分类号 _____ 学 号 1008240871

西安建筑科技大学

学 位 论 文

基于嵌入式系统的
迷宫机器人路径规划算法研究

作者 刘世泽

指导教师姓名 段中兴 教授

申请毕业级别 硕士 专 业 名 称 控制理论与控制工程

论文提交日期 2013.05 论文答辩日期 2013.05

毕业授予单位 西安建筑科技大学

答辩委员会主席

评 阅 人

声　　明

本人郑重声明我所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中已经标明引用的内容外，本论文不包含其他个人或集体已经发表或撰写过的研究成果，也不包含本人或其他人在其它单位已申请学位或为其它用途使用过的成果。与我一同工作的同志对本研究所做的所有贡献均已在论文中作了明确的说明并表示了致谢。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

论文作者签名：刘世峰　　日期：2013.5.27

关于学位论文使用授权的说明

本人完全了解西安建筑科技大学有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交学位论文的复印件和电子版，允许论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以采用影印、缩印或者其它复制手段保存学位论文。

（保密的论文在论文解密后应遵守此规定）

论文作者签名：刘世峰　　指导教师签名：郭中兴　　日期：2013.5.27

.....

本人授权中国学术期刊（光盘版）杂志社、中国科学技术信息研究所等单位将本学位论文收录到有关“学位论文数据库”之中，并通过网络向社会公众提供信息服务。同意论文提交后滞后：□半年；□一年；□二年发布。

论文作者签名：刘世峰　　指导教师签名：郭中兴　　日期：2013.5.27

注：请将此页附在论文首页。

基于嵌入式系统的迷宫机器人路径规划算法研究

专　　业：控制理论与控制工程
硕 士 生：刘世泽
指导教师：段中兴教授

摘要

随着科技技术的进步，从无人驾驶飞机到生产线上的机械手臂，从教学机器人到娱乐机器人, 智能移动机器人在生产生活得到了越来越广泛的应用。其中迷宫机器人是目前研究的一个热点。

本论文以在迷宫中可以自主进行路径规划的机器人为研究对象，主要对其算法进行了深入的研究，通过对比深度优先搜索算法、广度优先搜索算法、Flood及其改进算法进行优化选取，以及记忆功能实现和优化，和运动控制等方面提出改进措施。同时对它的控制系统，微控制器系统，电源电路各个模块，车体结构等方面进行设计研究。

首先，通过阅读大量参考文献与实验，完成了车体结构的总体设计。经过调试，完成了最小系统，各电源电路模块的设计和零部件的选取。绘制了系统原理图, 完成了硬件的设计。

其次，主要对一些经典的算法和改进算法应用于迷宫问题的求解，通过对比给出了深度优先搜索算法、广度优先搜索算法、Flood 及其改进算法的优缺点，最终确定本论文将采用洪水算法结合向心法则来对迷宫进行探索，在此算法基础上通过优化后记忆功能使得迷宫机器人的探索更为有效和快速，利用实验证明其在求解迷宫问题上的可行性。

最后，针对迷宫机器人在实际行走迷宫过程中行走姿态和电机进行控制，对转弯和直线加速进行优化，确保其顺利高效地完成迷宫搜索任务。

论文给出了机器人走迷宫的详细硬件设计和软件实现过程，实现了迷宫机器人在无人干预的情况下自主完成迷宫探索任务。

关键词：迷宫机器人；路径规划；洪水算法；向心法则；迷宫记忆

Research of Maze Robot Path Planning Algorithm Based on the Embedded System

Specialty: Control Theory and Control Engineering
Name: Liu shize
Instructor: Prof.Duan zhongxin

Abstract

With the advancement of science and technology, automobile Roberts have been widely applied in people's lives and the process of production, for instance, from the mechanical arms in production lines to the unmanned aircrafts, from the teaching Roberts to the entertainment Roberts. Particularly, maze Roberts are one of the hot spot in current research.

The main project of this paper is that whether the Roberts can plan their own path initiatively in mazes. The writer had a deep research in its algorithm through depth-first algorithm, the broad-first algorithm, Flood and other processed algorithm to carry out improved measures in order to realize the optimization of memorial functions and movement control of Roberts in mazes. While, the author also studied the controlling system, micro-controlling system, power supply and circuit, body structure and so on.

First of all, the author accomplished the over-all design of body structure by reading amounts of references and experiments. After debugging, the author decided the machine replacement parts and finished the design of minimum system，each mudole of power circuit. At the same time, the author drawn the system principle diagram.while finished the design of hardware by welding.

Secondly, answering some classic algorithms and improved algorithms in the problems of mazes, the writer compared the advantages and disadvantages of depth-first algorithm, the broad-first algorithm, Flood and other processed algorithm to assure that this paper would apply the method of Flooding algorithm combined to Center algorithm to study maze. On the basis of the method, this paper optimized the

memorial function to make the maze Roberts more effectively and quickly and testified the possibility to solve the difficulties of mazes through experiments.

Finally, the author focused on the control of movement of Roberts in actual walking and electric motor，optimized the acceleration in turns and straight to assure the Roberts could accomplish the searching tasks in mazes in high efficiency.

This paper tells the process of detailed hardware and software design to make the Roberts across mazes. Successfully, the writer realized that the Roberts can finish the searching tasks in mazes without manned intervention.

Keywords: Maze robot; Path planning; Flooding algorithm; Center algorithm; Maze memory

目　录

1 绪论

1.1 研究背景

1.1.1 智能移动机器人的发展概述

1920 年，捷克作家卡雷尔. 查培克(Karel Capek)发表了科幻剧本《罗沙姆的万能机器人》，卡雷尔. 查培克把剧中的人造机器人起名为捷克语″Robota″(意思是苦力，劳仆)，后来演变成英文的“Robot″，这被当作了“机器人”(Robot)一词的起源。20 世纪 60 年代，人们已经开始对移动机器人进行了研究。最早的自主移动机器人 Shakey 是由斯坦福研究院的 Nils Nilssen 和 Charles Rose 等人研制并命名的[1]。为的是实现在复杂的环境下对移动机器人进行实时控制，所研究的内容涉及到机器人导航与路径规划，目标识别与处理，多传感器信息融合和处理等诸多移动机器人的关键技术，在这一阶段是移动机器人走向产业化、实用化的开始阶段。到了 20 世纪 70 年代末，随着计算机技术、传感器技术的快速发展，移动机器人的研究达到了一个新的高潮。1980 年，工业机器人在日本大为普及，日本也因此赢得了“机器人王国”的美誉，这一年被称为“机器人元年”。到了 80 年代中期，为了满足整个汽车行业的蓬勃发展，设计和制造机器人的浪潮风靡全世界，此时，机器人所面对的环境也演变成为了真实的世界，各国军方及一大批世界著名的公司和大学的加入也使得移动机器人的研制更加迅猛地发展起来[2-4]，从而促进了智能移动机器人的发展。20 世纪 90 年代以来，计算机技术、微电子技术等得到了高速的发展，相应智能移动机器人技术也得到了快速的发展，装配智能移动机器人和柔性装配技术得到了大量地应用，并且进入了快速发展的时期，以研制高标准的传感器，高适应性的移动机器人控制技术，高真实环境下的路径规划技术等为标志，逐步开展了更高水平智能移动机器人的研究。

发展了近百年的机器人，大致可以分为三个成长阶段，也就是三个时代。第一个时代是可编程机器人。这个时代的机器人可以根据操作员所编写的程序，完成一些程序要求的操作。这一时期机器人从 20 世纪 60 年代后半期开始投入到实际使用当中，目前在工业领域得到了广泛的应用。第二个时代是感知机器人，即自适应机器人，它是在这一时期机器人基础上逐渐发展演变而来的，具有一定的“感知”能力。这类机器人现在在工业领域已有应用。第三个时代机器人将具有识别、推理、学习等智能机制，它可以把感知和行动智能地结合在一起，因而可

以在不同的环境下工作，称之为智能移动机器人[5-7]。目前，这类机器人处于试验阶段，将向实用化方向发展。

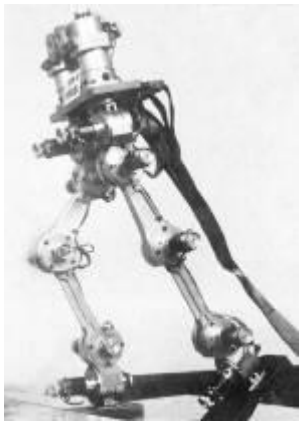


图 1.1 第一代机器人

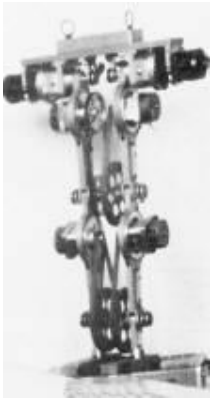


图 1.2 第二代机器人

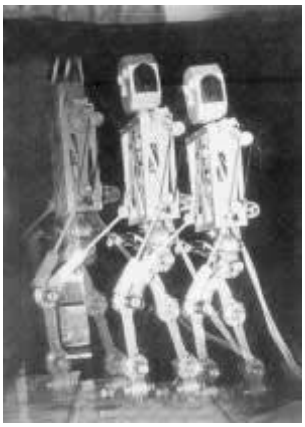


图 1.3 第三代机器人

1.1.2 智能移动机器人国内外现状

智能移动机器人是第三代的机器人[8-11]，该代机器人采用多种类型的传感器，利用多信息融合技术将多种传感器的信息进行融合，具有很强的学习机适应能力，在各种复杂环境下能够快速、有效的完成任务。

现阶段研制的智能移动机器人其智能水平并不高，只可以说是智能移动机器人的初级阶段。智能移动机器人研制当中的两个核心问题：一是如何提高智能移动机器人的自主性，即就是希望智能移动机器人能更多的借助于自己独立于人去完成任务，具有更为人性化的人机交互界面。未来的设想是操作人员只要给出智能移动机器人要完成的任务，其就能自助判断完成该任务的步骤和方法，并自主地去完成它。二是提高智能移动机器人的适应性，提高智能移动机器人适应不同环境的能力，这是相对于智能移动机器人与环境的关系而言，希望加强它们之间的交互关系[12]。

纵观现阶段国际上的智能移动机器人发展，美国的智能移动机器人技术一直处于领先地位，其技术全面，适应性也很强，性能可靠、功能全面、精确度高，其中视觉、触觉等人工智能技术已在汽车工业和航天领域中得到广泛应用。近年来由于一系列日本政府的扶植政策，日本的各类智能移动机器人也得到了飞速的发展。欧洲各国在智能移动机器人的研究和应用方面在世界上也处于公认的领先地位[13]。

早在“七五”计划的国家重点科研规划内容里机器人的研究已被列入其中。与此同时，沈阳建立起第一个全面开展机器人基本理论与基本元件研究的示范工

程。经过十几年的发展，我国已经掌握了相关技术并能熟练制造出用于搬运、弧焊、点焊、喷漆、装配等各类机器人。现如今，各类机器人已经投入到相应的领域当中，并获得了很好地收益与效果。其中在 20 余家企业中有 130 多套喷漆机器人在多条自动喷漆生产线得到广泛应用，并且弧焊机器人也已应用在多个汽车制造厂的焊装线。在我国长春第一汽车厂机东方汽车厂，我国自然研制生产的喷漆机器人以投入运行[13]。1986 年 3 月，我国已将研究、开发智能移动机器人的内容列入国家 863 高科技发展规划。我国的智能移动机器人和特种机器人在国家 863 计划的支持下已经取得了显著的成果。其中 6000 米水下无缆机器人的成果居世界领先水平，该机器人在 1995 年深海试验获得成功，使中国能够对大洋海底进行精确、高效、全覆盖的观察、测量、储存和进行实时传输，并能精确绘制深海矿区的二维、三维海底地形地貌图，从而推动了中国海洋科技的发展[13]。

自 1988 年开始,清华大学计算机系智能技术与系统国家重点实验室在在国家科工委和国家 863 计划的资助下，开始研制迷宫机器人系统。其研制的 THMR (Tsinghua Mobile Robot)系列取得了预期的效果。实验室在迷宫机器人的自主式系统、传感器信息的获取与处理、路径规划、感知动作、通讯及临场感应技术进行了深入的研究。其中 THMR-III系列在一般道路上的速度为 5km/h, 在自主道路跟踪时可达到 5-10km/h。除此之外，实验室又研制出了新一代的自主移动机器人 THMR-V，该自主机器人在一般道路设计的车速为 20km/h，在高速公路为 80km/h。

我国智能移动机器人的相关研究较国外起步晚，相对于美国，日本，新加坡等国在技术全面性和精度上还有不小的差距，我国还需要奋起直追。

1.2 研究意义

随着智能移动机器人的发展，其渐渐被应用到工业，国防以及人们的日常生活等各个方面。伴随着智能移动机器人的广泛应用，也产生了巨大的经济效益和社会效益，推动了相关学科和技术的发展，这使得智能移动机器人研究成为了一个极为活跃的研究领域。目前智能移动机器人主要应用于两个场合：其一是对人来说非常危险的场合，如核工业机器人，主要用来处理核工厂的一些放射性原料。军用机器人的特点是在完全自主的情况下完成地形识别和选择路线，并且具有对目标的自动搜索、锁定和摧毁的侦查能力与作战能力。如法国的自主式快速运动侦察车（DAIMS），美国的 Navlab 自主导航车、SSV 半自主地面战车，德国的 MV4 爆炸物处理机器人等;另一个是日常的生产生活中无法使用人类劳动力的场

合, 如水下机器人（美国的 AUSS、俄罗斯的 MT-88、法国的 EPAVLAI 等）：主要用来进行海洋石油钻探作业、管道铺设以及电缆检测维护等。空间机器人（美国的“勇气号”和“机遇号”），主要用来对外太空进行探索。

迷宫机器人是由微处理器、红外传感器、电机等多个零部件组成的智能行走设备，迷宫机器人可以完成自主搜索、自主记忆、自主选择路径的任务，最终到达指定的目的地[14]。

迷宫机器人所需要掌握的知识技能比其它技术平台更宽更广，其结合了光电、机械、控制、程序设计及人工智能等多方面的学科知识。迷宫机器人技术具有先进性和更新性,而且智能移动机器人总是使用当今诸多领域的前沿先进技术。不仅如此，智能移动机器人还自主吸收各个技术领域的最新成果，从而保证了迷宫机器人的技术走在时代的前列。迷宫机器人的实用性和其对科技的推动性，使得现如今迷宫机器人和其他智能移动机器人已经应用到人们生产和生活的各个方面，并推动了其他领域的发展。

1.3 迷宫机器人主要研究内容与研究难点

迷宫机器人是一个具备自主运动能力的智能移动机器人,它能在由 16 方块×16 方块组成的迷宫中，从起点开始，自主探索迷宫，直至找到终点，再由探索过迷宫的信息来自主判断得到最短路径。最后迷宫机器人以最佳路径返回起点后全力冲刺到终点。

迷宫机器人的硬件结构设计。包括迷宫机器人的车体结构，微控制器的选型，电源电路，红外线传感器系统，车体零部件等方面进行设计研究。

迷宫机器人的软件设计。包括初始化各个模块，记忆功能的实现与优化，路径规划算法，运动姿势避障控制等。

首先是软件壁垒，现如今的智能移动机器人中的算法都仅仅能够适用在被规划设计好的环境当中，对于不同的环境和任务，同一算法的智能移动机器人几乎是寸步难行。因此，不断提高智能移动机器人适应性的软件控制显得尤为重要。其次，迷宫机器人的研究涉及到机械、电机、控制、软件设计等诸多学科，彼此互相联系，互相制约。比如，迷宫机器人的速度控制由软件设定，但其走直线和转弯的速度快慢，有涉及到运动学原理的知识，迷宫机器人的硬件本身也是一个制约因素。

1.4 小结

在本章中，先对智能移动机器人的研究背景和意义做了简要介绍。接着叙述了本课题的主要研究内容和研究难点。

2 迷宫机器人控制系统与车体结构设计

2.1 迷宫机器人控制系统

迷宫机器人硬件由控制系统和车体两部分组成。控制系统由微处理器、传感器、电机等组成，车体结构由车轮、车架、电池等组成。微控制器是 Luminary 公司生产的 Cortex-M3 内核的 ARM 处理器 LM3S615。共有 5 组一体式红外线传感器，型号为 IRM8601S，每组红外线传感器由红外线发射和红外线接收组成，用來检测迷宫机器人与墙壁的距离和提供迷宫墙壁信息，校正运动时位置与方向的误差。电机选取为步进电机，电池为可充电的锂电池规格是电流 2200mAh，电压 7. 4V。数码管上可以实时显示电量，避免了电量不足所带来的麻烦。在电路板上有一个开始按键和一个复位按键，满足了运行时的实际需求。还预留了 6 个 GPIO 口，一个串口，一个 SPI 接口，以用于扩展。

