 2016, 18(2):1101-1104.
- [43] Reza M, Satapathy S K, Pattnaik S, et al. Optimized Point Robot Path Planning in Cluttered Environment Using GA[M]// Proceedings of Fifth International Conference on Soft Computing for Problem Solving. Springer Singapore, 2016:475-485.
- [44] Liu Y, Zhang W G, Guang-Wen L I. Study on Path Planning Pased on Improved PRM method[J]. Application Research of Computers, 2012, 29(1):104-106.
- [45] Melchior N A, Simmons R. Particle RRT for Path Planning with Uncertainty[J]. 2007:1617-1624.

## 攻读硕士期间发表论文及成果

### 一、申请已获得的专利

无

### 二、申请已获得的专利

- [1] 韩啸, 张钦宇, 徐开放, 等. 一种快速三维动态手势识别系统及手势数据采集设备: 中国, 201620310639.4[P]. 2016-08-01.
- [2] 张钦宇, 赵国钦, 徐开放, 等. 一种基于蓝牙通信的路面颠簸情况识别装置: 中国, 201620073367.0[P]. 2016-01-26.

## 哈尔滨工业大学学位论文原创性声明和使用权限

### 学位论文原创性声明

本人郑重声明：此处所提交的学位论文《基于 D\* Lite 算法的移动机器人路径规划研究》是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果，且学位论文中除已标注引用文献的部分外不包含他人完成或已发表的研究成果。对本学位论文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。

作者签名： 日期：2017 年 1 月 4 日

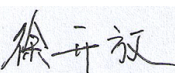
### 学位论文使用权限

学位论文是研究生在哈尔滨工业大学攻读学位期间完成的成果，知识产权归属哈尔滨工业大学。学位论文的使用权限如下：

(1) 学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文，并向国家图书馆报送学位论文；(2) 学校可以将学位论文部分或全部内容编入有关数据库进行检索和提供相应阅览服务；(3) 研究生毕业后发表与此学位论文研究成果相关的学术论文和其他成果时，应征得导师同意，且第一署名单位为哈尔滨工业大学。

保密论文在保密期内遵守有关保密规定，解密后适用于此使用权限规定。

本人知悉学位论文的使用权限，并将遵守有关规定。

作者签名： 日期：2017 年 1 月 4 日

导师签名： 日期：2017 年 1 月 4 日

## 致 谢

写着写着就来到了末尾，二年半的研究生学习生涯在这里就要画上句号了，心中的不舍已经使我打字的手开始颤抖。时间虽短，但成长很多，这两年半里承载了在实验室艰辛奋战的日日夜夜，承载与老师们与战友们的深切情谊。在此，我有很多人需要感谢。

首先衷心感谢导师张乃通老师对我的悉心指导，他在学术上的造诣令人望洋兴叹，他在课题上给了我很大的帮助，在思想上更是深深的影响了我，让我明白学术科研的真谛。他的言传身教必将使我终生受益。同时，我还要感谢张钦宇老师，生活中无比亲切可爱，科研中却又无比专注严谨，他为我们提供了一个很好的实验室平台，让我们更好完成科研工作。另外还要感谢林威老师，他的视野总能望到未来，他教会了我们如何把握趋势，如何快速学习。

其次，我要感谢实验室的师兄们，韩啸师兄总能在我遇到问题的时候为我开拓新的思路，生活上也给了我很大的帮助，薛瑞范师兄、刘昆师兄、陈少勇师兄等在课题和学习上也给了我很多帮助，师兄们毫无保留的指导让我很快进入课题，并在课题上不断前进。

最后，我还要感谢我的项目组战友刘易、杨智翔、谢国超、刘元震以及赵国钦同学，两年半以来我们一起讨论、搞科研、吃饭、熬夜、回宿舍，我们亲如兄弟姐妹，相互勉励支持。还有包容我的两位舍友李海波和张朝阳同学，生活中我们相互帮助，使得寝室总是那么温馨舒适。还有一起玩耍的王苗苗、李齐齐、丁宜、黄华等同学，我们的友谊深切永恒。还有我那最可爱的师弟师妹们，吴夕、李荣华、肖乔、吴柏倩等总能在需要的时候很认真的做我的得力助手。这段时光虽短，但在我记忆中永恒，我永远不会忘记你们。

在我的学习生涯中还有默默支持我的亲人，是他们的良好培养让我拥有正确的世界观价值观，让我相信真理相信未来。

在此向培育我的母校、老师和同学们致以最真挚的感谢！