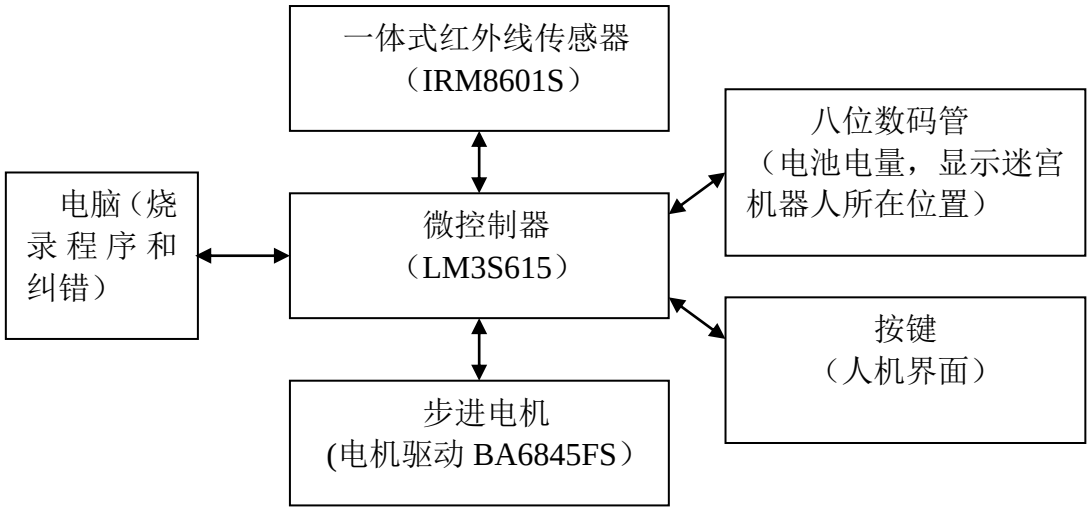


图 2.1 迷宫机器人硬件框架图

2.2 系统硬件选型及电路设计

2.2.1 微控制器及最小系统设计

LM3S615 系列微控制器是首款基于 Cortex-M3 内核的 32 位 ARM 控制器，

它提供了 32 位处理器的性能，价格却与之前的 8 位和 16 位微处理器基本相同，同时所有器件都以小型封装的形式提供。这款将 32 微处理器引入到嵌入式微处理器的芯片具有众多优点，如 LM3S615 微控制器可以兼容 Thumb 的 Thumb-2 指令集来降低内存的需求量，进而降低成本[15]。

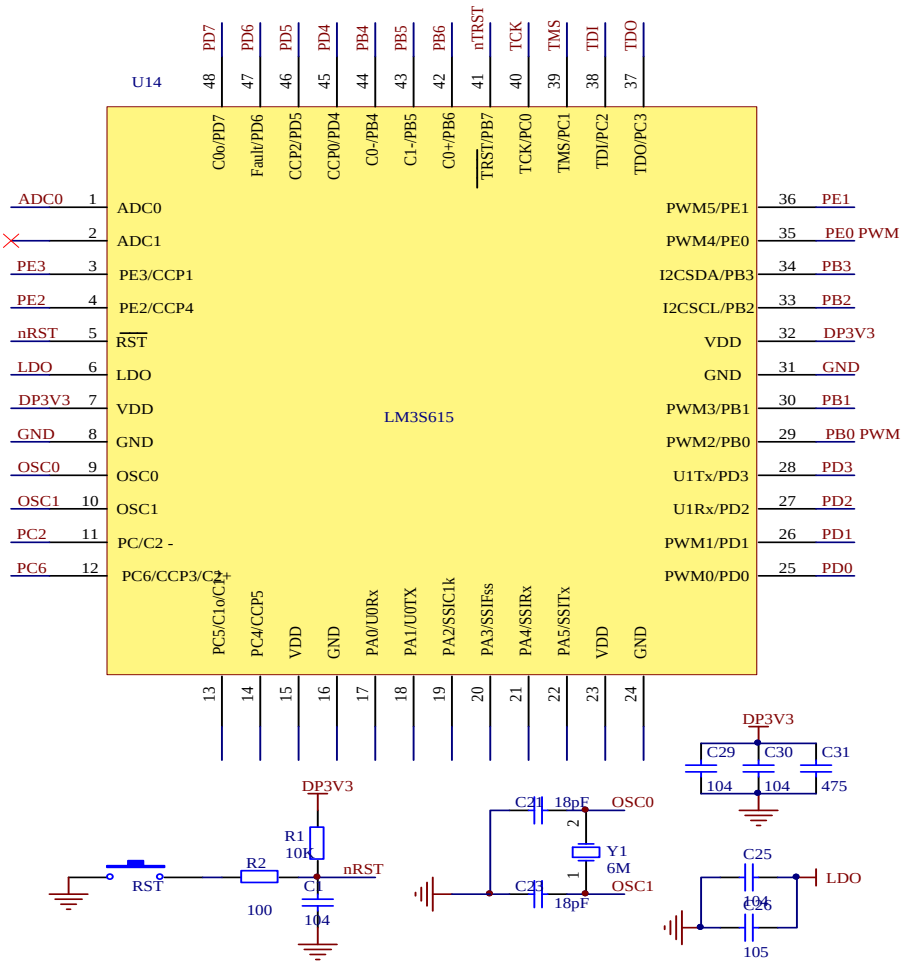


图 2.2 LM3S615 最小系统图

2.2.2 系统供电与电源电路

迷宫机器人采用外接的电池供电，并且由于要求不同要给各个模块提供不同的电压，分别为驱动电机提供 7.4V 电压，传感器提供 5V 电压供电，微控制器系统提供 3.3V 电压供电。

迷宫机器人的微控制器需要 3. 3V 供电，电路图如图 2.3 所示，外接电源连接 CON2 接头，经过 C36、C2 滤波，然后通过 SPX1117 将电压稳压至 3. 3V，其中二极管 D1 是为了防止电源正负极接反。SPX1117 为一个低功耗正向电压调节器，其可以用在一些高效率，小封装的低功耗设计中。所以说 SPX1117 电压调节器非常适合便携式电脑及电池供电的应用。SPX1117 有很低的静态电流，在满负载时

其低压差仅为 1.1V。当输出电流减少时，静态电流随负载变化，并提高效率。SPX1117 可调节，可以选择 1.5V，1.8V，2.5V，2.85V，3.0V，3.3V 及 5V 的输出电压。输出电流最大可以达到 800mA。电路图中 C3、C4 连接其输出端用来提升瞬态响应和稳定性[16]。POWER 为电源指示灯，系统上电后 POWER 指示灯应当被点亮。电阻 R9 和 R15 用来形成一个分压电路，网络标号 ADC0 与迷宫机器人的 ADC0 端口相连接，其用来检测电池电压。

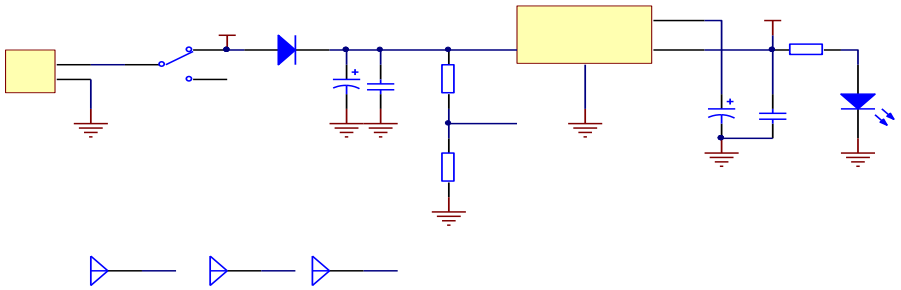


图 2.3 电源电路

2.2.3 电机驱动模块

电机的选择对迷宫机器人的机械结构起着至关重要的作用，它不仅直接影响迷宫机器人的体积和结构布局，而且对其的运动的效率和灵活性也起到了决定性的作用。电机在直流电机和步进电机之间考虑，它们的差异为：直流电机的优点是其在短时间内能够加速到一定的速度，与此同时需要加入测速传感器(如:光电编码器)或是陀螺仪等设备来保证在速度较高的同时还要保持其精准度。步进电机的缺点是跟直流电机同样的规格下，其体积会增大不少，重量也会增加许多。其优点是步进电机无需测速器件和减速器，减少了其硬件和编程的复杂性，容易操作控制。

综合迷宫机器人的需要，最终驱动电机采用步进电机，具体来说，它有下列优点：

（1）步距值不受各种干扰因素的影响。如电压的高低，电流的大小、温度的变化等。

（2）误差不长期积累。虽然迷宫机器人每走一步步进电机所转过的实际角度与理论角度之间总有一定的误差，从某一步到任何一步，也总有一定的累积误差，但是，每转一圈的累积误差为零，所以步进电机的累积误差不会长期的累积下去。

（3）控制性能好，启动、停车、90° 转弯、180° 转身都是在为数不多的脉冲内完成，并且在一定的频率范围之间运行时，一般不会发生失步的情况。所以，

步进电机已经被广泛应用于移动机器人当中。

迷宫机器人的电机由两个永磁式步进电机组成，由于电池的电压和电机驱动芯片要求电压相同，所以直接把输出电压传输到电机的驱动芯片上，两个永磁式步进电机分别控制左右轮。如图 2.4 所示，采用型号为 BA6845FS 的步进电机驱动芯片，每个芯片由两个 H 桥组成，驱动电流上限为 1A，且有三种输出模式，分别为正向、反向和停止，满足了迷宫机器人的实际需求。

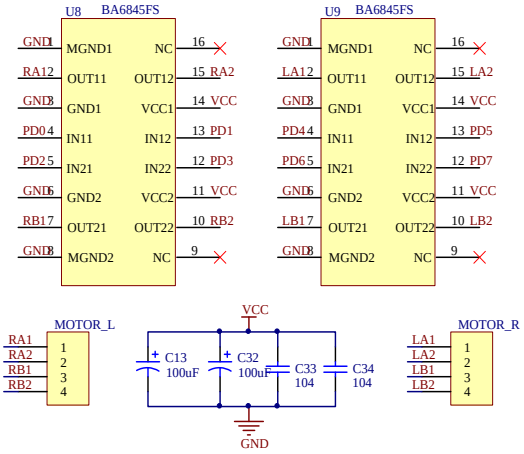


图 2.4 电机驱动电路

表 2.1 BA6845FS 真值表

IN11/21	IN12/22	OUT11/21	OUT12/22	模式
L	H	H	L	正向
H	H	L	H	反向
L	L	开路	开路	停止
H	L	开路	开路	停止

2.2.4 传感器电路

迷宫机器人利用红外线发光二极管搭配红外线感测器接收来自墙壁的反射，是用来检测迷宫墙壁的信息以及校正迷宫机器人行走姿态。这里论文探讨红外线发光二极管的工作原理与特性，分析下红外线发光二极管的测距原理。

调制（modulation）是将处理后的信号加到载波上，使得改变后的载波适合在信道传输的过程。简单来说，调制就是载波随信号改变的技术。信号源的信息成为基带信号，一般包含了频率较低的频率分量和直流分量。而基带信号往往不能直接作为传输信号，需要对基带信号进行升高频率转换成频率较高的信号在信道中传输，这个信号称为已调信号。基带信号也被称之为调制信号。调制的概念

为其基带信号是随着载体信号的相位、幅度或频率改变而改变的[17]。

调制是通过改变载体信号的幅度、相位或者频率，使其随着基带信号幅度的变化而变化来实现的[17]。输出信号称之为调制波，被控制信号称之为载波信号，输入信号称之为调制信号。调制波又根据调制信号所控制的载波信号参数类型分为调幅波、调频波和调相波[18-19]。图 2.5 所示为调幅。

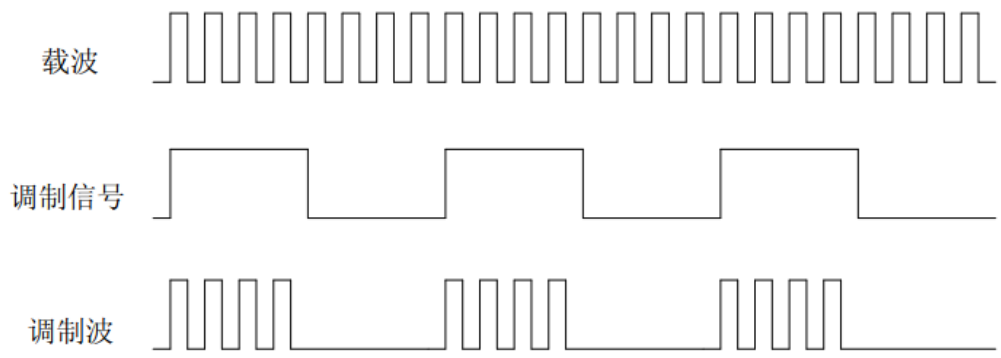


图 2.5 调幅示意图

论文选用的传感器型号为 IRM8601S，其含有 38KHz 中心频率的带通滤波器，故红外传感器最敏感的载波频率是 38KHz。由 IRM8601S 的数据手册可知，其调制信号的方波周期应为 1200μs [20]。当 IRM8601S 检测到信号时，输出低电平的有效电平，有效电平维持时间 TWL 的范围为：400μs<TEL<800μs。

不选用非调制的普通红外接收头是因为这种办法的抗干扰差，控制误差大。而选用调制的一体化接收头，这种方案的的抗干扰性得到了提高，但由于输出信号为数字信号，不能用输出信号的强弱来计算迷宫机器人和墙壁之间的距离。红外线的检测距离与发射的红外线强弱成正比。即当发射的红外线较强时，距离较远的障碍物也可以被检测到，而发射的红外线较弱时，只能检测距离较近的障碍物。控制红外线最直接的办法是改变其驱动电压或电流，但是这不适用于自动控制而要手动调节。论文引入了 D/A 控制，这样一来不仅适用于自动控制，也方便改变驱动电压。

论文利用了一体化的接收头抗干扰的关键原理来解决测距问题。如图 2.6 所示，红外线的载波频率越接近 38KHz，其衰减就越小；反之，越是偏离，其衰减就越大。利用这一特性，可以在不改变电压和电流的情况下，只改变载波频率就可以完成判断障碍物远近的任务。

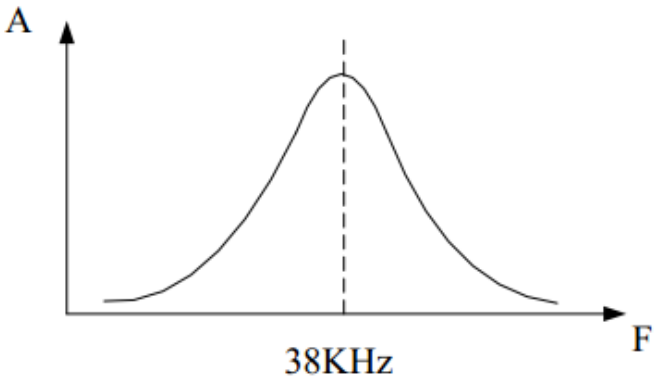


图 2.6 带通滤波示意图

迷宫机器人的传感器主要负责外部环境的监测和处理。选择传感器，主要考虑传感器在迷宫机器人中的作用是什么，需要达到什么样的效果。在论文中传感器的作用是判断周围是否有墙壁．所以适合使用红外传感器。论文选用了一体式红外线接收传感器，型号为 IRM8601S。IRM8601S 的工作电压为 5V，通常了考虑是可以把电池的输出电压经过 LD0 芯片稳压到 5V。但这样做的缺点是当电池电压瞬间改变被拉低时，传感器就不能得到稳定的电源提供，严重的影响了传感器的灵敏度。

经过以上考虑，决定采用 Exar 公司的低静态电流、高效率的升压芯片 SP6641A 把系统中的 3.3V 电压升到 5V，升压电路如图 2.7 所示。

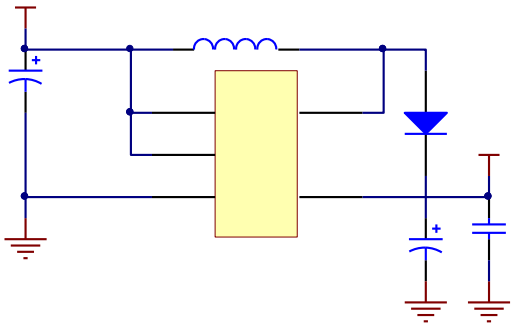


图 2.7 升压模块

红外线传感器用于迷宫墙壁的检测，分别位于迷宫机器人的右方、右前方、前方、左前方、左方五个方向，五个方向的传感器电路原理相同，其中的任意一个方向的红外线传感器电路如图 2.8 所示。

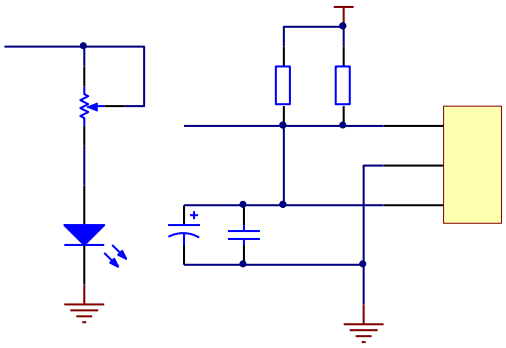


图 2.8 红外线传感器

论文的红外线传感器采用型号为 IRM8601S。该接收头内部集成带通滤波电路、自动增益控制电路及解码电路等。当它检测到载波频率为 38KHz 有效红外线信号时输出低电平，否则输出高电平。电路图中 RF2 为红外发射头，W2 为用来调节发射红外线的强度限流可调电阻。

2.2.5 JTAG 接口电路

迷宫机器人采用 10 引脚的 JTAG 接口。该接口和 LM LINK 兼容，其中 LM LINK 是由广州致远电子有限公司开发生产的 USB JTAG 调试器，它专门用于对 Luminary 系列单片机程序的调试和下载。它应用在 IAR 集成开发环境下，IAR Embedded Workbench for ARM（简称 IAR EWARM）是一个针对 ARM 处理器的集成开发环境。其管脚定义如图 2.9 所示。

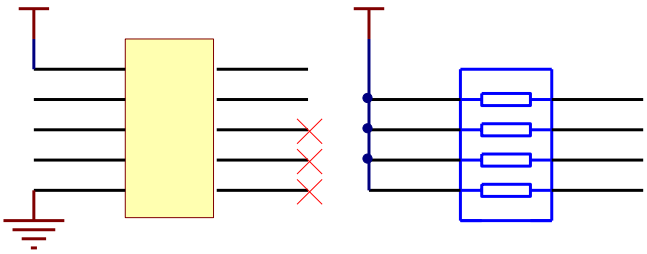


图 2.9 JTAG 接口电路

2.2.6 按键电路

迷宫机器人上有一个按键，其电路图如图 2.10 所示。电容 C24 用来消除按键抖动产生的毛刺干扰信号。

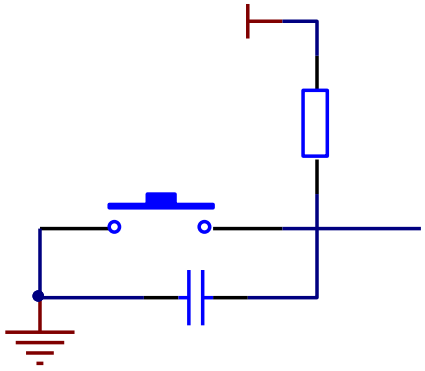


图 2.10 按键电路

2.3 迷宫机器人车体结构设计

2.3.1 齿轮简介

为了能让论文的迷宫机器人具有重量轻与低重心的特点，所以要对迷宫机器人的零部件作出探讨、比较，力求达到重量轻和重心低的要求。需要先瞭解齿轮的概念[21]。轮缘上有齿能连续啮合传递运动和动力的机械元件，图 2.11 齿轮的大小由齿轮模数和齿轮齿数来决定的，而其方程式为齿轮模数=节圆直径/齿数。

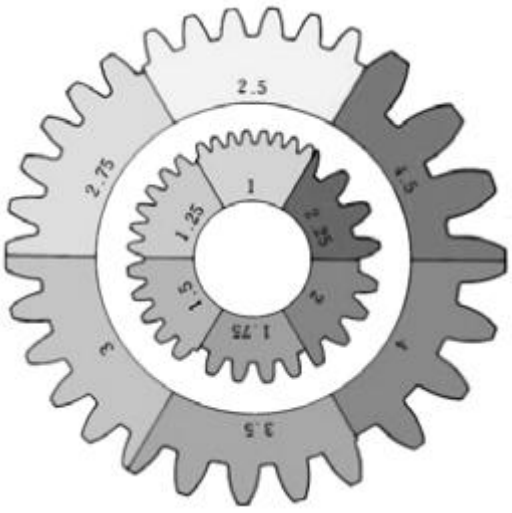


图 2.11 齿轮大小由齿轮模数和齿数组成

在轮面外有凹凸的齿，并令其相互嵌合，这就是齿轮的原型。两个齿轮的相切外缘称为此两个齿轮的节圆，因此两个能互相咬合的齿轮，其节圆必定相切于一点。所以知道了小齿轮和大齿轮的二个中心点的距离为二齿轮的中心距离=齿轮

模数（小齿轮齿数+大齿轮齿数）/2。

2.3.2 轮胎的组成

迷宫机器人的轮胎其实是由很多机械零件组成的，而且设计上是较为复杂的，为了让迷宫机器人在运动的时候可以拥有更平顺且稳定的状态，这里使用了双轴承以减少轮框造成的晃动，并尽量的减轻轮框的重量。图 2.12 为轮胎的组成图。

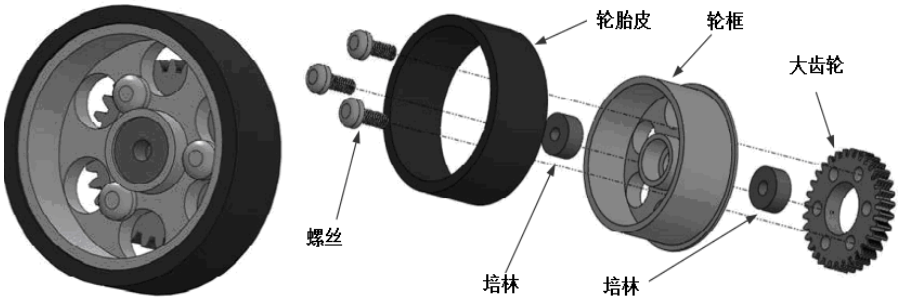


图 2.12 轮胎总体组成图

2.3.3 红外线传感器固定座

由于红外线传感器的发射与接收角度对迷宫机器人顺利完成探索任务非常重要，所以为了确保红外线传感器安装的角度不会随著迷宫机器人的不慎碰撞墙壁而改变，所以设计了红外线传感器固定座，除了固定红外线传感器发射与接收的角度外，另外还可以隔绝大部分外在的光源，可以降低环境光源对红外线传感器的影响。

2.3.4 迷宫机器人外观

电路设计上，除了要考虑电子零件的选型外，还需要注意到电子零件摆放的位置，以避免二者产生衝突。因为论文只使用电池当作电源供应，所以在布线的时候要特别注意数位讯号与类比的讯号的干扰[22]，本论文使用 Protel 来绘制相关的电路布线绘制迷宫机器人的原理图。

论文的迷宫机器人从硬体设计到现在，重量减轻了不少，外观也较精緻。电路上也由插件式元件改成表面黏著元件，在设计的过程中，除了不断的实验与测试外，也需要不断的吸收新知识，才可以保有创新与创意。

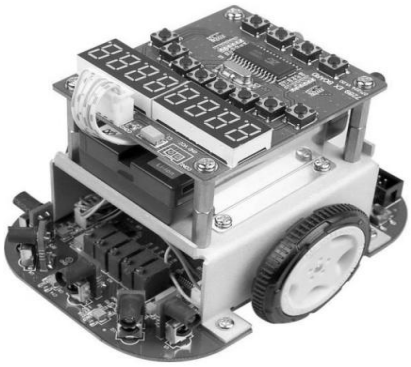


图 2.13 迷宫机器人

2.4 小结

本章首先对迷宫机器人硬件构成进行了整体介绍，并且分别按照功能模块详细说明了各个构成部分，例如微处理器、电机驱动模块、电源模块、传感器模块。绘制完成了原理图。对实物进行了各方面的硬件调试，确保为后续的代码设计提供稳定的硬件支持。

3 迷宫机器人的路径规划

3.1 路径规划概述

路径规划是指，在具有障碍物的环境中，按照一定的评价标准，寻找一条从起始状态到目标状态的无碰撞路径[23]。蒋新松在《水下机器人》中也为规划路径做出了这样定义：路径规划是自治式移动机器人的一个重要组成部分，它的任务就是在具有障碍物的环境内，按照一定的评价标准，寻找一条从起始状态(包括位置和姿态)到达目标状态(位置和姿态)的无碰路径[24]。由此可见，环境中障碍物与目标位置的不同直接影响到规划的路径。

路径规划是研究智能移动机器人的核心问题之一，同时也是研究人工智能问题的一个重要方面。通过研究使得未来的智能移动机器人能具有规划、感知和控制等高层能力。未来理想的移动机器人通过从周围的环境中收集信息，就可以构造符号化的环境模型，并且利用这个模型来执行由拥有者下达的高层任务[24]。当中的规划模块能生成大部分要执行的命令，这样就实现了移动机器人的使用者在较高层次上给机器人下达一些较宏观的任务，移动机器人系统自身就能填充那些较低层的细节问题，实现了移动机器人的自治。

路径规划既是移动机器人完成任务的有利保障，同时也是移动机器人自治化程度的重要标志。特别是在当前机器人的硬件系统在短期内无法得到大幅度提升的情况下，更需要将主要精力放在对路径规划算法的研究上面，对路径规划的研究可以有效地改变移动机器人的导航性能，提高移动机器人的自治水平，降低移动机器人在移动过程中出现的不稳定状态，提高移动机器人移动效率。路径规划的研究也为开发高智能的服务机器人、探测机器人、服务机器人、远距离搬运机器人和汽车自动驾驶系统打下基础。

3.2 迷宫的规范

最新的比赛规则是 2012 年国际电工和电气工程学会（IEEE）制定的电脑鼠走迷宫竞赛规则。其中在中国赛区的迷宫的规范制定如下[25]：

1. 迷宫由 16×16 个正方形单元所组成，其中每个单元的大小为 18cm×18cm。

2. 迷宫的墙壁高为 5cm，厚度为 1.2cm，因此两个墙壁之间所构成的通道的实际距离为 16.8cm。墙壁要将整个迷宫封闭。

3.迷宫墙壁的侧面为白色，顶部为红色，墙壁侧面和顶部的涂料能够反射红外线。迷宫的地面为木质，使用油漆漆成黑色，地板的涂料则能够吸收红外线。

4.迷宫的起始单元可选设在迷宫四个角落之中的任何一个单元格内。起始单元必须三面有墙壁，另一面存在出口。例如，如果没有墙壁的出口端为“北”时，那么迷宫的外墙就构成了位于“西”“南”“东”三面的墙壁。电脑鼠竞赛的终点设在迷宫中央，由四个的正方形单元格构成。

5.在每个单元格的四角可以插上一个小立柱，其截面为正方形。立柱长 1.2cm，宽 1.2cm，与墙壁厚度相同，高度为与墙壁同高的 5cm。小立柱所处的位置称为“格点”。除了终点区域的格点外，每个格点至少要与一面隔墙相接触。

6.迷宫制作的尺寸精度误差应不大于 5%，或小于 2cm。迷宫地板的接缝不能大于 0.5mm，接合点的坡度变化不超过 4 度。隔墙和之间的空隙不大于 1mm。

图 3.1 实际迷宫图

3.3 迷宫坐标的建立

由电脑鼠走迷宫竞赛规则可知迷宫是由行列各有 16 个方格，每个方格为 18cm*18cm 大小的正方形迷宫格组成的。为了让迷宫机器人记住所走过的各个迷宫格的信息，就要对迷宫的 16*16 一共 256 个迷宫格进行建立坐标系以便记忆迷宫机器人当前的位置和走过迷宫的墙壁信息。论文规定，迷宫左下角的迷宫格为(0,0)，最下方的一行的迷宫格编号依次为(0,0)、(1,0)一直到(f,0)。以此类推，最上方一行的迷宫格编号为(0,f)、(1,f)一直到(f,f)。可以看出，对迷宫格进行编号对于迷宫机器人记忆迷宫格是十分可行的。

论文规定如下，迷宫机器人放置于迷宫中的起点朝向方向为参照方向，Y 轴正方向为迷宫机器人的正前方，Y 轴负方向为后方，X 轴正方向为右方，X 轴负方向为左方。由于迷宫机器人的起点并不是唯一确定的，可能在(0,0)点，也可能在(f,0)点，故相对于迷宫机器人的终点方向可能在其右上方，也可能在其左上方，要根据其第一次的转弯方向确定。如图 3. 1 所示，如果迷宫机器人从(0,0)点出发，那么它第一次转弯方向是在它的右方, 同理，如果迷宫机器人从(f,0)点出发，那么它第一次转弯方向是在它的左方。

论文规定，迷宫机器人的向上方向定义为 0, 向右的方向定义为 1，向下的方向定义为 2，向左的方向定义为 3，如图 3.2 所示。

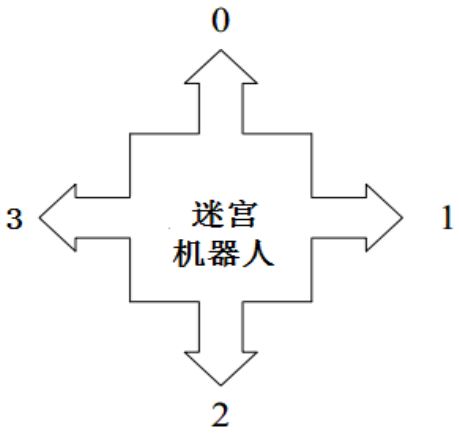


图 3.2 迷宫方向值的定义

3.3.1 相对方向和绝对方向的转换

对迷宫的坐标和方向定义后，迷宫机器人在其中行走就可以知道自己所在的单元格和行走的方向了。然而，对于迷宫机器人来说，红外线传感器的位置和方向是固定不变的，而对于迷宫来说，只要迷宫机器人转动方向其位置和方向就会发生改变，这个差异的出现是由于选取参照物不同产生的，由此引出了两个方向的问题：相对方向和绝对方向。其定义如下：

相对方向：参照方向为迷宫机器人当前前进方向，称之为为相对方向。

绝对方向：参照方向为迷宫绝对坐标平面，称之为绝对方向。

如图 3.2 中，迷宫机器人以符号 1 方向值前进，此时相对方向为向上前进，绝对方向为向右前进。从符号 1 方向值向符号 2 方向值转弯时，相对方向为向右转弯，绝对方向为向下转弯。

那么就要考虑红外线传感器所检测到的迷宫信息如何存储才能更利于处理。很明显，若以相对方向存储迷宫信息将会是十分混乱的，不仅存储起来麻烦，而

且记录和运用这些迷宫信息时，存储量也较大，处理起来也非常麻烦；而以绝对方向存储的资料就不必考虑迷宫机器人当时的方向即可进行处理，非常方便。

这样就会遇到相对方向与绝对方向互换。在论文中以变量 Dir 记录迷宫机器人前进方向上的绝对方向值，即迷宫机器人前方的绝对方向值始终为 Dir。这样迷宫机器人的相对方向转换为绝对方向如表 3.1 所示。

表 3.1 相对方向转换为绝对方向

相对方向	绝对方向
迷宫机器人前边	Dir
迷宫机器人右边	（Dir+1）%4
迷宫机器人后边	（Dir+2）%4
迷宫机器人左方	（Dir+3）%4

例如，若迷宫机器人当前的前进方向为迷宫的上方，此时 Dir=0，由表 3.1 可以计算出相对方向的左方向、后方向、右方向的绝对方向值分别为：3、2、1、1. 再参照图 3.2 可知，这三个值分别代表迷宫绝对方向的左方、下方和右方。可以看出，此时迷宫机器人的相对方向与绝对方向相同。

当前迷宫机器人的前进方向为迷宫绝对方向右方，即此时 Dir=3，由表 3.1 可以计算出其相对方向右、后、左三方向的绝对方向值为：0、1、2. 再参照图 3.2 可知，这三个值分别代表迷宫绝对方向的上方、右方和下方。可以看出，此时迷宫机器人的前方为迷宫的左方，左方为迷宫的下方，右方为迷宫的上方，后方为迷宫的右方。

上面讲的是相对方向向绝对方向的转换。有时还需要根据绝对方向求出相对方向，比如要控制迷宫机器人转向某一个绝对方向，这时就需要计算出该绝对方向处于哪个相对方向上，迷宫机器人根据相对方向来决定转向。

绝对方向向相对方向转换时，首先根据目标的绝对方向（Dir_dst）和当前的绝对方向（Dir）求出方向偏差值(△Dir)，如式 3-1 所示。

$$\Delta Dir = Dir\_dst - Dir \qquad \text{式(3-1)}$$

为了使△Dir 便于统一记录，其值必须都落在 0-3 的范围内，计算式改进为：

$$\Delta Dir = (Dir\_dst + 4 - Dir)\%4 \qquad \text{式(3-2)}$$

这时就可以根据方向偏差值求出迷宫机器人的相对方向，从而解决了绝对方向向相对方向的转换。如表 3.2 所示。

表 3.2 绝对方向转换为相对方向

绝对方向差值（ΔDir）	相对方向
0	迷宫机器人前边
1	迷宫机器人右边
2	迷宫机器人后边
3	迷宫机器人左边

综上可知，假设迷宫机器人已知当前位置坐标（X,Y），那么就可以求出其某绝对方向上的相邻坐标值，如表 3.3 所示，相对方向可按表 3.1 转换为绝对方向。该表是可逆的，即可以根据坐标值的变化求出绝对方向。从而建立起了相对方向，绝对方向，坐标三者之间的关系。

表 3.3 坐标转换

绝对方向	坐标
当前位置	（X,Y）
上边（0）	（X,Y+1）
右边（1）	（X+1,Y）
下边（2）	（X，Y-1）
左边（3）	（X-1,Y）

3.3.2 迷宫墙壁信息存储

当迷宫机器人到达一迷宫格时，需要运用一体式红外传感器 IRM8601S 检测结果并且记录下当前迷宫格的墙壁资料，为了便于管理和节省存储空间，每一个字节变量的低四位分别用来存储一个迷宫格四周的墙壁资料，如表 3.4 所示，迷宫共有 16*16=256 个迷宫格，所以可以定义一个 16*16 的二维数组变量来完整的保存整个迷宫墙壁资料。

迷宫墙壁资料的初始化变量全部为 0，凡是走过的迷宫格至少有一方没有墙壁，根据表 3.4 的迷宫墙壁信息存储方式，至少有一面墙壁信息为 1。这样就可以通过迷宫格存储的墙壁信息是否为 1 来确定该迷宫格是否已经被搜索过。

表 3.4 迷宫的墙壁信息存储方式

变量位	绝对方向	墙壁信息
Bit0	上方 0	1：无墙壁　0：有墙壁
Bit1	右方 1	1：无墙壁　0：有墙壁
Bit2	下方 2	1：无墙壁　0：有墙壁
Bit3	左方 3	1：无墙壁　0：有墙壁
Bit7-Bit4	保留	保留

迷宫机器人经常需要判断自己某个相对方向上的相邻格是否走过，而判断某个迷宫格是否走过的方法就是判断该坐标上的墙壁信息，如表格 3.4 所示，凡是走过的迷宫格格其墙壁信息资料为 1。

其 C 语言代码如下，算法思想为：把相对方向转换为绝对方向，根据绝对方向获取相邻格坐标，返回相邻格坐标上的墙壁信息。

```
uchar mazeBlockDataGet (uchar ucDirTemp)
{
char cX = 0,cY = 0;
/* 把迷宫机器人的相对方向转换为绝对方向 */
switch (ucDirTemp) {
case MOUSEFRONT:
ucDirTemp = GucMouseDir;
break;
case MOUSELEFT:
   ucDirTemp = (GucMouseDir + 3) % 4;
   break;
case MOUSERIGHT:
   ucDirTemp = (GucMouseDir + 1) % 4;
   break;
default:
break;
}
/* 根据绝对方向计算该方向上相邻格的坐标 */
switch (ucDirTemp) {
```

```
    case 0:
      cX = GmcMouse.cX;
      cY = GmcMouse.cY + 1;
    break;
    case 1:
      cX = GmcMouse.cX + 1;
      cY = GmcMouse.cY;
    break;
    case 2:
      cX = GmcMouse.cX;
      cY = GmcMouse.cY - 1;
    break;
    case 3:
      cX = GmcMouse.cX - 1;
      cY = GmcMouse.cY;
    break;
  default:
  break;
  }
  return(GucMapBlock[cX][cY]);   /*   返回迷宫格上的资料   */
  }
```

3.4 记忆功能的实现

3.4.1 等高图制作原理

等高图是等高线地图的简称，例如一般地图需要标出同一高度范围的地区和高度值，或者像气象报告时的等气压图，要标出相等气压的范围及数值。以此类推，论文将等高图运用在迷宫地图上，就可以标出每个迷宫格到起点迷宫格的步数数值，有了这些数值信息，在迷宫机器人遇到许多三面都有墙壁的死路和需要作出路径选择时就可以运用制作出等高图来解决这些问题，使迷宫机器人更容易

走出死路，少走一些已经走过的路径。

论文需要先创立一个 16*16 的二维数组空间（MapStep[16][16]），其中二维数组里的每一个元素代表迷宫中的一个迷宫格，用来储存迷宫中各迷宫格至起点的最短路径步数，即就是路径中经过的迷宫格数。

当起点坐标处标识为 1 时，可以直接到达的相邻方格均为 2，再远的迷宫格的等高值依次递增。以此类推，距离越远的迷宫格等高值越大。

3.4.2 等高图的实现

论文为了说明简明，这里用 4*4 的小迷宫作为范例，并以坐标为(0,0)迷宫格作为起点。对于 16*16 并且起点在(0,0)的迷宫，等高图的制作原理是一样的。

步骤一：

1. 将所有迷宫格等高值填为 0xff.（0xff 是迷宫格等高图的值的最大值）

2. 迷宫的起点坐标记为(0,0)，其等高值为 1.

3. 到达迷宫的(1,0)坐标后，记录等高值的变量 Step 加 1，由 1 变为 2.

4. 在堆栈中存入起点坐标(0,0)。

5. 得到如图 3.3 所示。

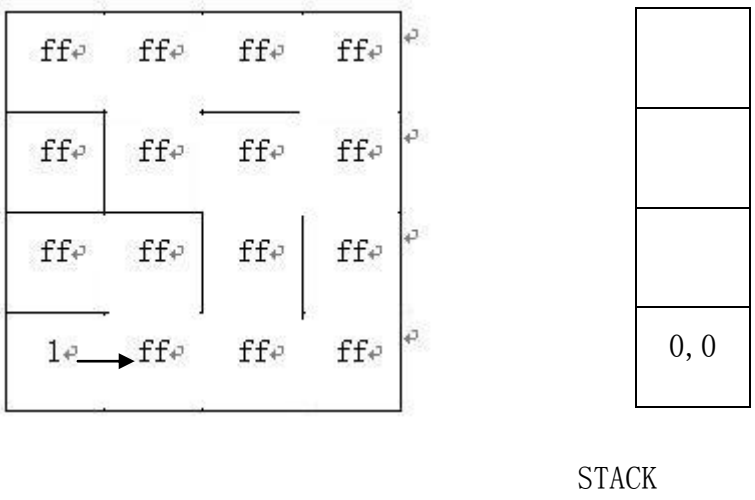


图 3.3 等高图 1

步骤二：

1. 观察图 3.3 中的(0,0)点四周仅有一个可前进方向。可前进的定义是说箭头所指方向的迷宫格等高值比当前坐标上的值大于 2 以上，如坐标(1,0)的值为 0xff, 则 0xff-1>1，则坐标(1,0)为可前进方向。

2. 进入箭头所指方向的方格，此时到达坐标为(1,0)迷宫格。

3. 等高值的变量 Step 加 1，坐标(1,0)的等高值变为 2.

4. 得到如图 3.4 所示。

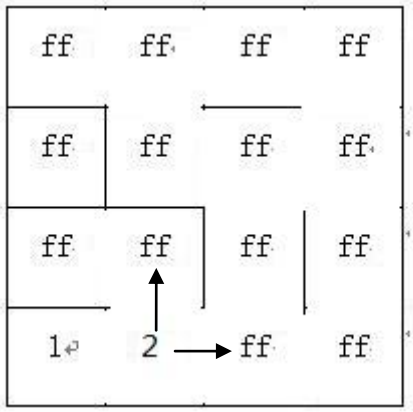


图 3.4 等高图 2

步骤三：

1. 图 3.4 中(1,0)四周有两个可前进方向，则此坐标为一岔路口。

2. 如坐标为岔路口，则将此坐标存入堆栈。

3. 进入箭头所指的任意一个方向（可以任意选一个，走迷宫是会规定算法来解决行走方向），此时坐标为(1,1)。

4. Step=Step+1. 到达坐标(1,1)。

5. 得到如图 3.5 所示。

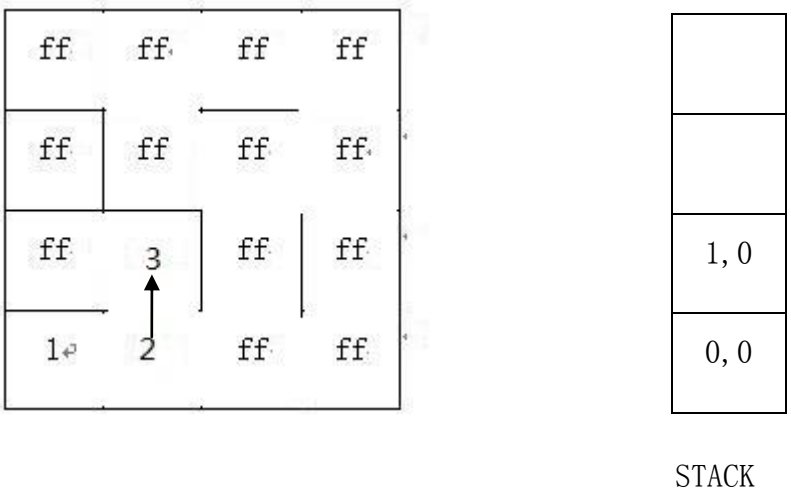


图 3.5 等高图 3

步骤四：

1. 图 3.5 仅有一个可前进的方向。

2. 到达箭头所指迷宫格，此时坐标为(0,1)。

3. Step=Step+1.

4. 得到如图 3.6 所示。

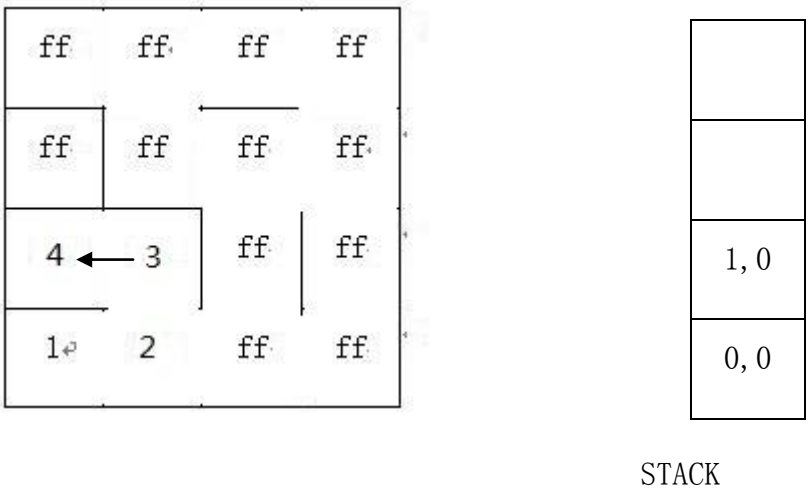


图 3.6 等高图 4

步骤五：

1. 坐标在(0,1)时，四周没有可前进方向。

2. 取出栈顶中的内容，调到其保存的坐标(1,0)。

3. Step=Step+1.

4. 得到如图 3.7 所示。

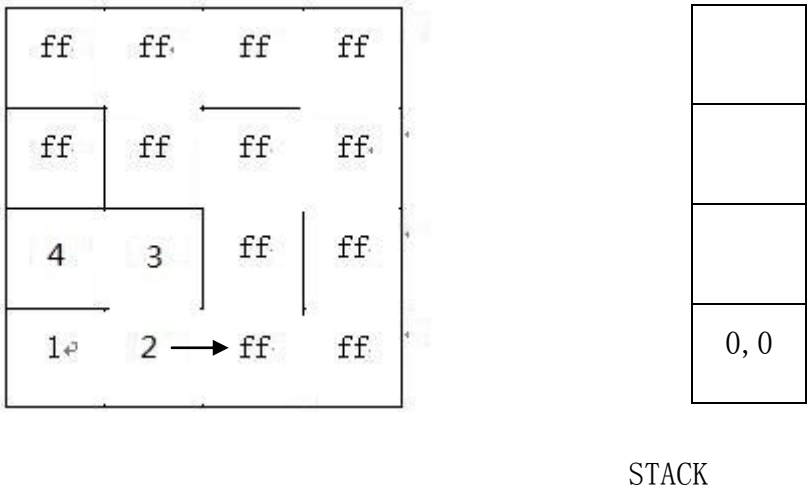


图 3.7 等高图 5

步骤七：

1. 图 3.8 中坐标(1,0)点有一个可前进方向，坐标(0,0)处 Step 值为 1，不比当前坐标的步数值大于 2 以上，所以不是可前进的方向。

2. 进入箭头所指方向，此时坐标为(2,0)。

3. Step=Step+1.

4. 得到如图 3.8 所示。

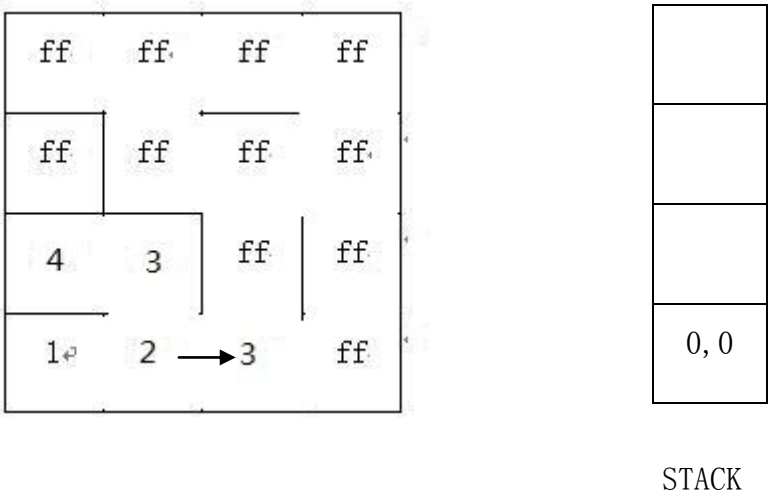


图 3.8 等高图 6

步骤八：

1. 图 3.8 中的坐标(2,0)有两个可前进方向。

2. 将该岔路口坐标入栈。

3. 进入箭头所指的任意方向，选择进入坐标为(2,1)的迷宫格。

4. Step=Step+1.

5. 得到图 3.9.

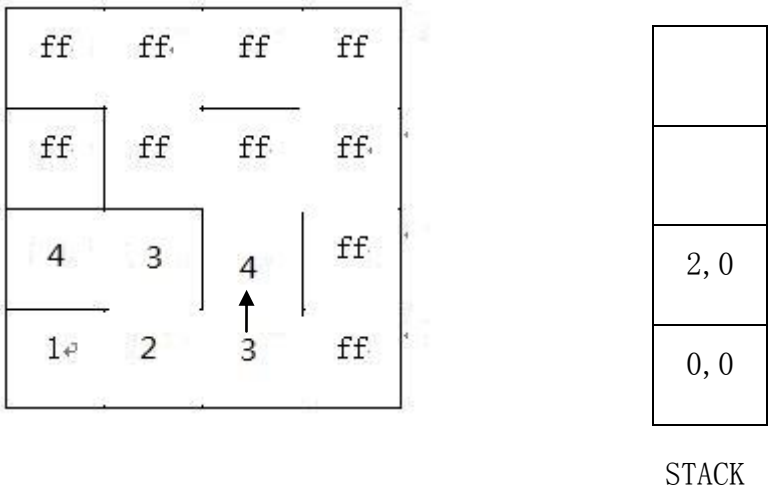


图 3.9 等高图 7

以此类推，直到到达的坐标没有可前进方向，并且堆栈中没有未处理完的坐标点时记为结束。最终可以得到如图 3.10 的等高图。等高图的数字即为步数，也就是代表从起点(0,0)出发到达该坐标的最少迷宫格数，如坐标(3,3)距离起点有 7 步，坐标(2,3)距离起点有 8 步，坐标(0,2)距离起点为 ff, 记为不可到达的坐标点。

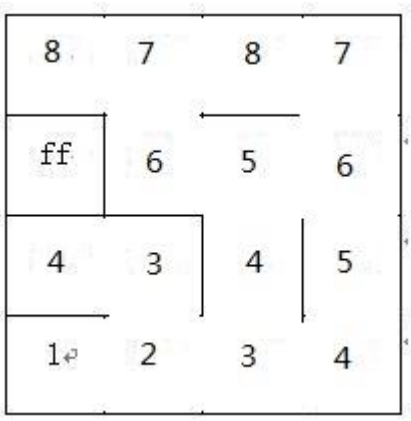


图 3.10 等高图 8

3.4.3 程序设计

程序代码如下：

```
void mapStepEdit (int8    cX, int8    cY)
{
     uint8 n= 0;                    /*    GmcStack[]下标     */
    unit8 ucStep=1;                 /*    等高值             */
    unit8 ucStat=0;                 /* 统计可前进的方向数*/
    uint8 i,j;
 GmcStack[n].cX= cX;                /*    起点 X 值入栈      */
  GmcStack[n].cY= cY;               /*    起点 Y 值入栈      */
     n++;
/*   初始化各坐标等高值                                      */
    for (i = 0; i < MAZETYPE; i++) {
          for (j = 0; j < MAZETYPE; j++) {
               GucMapStep[i][j] = 0xff;
          }
     }
/*    制作等高图，直到堆栈中所有数据处理完毕               */
     while (n) {
          GucMapStep[cX][cY]=ucStep++;
/*   填入等高值                                              */
/*   对当前坐标格里可前进的方向统计                          */
```

```
            ucStat = 0;
    if((GucMapBlock[cX][cY]&0x01)&&  (GucMapStep[cX][cY  +  1]  >  (ucStep)))
/*   前方等高值大于计划设定值                                          */
      {ucStat++;                        /*   可前进方向数加 1 */
          }
           else {
                if(ucStat>1)          /* 有多个可前进方向，保存坐标   */
    {
    GmcStack[n].cX = cX;              /*   横坐标 X 值入栈      */
    GmcStack[n].cY = cY;               /*   横坐标 Y 值入栈      */
        n++;
                }
     /*任意选择一条可前进的方向前进                                      */
     if((GucMapBlock[cX][cY]&0x01)&&(GucMapStep[cX][cY + 1] > (ucStep)))
    /*   上方等高值大于计划设定值                                       */
    {
                    cY++;           /*   修改坐标             */
                continue;
            }
    }
```

迷宫机器人的程序设计加入等高图其实就是对其记忆功能的实现，能让迷宫机器人记住走过的路，对整个迷宫探索后构建出等高值地图。

3.5 记忆功能的优化

3.5.1 竞赛的规则

迷宫机器人的基本功能是从起点开始走到终点所花费的时间称为“运行时间”。迷宫机器人从第一次激活到运行开始所花费的时间称为“迷宫时间”。迷宫机器人在比赛时手动辅助的动作称为“碰触”[26-29]。

迷宫机器人的得分通过计算每次运行的“排障时间”来衡量，即将迷宫时间加上一次运行时间的 1/30。如果被碰触过，那再减去 10 秒，这样得到的就是排

障时间。迷宫机器人在迷宫中停留或运行的总时间不可超过 15 分钟，在限时内允许运行多次。

迷宫机器人上电后，首先进行探测和记忆，这次运行被称为“试跑”。在试跑时，需要按照一定的算法得到墙壁信息，并到达终点，即迷宫搜索算法；在搜索结束后根据搜索所获得的迷宫墙壁信息，按照最优路径算法确定最佳路径，并以最快的速度到达目的地，这次运行被称为“冲刺”。冲刺后还可以继续多次试跑和冲刺。

3.5.2 记忆策略

迷宫机器人位于某个迷宫格时，需要完成下面的任务:

1. 检测是否到达过该迷宫格。

2. 通过红外线传感器探测迷宫格可以行走的方向。

3. 根据该迷宫格可走的方向数结合算法，选择下一步走的方向。

4. 检测出附近封闭的循环路径和死路，并更新相应的墙壁信息。

5. 根据坐标检测是否到达终点坐标。

6. 移动到下一个迷宫格，更新当前坐标。

迷宫机器人的主要任务就是能够在各种不同迷宫环境下提出下一步怎么做的解决方案。有三种环境，分别对应三种不同的处理程序：死路程序、单路程序和岔路程序。

当迷宫机器人面对的情况是左右中三面都有墙壁时，需要执行死路程序。死路程序会将当前所处迷宫格的信息标记为真，表示无论在探索还是寻找最短时都不进入该迷宫格，并且将迷宫机器人向后旋转 180° 。将图 3.11 用上述方法建立坐标系后，在坐标为(0,0)的迷宫格中,迷宫机器人三面都有墙壁。如果只有一条路可以前进时，执行单路程序。单路程序会选择唯一的可前进路径。从坐标为(0,3)到(0,5),从(5,3)到(5,4)的迷宫格，也执行单路程序。如果有两条或是以上的路可以前进时，则执行多路程序，如坐标为(0,1)的迷宫格。多路程序需要根据选定的洪水算法结合搜索法则选择之后的路径。一种情况是至少有一个迷宫格没有走过。这样的话，迷宫机器人选择的没有走过的迷宫格作为前进方向。还有一种可能是有超过一个迷宫格未走过，则需要根据算法来进一步决定。

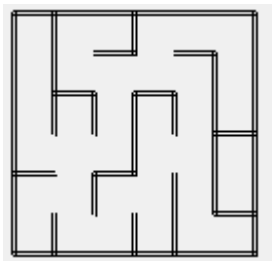


图 3.11 迷宫

3.5.3 三面有墙的路径标记与优化

当迷宫机器人行走至图 3.11 中(0,5)坐标点时，只有一面没有墙壁，迷宫机器人的三面都有墙壁。迷宫格(0,5)被标记为真，迷宫机器人向后旋转 180°。当遇到此类路径时，迷宫机器人应该标记除了起点和终点的所有属于该路径且执行单路程序的迷宫格。图 3.11 中被标记为三面都有墙壁的路径终点是(0,5)，(1,3)，(2,5)，(3,0),(3,3),(5,0),(5,3)。三面有墙壁的迷宫路径分别为(0,3)→(0,4)→(0,5), (1,2)→(1,3), (1,5)→(2,5),(3,1)→(3,0),(3,2)→(3,3),(4,0)→(5,0),(5,4)→(5,3）。一旦迷宫格被标记为属于三面都有墙壁的路径，下一次迷宫机器人经过该路径时，将不会把该路径作为可选路径。从而节省了探索时间，提高了搜索效率。

3.5.4 行走环内部的避免

由于迷宫竞赛的比赛地图是随机抽取的，因此迷宫具有随机性，这里论文考虑到其中一些迷宫在迷宫机器人在探索过程中的特殊情况：第一种情况，若从某个迷宫格出发经过一些迷宫格又回到了该迷宫格，行走路径形成环形；第二种情况，迷宫机器人从迷宫的外围墙壁一侧探索到另外一侧的路径与迷宫的最外围墙壁形成环形。以上两种情况则会被标记为环。可以通过程序来判断行走的路径是否为环。形成环的一个前提是环内部不包含迷宫终点的坐标。当迷宫机器人到达某迷宫格，发现该点以前已经走过则检查前一个迷宫格，直到发现某个迷宫格有岔路口选择。判断该迷宫格是否为当前迷宫格，如果是且迷宫终点的坐标不在环内，则标记环内的迷宫格为死路；如果不是，则表明不是环。如图 3. 12 所示，坐标(3,0), (3,1), (3,2), (4,0), (4,2), (5,0), (5,1), (5,2)组成了环。当迷宫机器人探索完上述坐标后第二次到达(3,1)迷宫格，发现相邻的三个迷宫格全部走过，则检测前一个迷宫格 ，例如前一个迷宫格(3,0)，检测到该迷宫格全部走过，则再检测上一个迷宫格，直到遇到有多路选择的迷宫格。本例中检测迷宫格会退回到迷宫机器人的开始位置(3,1)，所以可以得到迷宫格(3,0), (3,1), (3,2), (4,2), (5,2), (5,1), (5,0),

(4,0)是环，将这些迷宫格标记为真。通过标记死路和环，算法思想是不会进入死循环和探索不必要的路径。

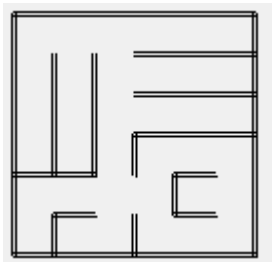


图 3.12 迷宫封闭循环路径

算法流程图如图 3.13 所示：

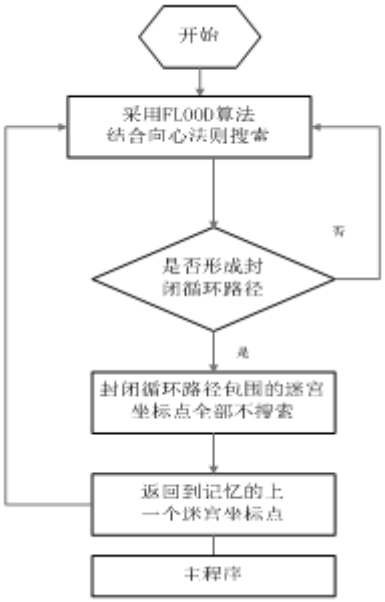


图 3.13 迷宫封闭循环路径算法流程图

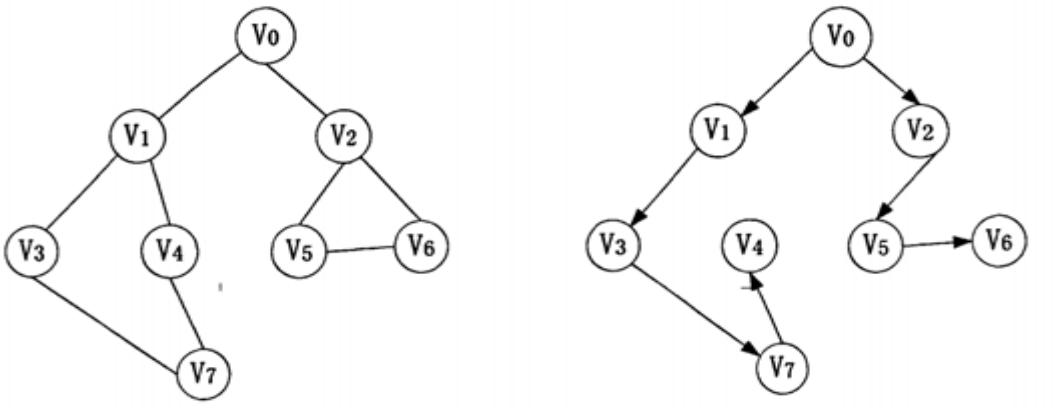
3.6 迷宫算法

在没有预知迷宫墙壁信息的情况下，迷宫机器人必须要先探索迷宫中的所有单元格，直到抵达终点为止。在探索迷宫时, 迷宫机器人要随时知道自己的位置和姿势，同时要记录下所有访问过的迷宫单元格四周的墙壁信息。在探索过程中，还要尽量避免重复走已经走过的地方。如果考虑迷宫机器人尽快到达目的地，可以缩短探索迷宫所需的时间，但是不一定能够构建整个迷宫地图，找到的路径不一定是最短路径。如果考虑搜索整个迷宫，可以得到整个迷宫的墙壁信息，这样就可以求出最优路径，但可能在探索迷宫时消耗更多时间。论文探讨下几种迷宫搜索算法。

3.6.1 深度优先搜索算法

深度优先搜索算法和类似于树的先根遍历，是树的先根遍历的推广[30]。其基本思想是：假定图中所有节点都没有被访问过，则该算法可以从图中任何一个节点v出发, 访问次节点，然后依次从v的没有被访问到的相邻节点出发深度优先遍历图，直到图中的所有的和v路径相通的节点都被访问结束。

以图 3.14(a)中无向图 G 为例，深度优先搜索如图 3.14(b)所示，从节点v_0出发开始搜索，在访问节点v_0之后，选择邻接点v_1。由于v_1没有访问过，则再从v_1开始出发进行搜索。以此类推，接着搜索v_3、v_7、v_4。在访问了v_4节点之后，由于v_4的邻接节点都已经被访问过，则搜索回到v_7。出于同样的理由，搜索回到v_3、v_1、v_0，此时由于v_0的另一个邻接节点没有被访问过，搜索又从v_0到v_2接着访问下去。由此，得到的节点访问序列为：$v_0 \rightarrow v_1 \rightarrow v_3 \rightarrow v_7 \rightarrow v_4 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6$。



（a）G 的无向图　　（b）G 的深度优先探索

图 3.14 深度优先探索过程

因为深度优先搜索是一个递归的过程，其搜索本质上是访问每一个节点的邻接节点。深度优先搜索所花费的时间取决其采用的存储结构。假设图中的节点个数为n，图用邻接矩阵来表示时，搜索一个节点的所有邻接节点的时间花费就为$O(n)$，从n个节点出发搜索的时间就为$O(n^2)$，所以深度优先搜索算法的时间复杂度就为$O(n^2)$。

3.6.2 广度优先搜索算法

广度优先搜索算法和类似于树的按层次遍历的过程，是树的按层次遍历的延伸[31]，其基本思想是：假定开始从图中节点v出发，访问了v之后依次访问v的没有被访问过的各个邻接点,然后从这些访问的邻接点出发依次访问它们的邻接点，先被访问的节点的邻接点先于后被访问的节点的邻接点被访问，以此类推，直到

图中所有节点的邻接点都被访问到。

选取图 3.14(a)的无向图 G 为例，深度优先搜索如图 3.15 所示，首先访问节点v_0和v_0的邻接点v_1、v_2。如果先访问节点v_1后访问节点v_2，那么v_1的邻接节点先于v_2的邻接节点被访问，即类似于队列的先进先出存储特性。如果先访问节点v_1，则接着访问节点v_3、v_4。当v_1的所有邻接点访问结束后，开始访问v_2的邻接点v_5、v_6，最后访问v_3邻接点v_7。此时，这些节点的邻接点均已被访问，完成了图的搜索。得到的节点访问顺序为：$v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7$。

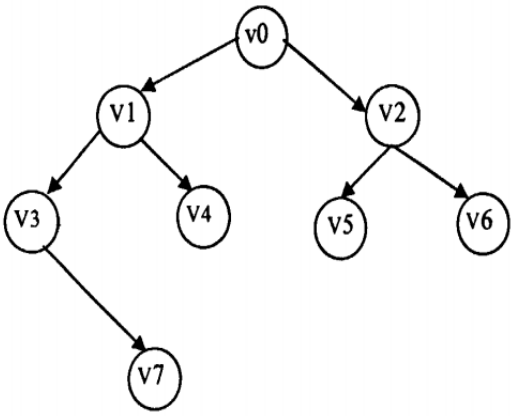


图 3.15 广度优先探索过程

深度优先搜索算法每个节点最多进入一次队列，故深度优先搜索算法的实质是通过边找邻接节点的过程。在和广度优先搜索算法采取相同的存储结构的情况下，深度优先搜索算法的时间复杂度和广度优先搜索算法相同，都为$O\left(n^2\right)$。

3.6.3 Flood 算法

Flood 算法[32]的基本概念就好像洪水流动一样。Flood 算法也称为贝尔曼算法，该算法利用距离和迷宫格的关系去寻找到达迷宫中心的较短路径。在这个算法当中，每一个迷宫格都存在一个代表距离的物理量，此物理量代表该迷宫格与迷宫中心可能的曼哈顿距离。一开始时假设迷宫中没有墙壁存在，每个迷宫格距离终点的距离如图 3.16 所示。

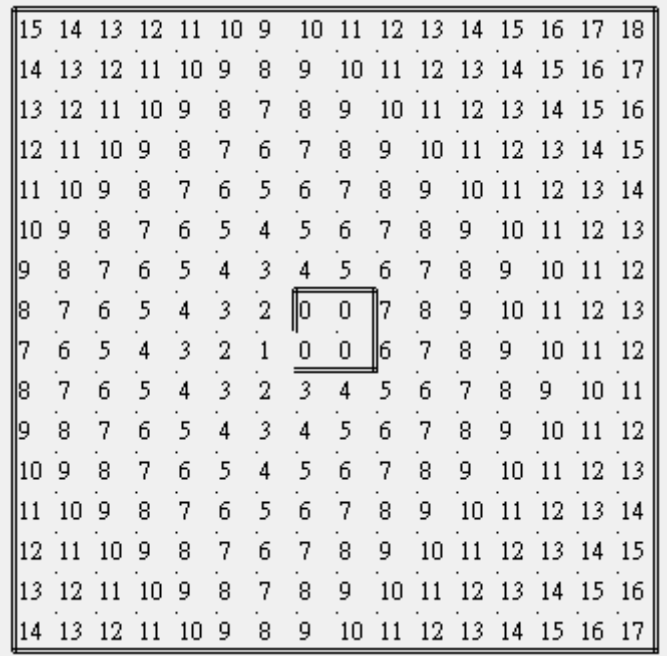


图 3.16 无墙壁迷宫每个迷宫格距离终点步数

这里以 16×16 迷宫大小为例，首先按论文 3.3 章节建立坐标系。迷宫起点为（0,0），迷宫终点为迷宫中心, (7,7), (7, 8), (8, 7), (8, 8) 四个迷宫格，一开始先不考虑直行或转弯对洪水流速的影响，即迷宫中距离为 1 的迷宫格无论是直走还是转弯都认为其用时相同，在论文 4.3 章节中也会提出并解决转弯对速度的影响。假设迷宫终点的距离值为 0，洪水每流过 1 个迷宫格，该迷宫格与上一个迷宫格相对于迷宫终点的距离值加 1，利用这个思想来计算每一个迷宫格与迷宫终点的距离值，持续到计算出迷宫起点相对於迷宫终点的距离值为止。依据每一个迷宫格与迷宫终点的距离值，再由大到小的方式排列，这样就可以在迷宫中找出一条最短路径。

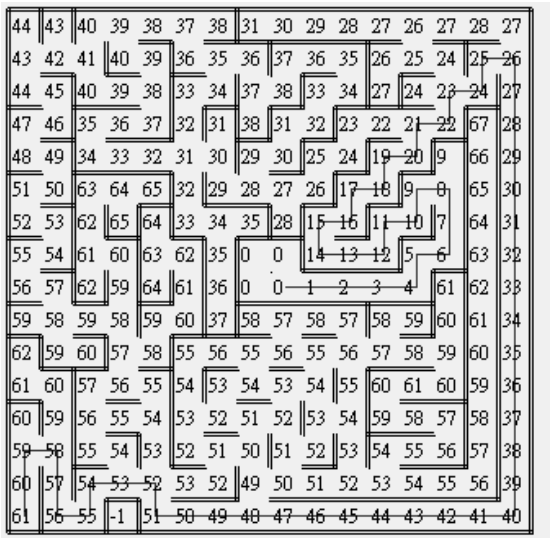


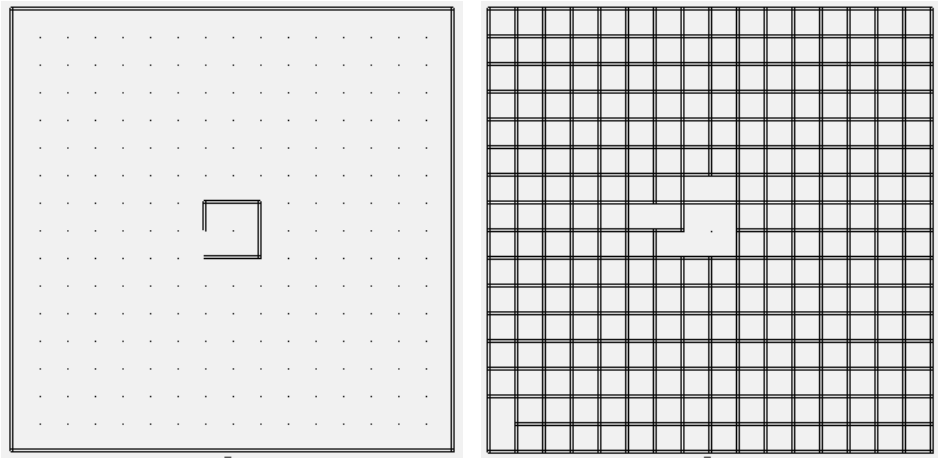
图 3.17 迷宫中 FLOOD 算法得到的最短路径

3.6.4 Flood 算法结合中右左法则

在迷宫机器人的竞赛中，一开始并不会知道所选用的迷宫地图，即迷宫中的具体墙壁信息是不知道的，所以迷宫机器人需要一个有效的迷宫算法，这个迷宫演算法除了需要记忆迷宫墙壁信息外，还要在迷宫中找出一条从起点找到终点的最短路径。如果想要让迷宫机器人可以在迷宫中找出一条最短路径，那么就需要先搜寻迷宫墙壁信息。

首先是对迷宫建立坐标系，以迷宫大小 16×16 为例，左下角起点 S 的坐标为(0,0)，往上正北方向为 Y 轴正方向，往右边正东方向为 X 轴正方向。因此，起点 S 右边迷宫格的坐标依序为(0,0)到(0,15)，起点 S 上方第一排迷宫格往右边的坐标依序为(1,0)至(1,15)，依此类推。

建立如图 3.18 的二种关于迷宫格的信息地图：图 3.18(a)除了迷宫四周与起点(0,0)终点墙壁信息已知外，其他迷宫格的预设值都是没有墙壁；图 3.18(b)除了起点(0,0)和终点墙壁信息已知外，其他迷宫格预设值都是有墙壁。



（a）没有墙壁迷宫　　（b）有墙壁迷宫

图 3.18 迷宫

中右左法则[33]是当迷宫机器人行进探索迷宫的时候，如果碰到有两个以上的方向可以选择时，假定了选择的优先顺序是中右左，也就是首先如果原来行进的方向可以走则走原方向，其次选择右转这条路径，最后选择左转方向。因此，可以从图 3.19 看出，一开始迷宫机器人在迷宫起点朝 Y 轴正方向移动。每到一个未走过的迷宫格开始更新墙壁信息并存贮这一迷宫格的信息。到了坐标为(0,2)的迷宫格更新墙壁资讯后，迷宫机器人依据中右左法则朝(0,3)的迷宫格前进。到达坐标(0,3)的迷宫格后，由于前方以及左右两边都没有路径，因此第一次在三面都有墙壁的路径终点前停下来。

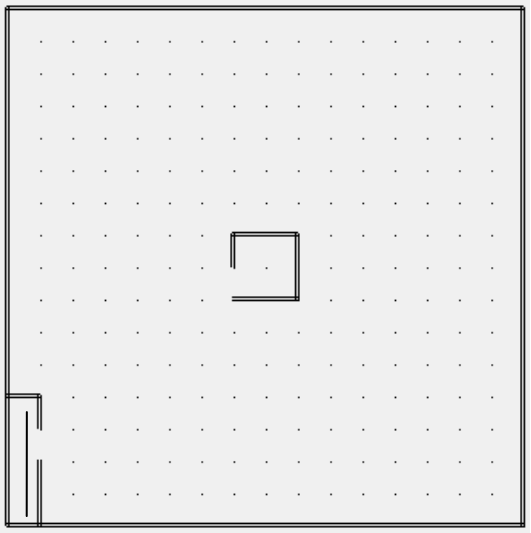
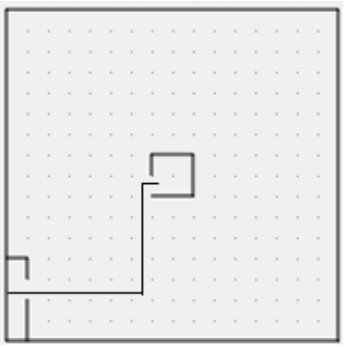
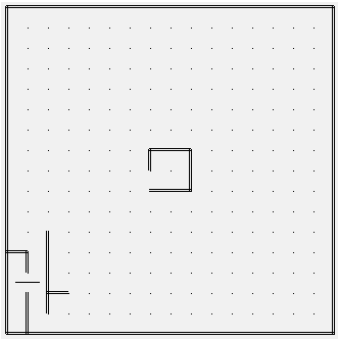


图 3.19 Flood 算法结合中右左法则

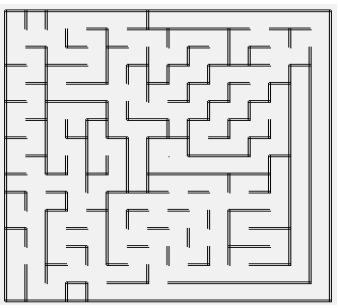
根据现有的迷宫墙壁信息，图 3.20(a)利用没有墙壁的迷宫信息地图，执行 Flood 算法，此时 Flood 算法会提供一条假设性的路径，让迷宫机器人沿着这条假设性的路径继续前进。当迷宫机器人发现有其他的分支路口时，即再次应用中右左法则进行判断。图 3.20(b)这条假设性的路径会使迷宫机器人行进至(1,2)的迷宫格，(1,2)是尚未走过的迷宫格。图 3.20(c)为实际迷宫信息地图。



(a)



(b)



(c)

图 3.20 没有墙壁的迷宫信息，Flood 算法算出的一条路径

迷宫机器人重复上述的思想，在遇到死路或是无法利用中右左法则时，再次

执行 Flood 算法，让 Flood 演算法提供一条假设性的路径。迷宫机器人就可以在任何 16*16 的迷宫地图(起点到终点有路可走)顺利找到终点，搜索后最短路径为(0,0)→(0,1)→(0,2)→(1,2)→(1,1)→(1,0)→(2,0)→(2,1)→(3,1)→(4,1)→(4,0)→(5,0)→(6,0)→(7,0)→(8,0)→(9,0)→(10,0)→(11,0)→(12,0)→(13,0)→(14,0)→(15,0)→(15,1)→(15,2)→(15,3)→(15,4)→(15,5)→(15,6)→(15,7)→(15,8)→(15,9)→(15,10)→(15,11)→(15,12)→(15,13)→(15,14)→(14,14)→(14,13)→(13,13)→(13,12)→(12,12)→(11,12)→(11,11)→(10,11)→(10,10)→(9,10)→(9,9)→(9,8)→(10,8)→(11,8)→(11,9)→(12,9)→(12,10)→(13,10)→(13,9)→(13,8)→(12,8)→(12,7)→(11,7)→(10,7)→(9,7)→(8,7)(终点)。图 3.21 为利用 Flood 算法结合中右左法则找到迷宫起点到终点的最短路径。

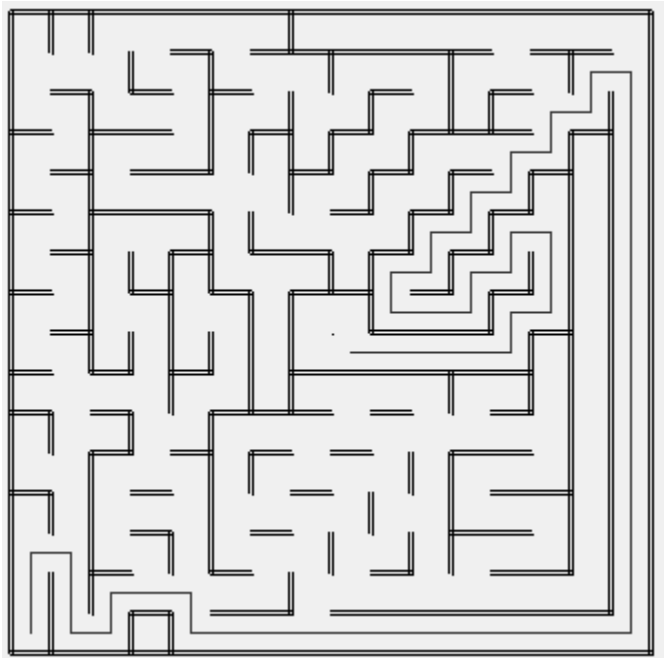
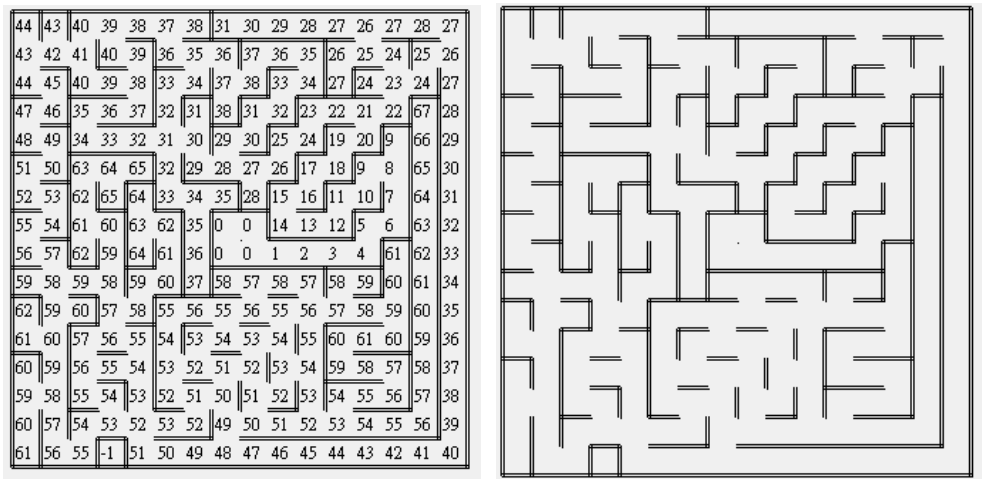


图 3.21 Flood 算法结合中右左法则找到迷宫终点

当迷宫机器人走到迷宫终点的时候，利用已经探索到的迷宫地图信息，执行 Flood 算法找出“最短路径”，可以看出这条路径，从起点 S 距离终点 G 的距离值为 62，即走过的迷宫格数为 62 格，但这条“最短路径”不一定是最短的，原因是迷宫墙壁信息不足所造成的，也就是说没有对迷宫进行全局探索。

为了使迷宫机器人在搜索的时候，能够找到全部的迷宫信息，构建起一个完整的迷宫地图，论文需要当迷宫机器人搜索到终点的时候，再次利用 Flood 算法结合中右左法则走回起点。在实际的程序中，Flood 算法需要将迷宫的终点 G 与迷宫的起点 S 互换，以避免对迷宫格坐标编号计算时造成的错误。图 3.22(a)为从终点搜索到起点后构建的完整迷宫信息地图，其中每一迷宫格的数字为该迷宫格

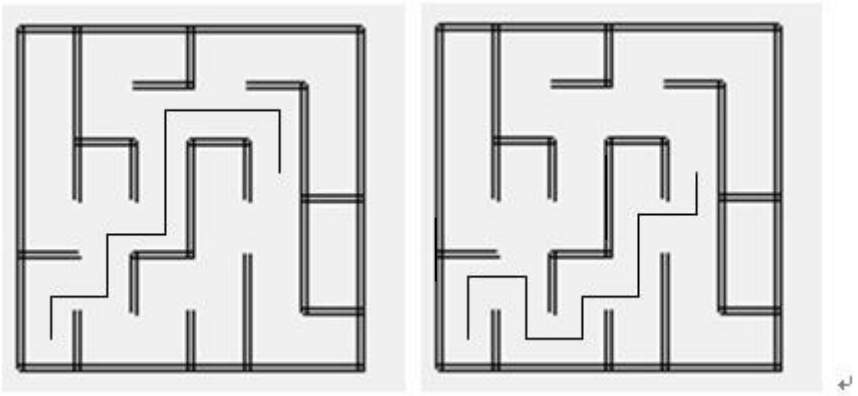
距离迷宫终点的最短步数。图 3.22(b)为实际迷宫信息地图。



(a) (b)

图 3.22 最终 Flood 算法结合中右左法则搜索

需要考虑到一种特殊情况是：如果到达某一点有两条同样长度的路径时，应该如何选择。假设迷宫地图如下，起点 S 为左下角(0,0)点，终点为右上角(5,5)点。迷宫机器人在坐标为(4,3)的迷宫格时，利用没有墙壁的迷宫信息地图，执行前往起点 S 的 Flood 算法时，发现从(4,3)的迷宫格到起点 S，有二条一样长度值 9 的建议路径，请参考图 3.23。



(a) (b)

图 3.23 在坐标为(4,3)的迷宫格发现二条长度相同的建议路径

图 3. 24 选择其中坐标为(4,4)的迷宫格为行走路径。搜寻迷宫后的最短路径为(4,3)→(4,4)→(3,4)→(2,4)→(2,3)→(2,2)→(1,2)→(1,1)→(0,1)→(0,0)(起点)。如果选择坐标为(4,4)的迷宫格这条假设性的路径，从(4,4)的迷宫格回到起点 S，利用有墙壁的迷宫信息地图，执行 Flood 算法所找到的路径，可以发现，从起点 S 距离终点 G 的距离值为 11，这是一条最短路径，可参考图 3.24。

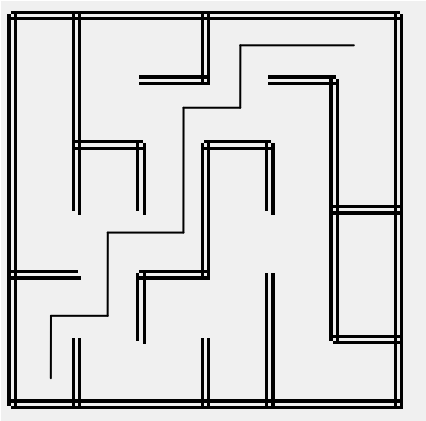


图 3.24 选择坐标为（4,4）的迷宫格为行走方向找到的最短路径

图 3.25 选择其中坐标为(4,2)的迷宫格为所行走的路径。搜索迷宫后的路径(4,3)→(4,2)→(3,2)→(3,1)→(2,1)→(2,0)→(1,0)→(1,1)→(0,1)→(0,0)（起点）。如果选择坐标为(4,2)的迷宫格为假设性的路径，从坐标(4,2)的迷宫格回到起点 S，利用有墙壁的迷宫地图，执行 Flood 算法所找到的路径，这里可以发现，从起点 S 距离终点 G 的距离值为 15，这并不是一条最短路径，原因出在搜索迷宫的时候，所找到的迷宫信息地图并不够完整，没有进行无墙壁的 Flood 算法找最短路径。

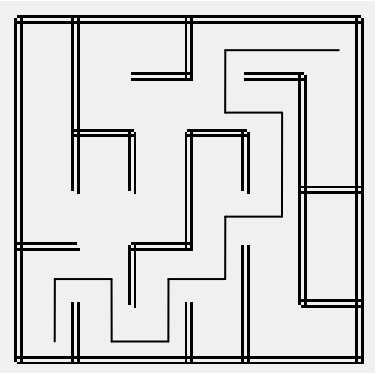


图 3.25 选择坐标为(4,2)的迷宫格为行走方向找到的最短路径

为了确保搜索时迷宫地图的墙壁信息能够完整，利用无墙壁与有墙壁的迷宫地图，各做一次 Flood 算法探索迷宫，比对二条最短路径是否一致。图 3.24 利用没有墙壁的迷宫信息地图，执行 Flood 算法所找到的路径，从起点 S 距离终点 G 的距离值为 11；图 3.25 利用有墙壁的迷宫信息地图，执行 Flood 算法所找到的路径，从起点 S 距离终点 G 的距离值为 15。发现没有墙壁与有墙壁的迷宫信息地图，找到的最短路径并不一致。一旦有墙壁和无墙壁的最短路径不一致时，可以假设迷宫机器人所搜索到的迷宫地图，可能还存在着一条真正地最短路径。

迷宫机器人如果认为所搜索到的迷宫地图，可能还存在一条长度更短的“最短路径”时，可以再次利用中右左法则，再次从起点 S 搜寻到终点 G，重复这样

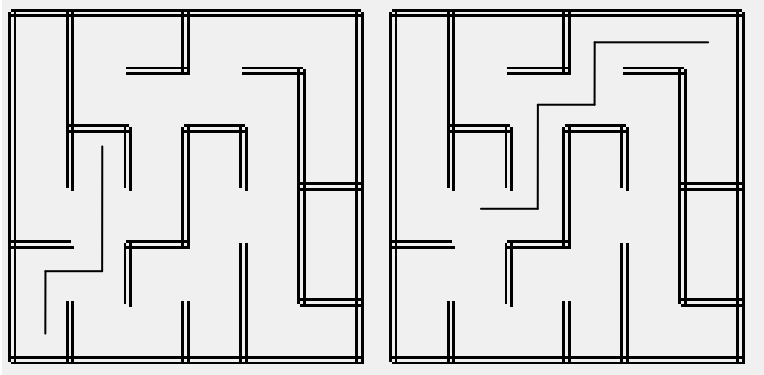
的思想，直到没有墙壁与有墙壁的迷宫信息地图，找到的最短路径一致。首先利用没有墙壁的迷宫地图，执行朝向终点 G 的中右左法则结合 Flood 算法，找到一条建议地最短路径。到达终点 G 后，再次利用中右左法则结合 Flood 算法，从终点 G 搜索到起点 S，搜索尚未走过的迷宫，实现对迷宫墙壁信息的全局探索。运用这种思想，可以发现，找到的最短路径已经是一条最短路径。

没有墙壁的迷宫地图，在没走过的迷宫格都是假设没有墙壁的，可以找到的路径弹性是最高的，即就是说如果有更短路径，由于是没有墙壁的迷宫地图，再次利用 Flood 算法结合中右左法则进行搜索时，会逼近于全局探索，故可能有更短路径也会被找到。

3.6.5 Flood 算法结合向心法则

从上一节介绍的中右左迷宫搜索法则结合 Flood 算法中可以看出，当迷宫机器人走到死路时， 便会利用没有墙壁迷宫信息地图提供一条假设性的路径。既然如此，只要在每个岔路口迷宫格都执行一次 Flood 算法，让 Flood 算法在每个迷宫格，都可以提供一条通往迷宫终点的路径，那么就可以有效率地在每一格迷宫方块都朝着终点行进。结合迷宫和上述算法的特点，论文提出 Flood 算法结合向心法则，利用这一算法搜索迷宫路径。向心法则是一种改良过的深度优先搜索法则，此方法在遇到岔路或是死路时，会辨别目前的绝对位置相对于绝对中心位置的方位，选择向迷宫终点的路口优先进入[34]。

向心法则：遇有交叉时, 以指向迷宫终点的方向为优先的前进方向。利用向心法搜索迷宫后的路径(0,0)(起点)→(0,1)→(1,1)→(1,2)→(1,3)，如图 3.26(a)所示。当迷宫机器人行进到坐标为(1,3)的迷宫格时出现死路，这是迷宫机器人会退回到最近的一个未分支路口坐标为(1,2)的迷宫格，在该迷宫格处再次利用 Flood 算法和向心法则得到一条假设路径(1,2)→(2,2)→(2,3)→(2,4)→(3,4)→(3,5)→(4,5)→(5,5)（终点），如图 3.26(b)所示。如此就可以很快地找到迷宫终点。



(a) (b)

图 3.26 Flood 算法结合向心法则搜索迷宫终点

再让迷宫机器人利用同样思想找回起点，当它到达(5,5)（终点）后的路径为(5,5)→(5,4)→(5,3)。当迷宫机器人运行到坐标为(5,3)的迷宫格时出现死路，这是迷宫机器人会退回到最近的一个的分支路口坐标为(5,5)的迷宫格，即终点处。在该迷宫格处再次利用 Flood 算法和向心法则得到一条假设性路径并返回迷宫起点(5,5)→(4,5)→(3,5)→(3,4)→(4,4)→(4,3)→(4,2)→(3,2)→(3,1)→(2,1)→(2,0)→(1,0)→(1,1)→(0,1)→(0,0)(起点), 如图 3.27 所示。

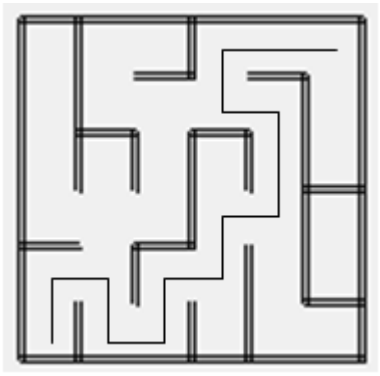


图 3.27 在终点运用向心法则回到起点

使用向心法则，再利用没有墙壁的迷宫信息地图，执行 Flood 算法所找到的最短路径，探索从起点(0,0)到终点(5,5)时的距离值为 11，探索从终点(0,0)到起点(5,5)时的距离值为 15 这是一条最短路径，可见 11 为已知迷宫墙壁信息的最短距离。显然向心法在搜索迷宫终点时所搜寻到的迷宫格较少，而且找到的最短路径也是最短的。

在 16*16 的迷宫当中，由于终点在迷宫中心(7,7),(7,8),(8,7),(8,8)四个迷宫格内，所以将迷宫划分成四个区域，每个区域用不同的法则来完成探索任务。这里规定：迷宫机器人在迷宫右下角时，当其向上运动时，遵循中左法则，当其向右运动时，遵循左手法则，当其向下运动时，遵循右手法则，当其向左运动时，遵

循中右法则；迷宫机器人在迷宫右上角时，当其向上运动时，遵循左手法则，当其向右运动时，遵循右手法则，当其向下运动时，遵循中右法则，当其向左运动时，遵循中左法则；迷宫机器人在迷宫左下角时，当其向上运动时，遵循中右法则，当其向右运动时，遵循中左法则，当其向下运动时，遵循左手法则，当其向左运动时，遵循右手法则；迷宫机器人在迷宫左上角时，当其向上运动时，遵循右手法则，当其向右运动时，遵循中右法则，当其向下运动时，遵循中左法则，当其向左运动时，遵循左手法则。

当迷宫机器人在迷宫的右上角时向心法则代码如下：

```
void centralMethod (void)
{
    if (GmcMouse.cX & 0x08) {
        if (GmcMouse.cY & 0x08) {
/*  此时迷宫机器人在迷宫的右上角  */
            switch (GucMouseDir) {
              case UP:                          /*  当前迷宫机器人向上  */
                leftMethod();                   /*  左手法则            */
                break;
              case RIGHT:                       /*  当前迷宫机器人向右  */
                rightMethod();                  /*  右手法则            */
                break;
              case DOWN:                        /*  当前迷宫机器人向下  */
                frontRightMethod();             /*  中右法则            */
                break;
              case LEFT:                        /*  当前迷宫机器人向左  */
                frontLeftMethod();              /*  中左法则            */
                break;
              default:
                  break;
            }
```

3.7 算法方案比较

深度优先搜索算法是从迷宫的起点出发，顺着某一方向向前探索，若能走通，则继续前进，否则沿原路退回，换一个方向再继续探索，直到所有可能的路径都探索到为止。深度优先搜索算法可以找到迷宫的起点到终点的路径，但是路径不一定是迷宫的最短路径。广度优先搜索是从起点出发，离开起点后依次搜索与当前位置相邻的迷宫单元格，然后分别从这些相邻单元格出发依次访问它们的邻接格，依次类推直到找到迷宫出口，广度算法可以找到迷宫的最优路径，但探索点会随着探索的深入急剧增加，需要大量的内存空间用来保存探索过程的记录[35-37]。Flood 算法结合中右左法则从入口处发，在岔路口按照中右左法则进行迷宫搜索，当遇到死路或已走过的路径时利用 Flood 算法提出一条假设性路径，在假设性路径的搜索上仍然按照中右左法则搜索迷宫，直到找到迷宫终点。Flood 算法结合中右左搜索法没有优先搜索迷宫终点区域的思想，在探索阶段耗费的大量时间。Flood 算法结合向心法则从入口处发，在岔路口优先选取在无墙壁迷宫中接近迷宫终点的方向为前进方向，当遇到死路或已走过的路径时利用 Flood 算法提出一条假设性路径，在假设性路径的搜索上仍然按照向心法则搜索迷宫，直到找到迷宫终点。

为了比较哪一种算法在迷宫路径规划上最有效。论文将迷宫机器人的路径规划进行仿真实验，具体的实验参数设定如下：按照 IEEE 全面电脑鼠走迷宫竞赛规则，迷宫选取为 16*16 的平面坐标系，迷宫的起点为平面坐标系的左下角(0,0)点，迷宫终点为平面坐标系的四个终点坐标(7,7), (7,8), (8,7), (8,8),迷宫机器人到达其中任意一个终点坐标即生成最短路径。迷宫的地图选取 2010 年 IEEE 全面电脑鼠走迷宫竞赛全国赛区地图库。实验的目的是为了通过对比行走迷宫格数、最短路径迷宫格数、转弯次数这三个直接影响运行时间的因素，比较哪种算法更高效。实验结果如表 3.5-3.7 所示，其中不同迷宫地图以阿拉伯数字区分：

表 3.5 不同算法行走迷宫格数比较

地图	深度优先搜索算法行走迷宫格数	广度优先搜索算法行走迷宫格数	FLOOD 算法结合中右左法则行走迷宫格数	FLOOD 算法结合向心法则行走迷宫格数
1	452	386	442	310
2	354	348	318	302
3	406	344	320	294

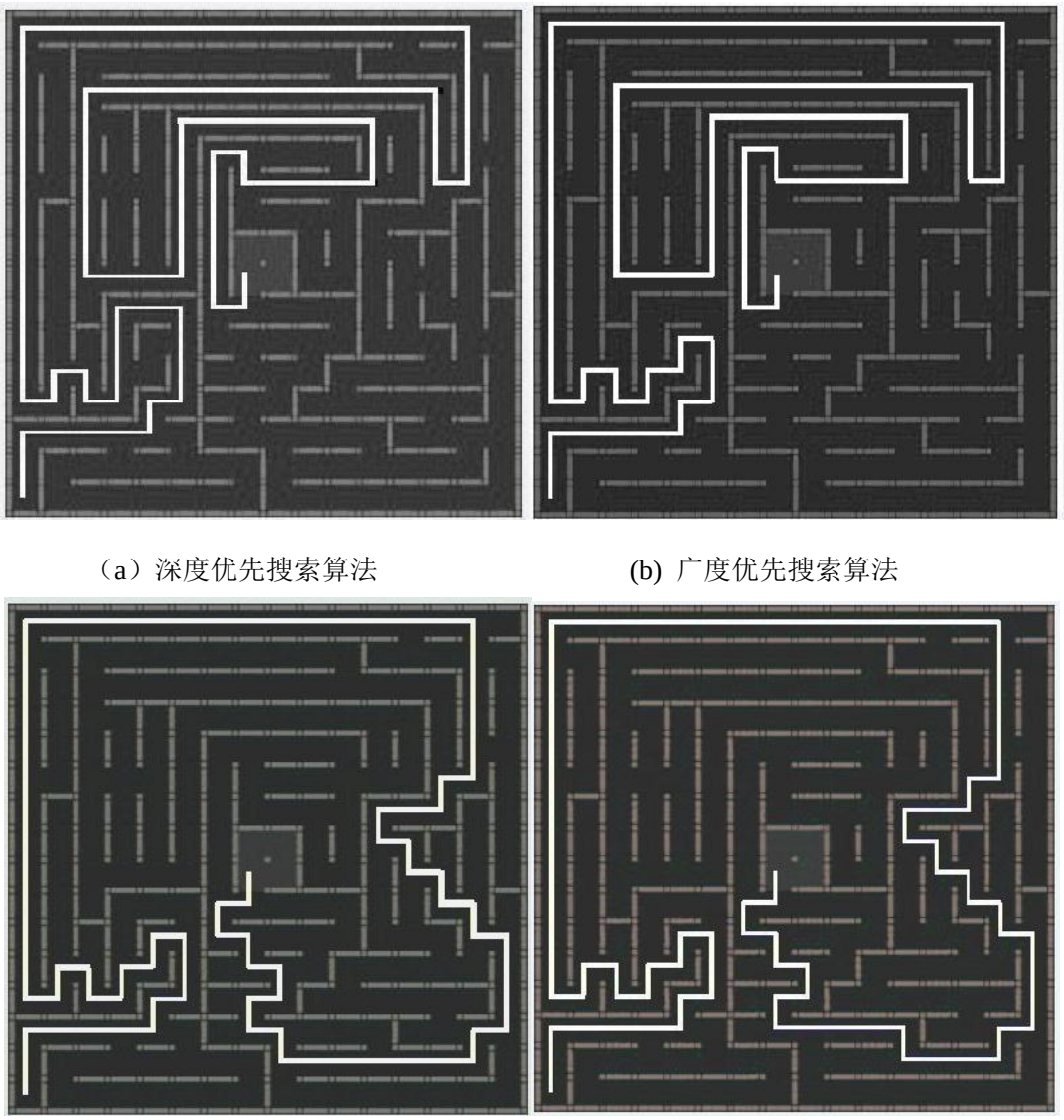
表 3.6 不同算法最短路径迷宫格数比较

地图	深度优先搜索算法最短路径迷宫格数	广度优先搜索算法最短路径迷宫格数	FLOOD 算法结合中右左法则最短路径迷宫格数	FLOOD 算法结合向心法则最短路径迷宫格数
1	102	100	84	84
2	60	60	44	44
3	52	52	52	52

表 3.7 不同算法转弯次数比较

地图	深度优先搜索算法转弯次数	广度优先搜索算法转弯次数	FLOOD 算法结合中右左法则转弯次数	FLOOD 算法结合向心法则转弯次数
1	108	132	98	77
2	83	101	106	55
3	116	133	73	56

其中迷宫地图 1 的不同算法最短路径仿真图如下所示：



（a）深度优先搜索算法　　(b) 广度优先搜索算法

(c) FLOOD 算法结合中右左法则　　(d) FLOOD 算法结合向心法则

图 3.28 不同算法最短路径

经过实验数据和仿真可以看到，深度优先搜索算法行走的迷宫格数相对于其它算法较多，是因为这种算法在遇到岔路口会选择走直线路径，而迷宫本身的设计不会这么简单决定的。广度优先搜索算法转弯次数比其它算法多，是由于在岔路口是依次搜索相邻迷宫格导致的。Flood 算法结合中右左法则没有优先搜索迷宫终点区域的思想，效率也不如 Flood 算法结合向心法则。Flood 算法结合向心法则的效果要优于其它算法，证明论文方案采用这一算法的可行性和优越性。

论文算法相比较上述的其它算法更为有效，其算法思想为在搜索迷宫时采用 Flood 算法结合向心法则，在遇到岔路口时，由记忆功能记忆岔路口的迷宫坐标点，如果遇到死路或已走过的路径，则退回到记忆的上一个迷宫坐标点，直到搜索到迷宫的终点。算法流程图如图 3.29 所示：

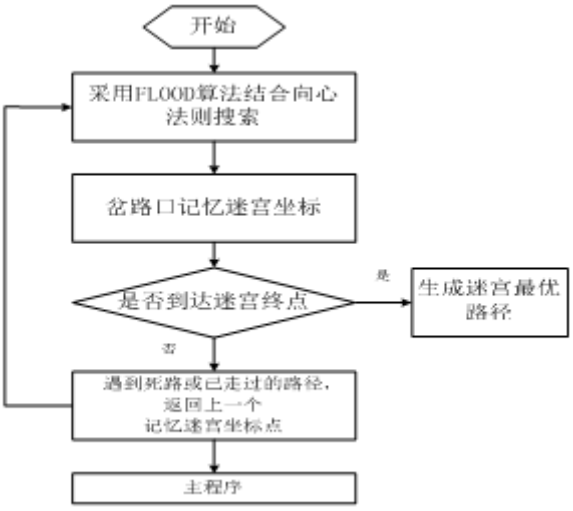


图 3.29 论文选取算法流程图

3.8 小结

本章主要介绍了迷宫机器人记忆功能的实现和优化，详细描述了深度优先搜索算法，广度优先搜索算法，Flood 算法结合中右左法则和 Flood 算法结合向心法则，通过对比仿真得出 Flood 算法结合向心法则的效率最高。

4 迷宫机器人的运动控制

4.1 迷宫机器人行走姿态修正

迷宫机器人能在迷宫中准确的走直线运行，这是迷宫机器人搜索迷宫的第一步。在 16*16 的迷宫中，每个迷宫格至少有一面有墙壁。因此可以让迷宫机器人在迷宫中直线行走时，通过检测四周墙壁的距离来控制行走姿态。

4.1.1 红外传感器布局

为了避免迷宫机器人在行走时碰撞墙壁，需要对其行走姿态实时监控和调整。迷宫机器人在迷宫中的理想姿态是处于迷宫格的中央，且行走方向平行于墙壁，在这种状态下迷宫机器人就不会碰到墙壁，但由于地面摩擦力的变化，步进电机失步等原因，必须要依靠红外传感器检测四周墙壁的距离来控制姿态来防止触碰墙壁。

论文方案的迷宫机器人上共有 5 组红外传感器，如图 4.1 所示。U1-U5 为红外接收头传感器，RF1-RF5 为发射红外线的装置，W1-W5 用来分别调节 RF1-RF5 发射红外线的强度，即用来调节传感器的检测范围。根据迷宫机器人上传感器的安装角度，可以分为两组：U1、U3、U5 为一组；U2、U4 为一组。第一组中它们分别用来检测左、前、右三个方向的墙壁信息；第二组中主要用来修正迷宫机器人的行走姿态。

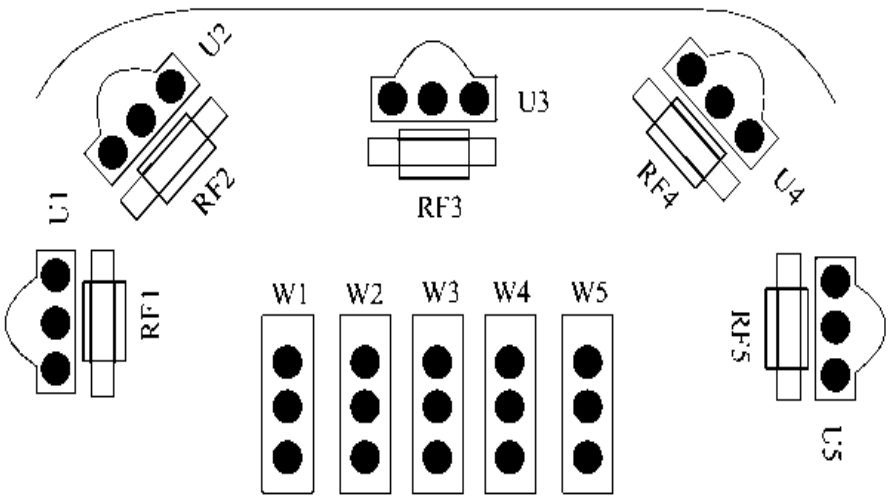


图 4.1 迷宫机器人红外线传感器布局

4.1.2 两侧都存在墙壁的修正原理

如图 4.2 所示，该迷宫机器人正处于正确的行走姿态是不需要进行修正的。图 4.3(a)中表示迷宫机器人在行走时，其前进方向向左倾斜，需要进行姿势调整，否则会撞向左侧墙壁。 图 4.3(b)中表示迷宫机器人在行走时，其前进方向平行靠近右侧，这也需要进行姿势调整，使去走到迷宫格中间去。图 4.3(c)表示迷宫机器人行走时前进方向倾斜偏右，需要调整姿势，否则就会撞向右侧墙壁。图 4.3(d)表示迷宫机器人行走时平行靠近右侧，这时也应该调整，使迷宫机器人走到迷宫格中间去。

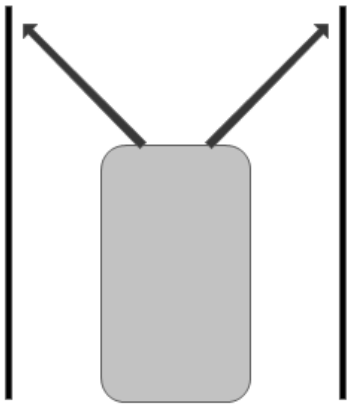
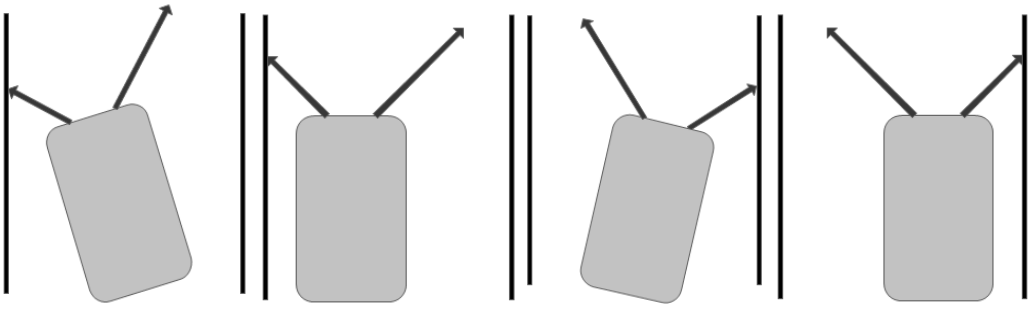


图 4.2 正确姿态



（a） (b) (c) (d)

图 4.3 行走姿态斜左（a）偏左（b）斜右（c）偏右（d）示意图

迷宫机器人在迷宫格左侧发生偏移的校正方法，与迷宫机器人在右侧发生偏移的校正方法与之相同，下面介绍如何使用红外传感器来检测位置偏差。图 4.3 中，有一段带有箭头的线段，这是红外接收传感器 U2、U4 检测的示意。该线段的长度表示传感器探测的距离。从图 4.3 中可以看出，传感器检测到的左右墙壁的距离长度不同，这是就需要对迷宫机器人进行姿态丢在。当迷宫机器人靠近右侧挡板时，修正方法相同。

一体式红外线接收传感器是迷宫机器人获取迷宫方式的唯一方式。迷宫机器人通过一体式红外线接收传感器获取迷宫墙壁信息，构建迷宫地图。这里给出一个一体式红外线接收传感器 IRM8601S 在正常环境下测量与墙壁不同距离实际值与测量值的数据。如表 4.1 所示：

表 4.1 红外线测量值与实际值

实际值	10cm	20cm	30cm
测量值	10.8cm	20.1cm	29.7
	10.3cm	19.5cm	29.9cm
	9.5cm	19.8cm	30.5cm
测量平均值	10.2cm	19.8cm	30.0cm

一个迷宫格的大小为 18cm*18cm。由此可知，论文选用的一体式红外线接收传感器测量值是比较准确的，可以用实际值调整迷宫机器人姿态。

4.1.3 只有一侧存在墙壁的修正原理

如图 4.4 所示，假设只有左侧墙壁存在。当迷宫机器人靠近左侧墙壁行走时，其修正方法与上面类似。

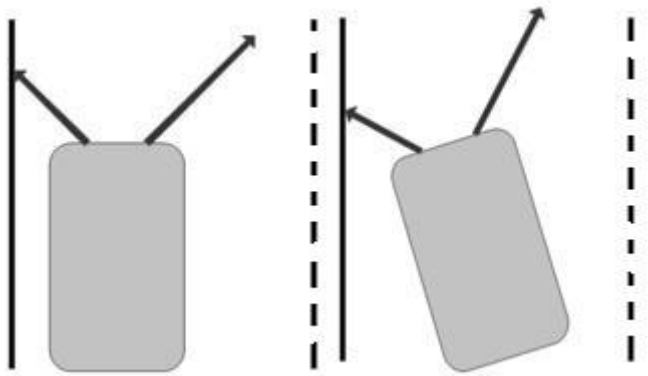


图 4.4 只存在左侧墙壁的偏左（a）斜左（b）示意图

当迷宫机器人在左侧是，由于没有右侧墙壁，会出现如图 4.4 中的情况。这就要使用其正左方的 U1 传感器。当检测到左侧墙壁，并且距离墙壁太近时，迷宫机器人就应该向右调整，达到修正姿势的目的。迷宫格中只有右边有墙壁时，它的修正情况与之类似，此时只用到 U5 红外传感器。

4.1.4 当前方存在墙壁时的修正方法

如图 4.5 所示，当迷宫机器人处于该位置时，U2、U4 传感器用于左右修正姿态，其探测到的是前方的墙壁，这是就会给姿态修正带来误判。因此当前方出现

墙壁时，不处理 U2、U4 传感器的检测结果。

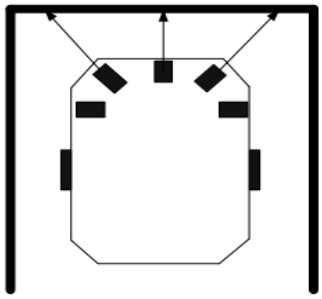


图 4.5 前方存在墙壁

迷宫机器人应当利用处于正上方的 U3 红外传感器来分别检测前方是否存在墙壁，如图 4.6 所示。

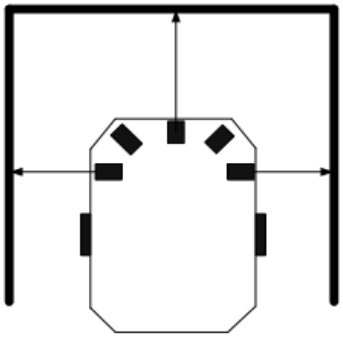


图 4.6 正上方墙壁检测

利用 matlab/simulink 设计一个误差校正仿真，其中包含了直走速度命令控制器，误差产生器。移动轨迹模块计算迷宫机器人的运动轨迹。通过仿真来迷宫机器人运动时产生的误差校正。如图 4.7 所示，*x* 坐标轴为迷宫机器人运动方向，*y* 坐标轴为校正方向。

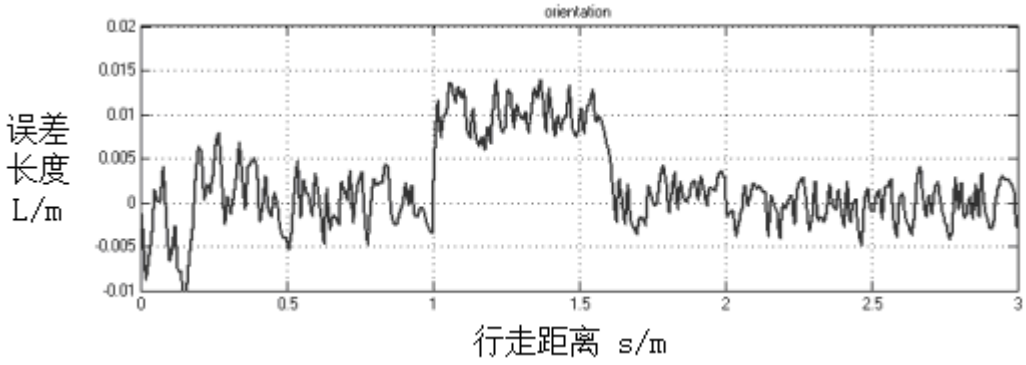


图 4.7 迷宫机器人误差校正输出波形

4.1.5 程序代码设计

通过上面的分析可知，红外传感器 U1、U5 不但要检测是否存在墙壁，还要

在部分情况下检测迷宫机器人是否偏离太远；红外传感器 U3 只需要检测正前方是否存在墙壁；U2、U4 也只是用来检测迷宫机器人行走时的姿态是否正常。

根据具体需要要把各个传感器检测到的结果用一个变量保存。为了管理方便，将 5 个传感器的状态统一保存在一个数组里。GucDistance[0]- GucDistance[4]依次代表红外传感器 U1-U5 的状态，C 语言代码如下：

```
Static unit8 GucDistance[5]   =0          /* 记录传感器状态 */
```

为了方便程序设计，需要对五个红外传感器宏定义，代码如下：

```
#define LEFT       0          /* 左方传感器      */
#define FRONTL     1          /* 左前方传感器    */
#define FRONT      2          /* 前方传感器      */
#define FRONTR     3          /*右前方传感器     */
#define RIGHT      4          /*右方传感器       */
```

变量 Gucdistance[]保存传感器状态的数据含义参见表 4.2 所示。

表 4.2 变量 Gucdistance[]说明

变量名	值	含义
Gucdistance[LEFT]	0x00	左边没有墙壁
	0x01	左边存在墙壁，但是距离较远
	0x03	左边存在墙壁
	其它	无意义
Gucdistance[FRONTL]	0x00	左前方未检测到墙壁
	0x01	左前方检测到距离墙壁太近
	其它	无意义
Gucdistance[FRONT]	0x00	前方无墙壁
	0x01	前方存在墙壁
	其它	无意义
Gucdistance[FRONTR]	0x00	右前方未检测到墙壁
	0x01	右前方检测到距离墙壁太近
	其它	无意义
Gucdistance[RIGHT]	0x00	右方没有墙壁
	0x01	右方存在墙壁，但是距离较远
	0x03	右方存在墙壁

当迷宫机器人前进时，要根据传感器实时检测到的墙壁信息，及时的调整自身姿态。如果需要迷宫机器人向右靠近时，就需要控制左边电机比右边电机转动速度快；反之要使右边电机比左边电机转动速度快。

分别调节两侧电机的速度完成姿态修正。但为了方便，本论文通过暂停功能控制相应一侧电机少走一步来进行姿态修正。这里给出了驱动右电机的定时器 0 中断服务函数程序，代码如下：

```
/*  电机驱动程序  */
break;
default:
break;
 }                              /*  是否完成任务判断 */
if (__GmRight.cState != __MOTORSTOP)
 {
__GmRight.uiPulseCtr++;      /*  运行脉冲计数       */
__speedContrR();             /*  速度调节           */
 }
if (__GmRight.uiPulseCtr >= __GmRight.uiPulse) {
__GmRight.cState = __MOTORSTOP;
__GmRight.uiPulseCtr = 0;
__GmRight.uiPulse = 0;
__GmRight.iSpeed = 0;
}
```

4.2 电机控制

迷宫机器人在转弯，行走已经记忆的迷宫以及最后的冲刺阶段都需要对电机进行加速和减速控制。为了使其在加速和减速时减少轮胎的打滑、晃动，防止电机的震荡与失步，一种有效地解决方案是对电机进行匀加减速的控制，论文的迷宫机器人使用的是步进电机。步进电机不能像直流电机那样自动加减速，它的加减速需要通过设定节拍的频率来实现[38-40]。假设每步的平均速度等于中心时间的速度。

电机某一步的持续时间（即定时器的延时时间）为：

$$T_n = \frac{C_n}{f},\quad V_n = \frac{f}{C_n} \qquad \text{式（4-1）}$$

其中C_n为第n步定时器的计数值，f为给定时器提供的时钟源频率。转速为V_n（单位：步/秒）。

加速度和减速度β（单位：步/秒^2）为：

$$\beta = \frac{V_{n+1}-V_n}{\frac{T_{n+1}}{2}+\frac{T_n}{2}} = \frac{\frac{f}{C_{n+1}}-\frac{f}{C_n}}{\frac{C_{n+1}}{2f}+\frac{C_n}{2f}} = \frac{2f^2(C_n-C_{n+1})}{C_nC_{n+1}(C_{n+1}+C_n)} \qquad \text{式（4-2）}$$

式（4-2）假设了每步的平均速度等于中心时间的速度。如果为加速状态，则C_n大于C_{n-1}，β大于 0；如果为减速状态，则C_n小于C_{n-1}，β小于 0。电机转过的步数n为：

$$n = \int_0^t v(t)dt = \int_0^t \beta t dt = \frac{\beta}{2}t^2 \qquad \text{式（4-3）}$$

转过n步所花费的时间t_n为：

$$t_n = \sqrt{\frac{2n}{\beta}} \qquad \text{式（4-4）}$$

第n步的定时器计数值C_n为：

$$C_n = f(t_{n+1}-t_n) = f(\sqrt{\frac{2(n+1)}{\beta}}-\sqrt{\frac{2n}{\beta}}) \qquad \text{式（4-5）}$$

由上面两式可得第 0 步的计数值C_0为：

$$C_0 = f\sqrt{\frac{2}{\beta}} \qquad \text{式（4-6）}$$

再由上面三个公式，可得第n步的定时器计数值C_n也可以表示为：

$$C_n = C_0(\sqrt{n+1}-\sqrt{n}) \qquad \text{式（4-7）}$$

由公式（4-6）可知，第 0 步的计数值C_0包含了加速度（或减速度）β的信息，只要把C_0设定好，后面的实时计算就不用再考虑β了。相邻两步比值为：

$$\frac{C_n}{C_{n-1}} = \frac{C_0(\sqrt{n+1}-\sqrt{n})}{C_0(\sqrt{n}-\sqrt{n-1})} = \frac{\sqrt{1+\frac{1}{n}}-1}{1-\sqrt{1-\frac{1}{n}}} \qquad \text{式（4-8）}$$

根据泰勒变换可得：

$$\sqrt{1\pm\frac{1}{n}}=1\pm\frac{1}{2n}+\frac{1}{8n^2}+o[\frac{1}{n^3}] \qquad \text{式（4-9）}$$

把公式（4-9）代入公式（4-8）可近似得到：

$$\frac{C_n}{C_{n-1}}\approx\frac{(1+\frac{1}{2n}-\frac{1}{8n^2})-1}{1-(1-\frac{1}{2n}-\frac{1}{8n^2})}=\frac{\frac{1}{2n}-\frac{1}{8n^2}}{\frac{1}{2n}+\frac{1}{8n^2}}=\frac{4n-1}{4n+1}=1+\frac{2}{4n+1} \qquad \text{式（4-10）}$$

由上式可得到递推公式：

$$C_n=C_{n-1}-\frac{2C_{n-1}}{4n+1} \qquad \text{式（4-11）}$$

$$C_i=C_{i-1}-\frac{2C_{i-1}}{4i+1} \qquad \text{式（4-12）}$$

公式（4-12）将公式（4−11）的 n 换成 i. 减速过程中要转动 m 步，可得

$$C_i=C_{i-1}+\frac{2C_{i-1}}{(4m-1)-4i} \qquad \text{式（4-13）}$$

现在根据公式（4-8）和公式（4-10）分别列出相邻两步C_n值之比的确切值，如表 4.3 所示。这里假设C_0等于 1。

表 4.3 近似值的误差

n	精确的 C_n/C_{n-1}	近似的 C_n/C_{n1}	近似值误差
1	0.4142	0.6000	0.4485
2	0.7673	0.7778	0.0136
3	0.8430	0.8462	0.00370
4	0.8810	0.8824	0.00152
5	0.9041	0.9048	7.66×10^{-4}
10	0.9511	0.9512	9.42×10^{-5}
100	0.9950	0.9950	9.38×10^{-8}
1000	0.9995	0.9995	9.37×10^{-11}

从上表可以看出，除C_1/C_0外，其它比值的误差都很小。由于每一步的延时

时间是由前一步计算出来的，如果C_1的误差较大，后面每一步的延时误差都增大，解决办法是把C_1单独挑出来，使它等于 0.4142 倍C_0，其它值依然按照公式（4-11）递推计算，如公式（4-14）所示：

$$C_1 = 0.4142 * C_0 \qquad \text{式（4-14）}$$

最后，由公式（4-4）可推导出步进电机从速度为 0 加速到 s, 以及速度从 s 减速到 0 所要走过的步为:

$$n = \frac{s^2}{2\beta} \qquad \text{式（4-15）}$$

在论文中，加速度和减速度都为 1000 步/秒^2，给定时器提供的时钟源频率为 50M, 根据式（4-6）和式（4-14）可以计算出C_0和C_1的值，根据式（4-11）就可以推导出C_n的值。

加减速代码设计：

```
uint n;
GuiAccelTable[0] = 2236068;        /*计算第一步定时器计数值                  */
c0=f*sqrt(2/w)
GuiAccelTable[1] = 926179;         /*第二步定时器计数值   c1=0.4142*c0 */
for(n = 2; n < 500; n++)
{
GuiAccelTable[n] = GuiAccelTable[n - 1] - (2 * GuiAccelTable[n - 1] / (4 * n + 1));
}
```

4.3 连续转弯

迷宫机器人在迷宫中行走时遇到转弯路口需要转弯，现在的转弯方式主要有以下两种：一种的传统的原地转弯，这种转弯是车体先在转弯路口停下，然后一个车轮反转，另一个车轮正传。还有一种方式是在转弯路口不停及前进中转弯，

这种转弯方式需要一侧电机慢转，另一侧电机快转。前进中转弯节约时间，效率高，而原地转弯控制较为简单。由于原地转弯需要先减速停下来，转完后再加速，非常浪费时间，故选取连续转弯方法。R_1表示内侧轮胎划出的轨道半径，R_2表示外侧轮胎划出的轨道半径。当需要转弯时，外侧轮保持当前速度不变，而内测轮的速度降至外侧轮的$K = R_1 / R_2$倍，迷宫机器人便开始转弯，直到走过一定步数或内侧传感器检测到有墙时，此次转弯便结束。但是由于左右轮的摩擦力和惯性不同，实际的 K 值不等于R_1 / R_2。

4.3.1 关于摩擦力的介绍

当一个物体在另一个物体的表面上滑动时，有存在于两接触面间一种阻止物体运动的作用力就是摩擦力[41]，其摩擦力的方向永远与运动方向相反。图 4.7 摩擦力的种类有以下三种：1.静摩擦力：若两物体相互接触且相互挤压，而又相对静止，在外力作用下如只具有相对滑动趋势，而又未发生相对滑动，则它们接触面之间出现的阻碍发生相对滑动的力，谓之“静摩擦力”。在物体没有被拉动前，拉力均与摩擦力相等，因此摩擦力并非定值。2.最大静摩擦力：施拉力于物体，当力增加到使物体恰可开始运动时，物体间存在的摩擦力，称为最大静摩擦力，且在物体环境固定的情况下最大静摩擦力为一定值。3.滑动摩擦力：物体开始运动后，摩擦力仍然存在，此时的摩擦力称为滑动摩擦力。

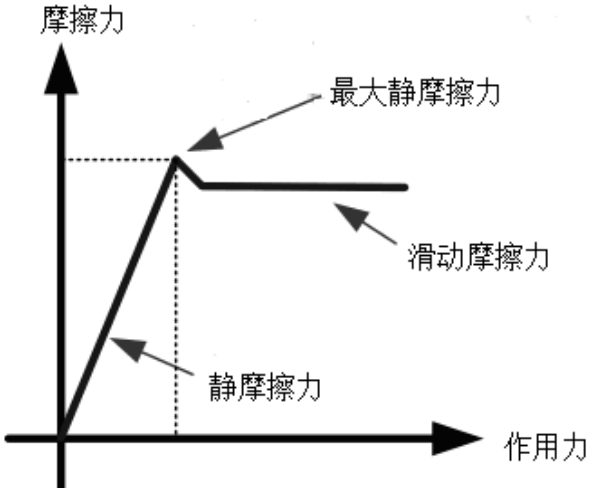


图 4.8 摩擦力示意图

摩擦力是一个相当复杂的物理现象。现在用一个简化的情形来看，假设介于干燥的二个平面之间且没有润滑的介质，列奥纳多·达·芬奇发现：1、摩擦力正比于与单位面积的接触力。2、而且与接触面积大小无关。3、摩擦力与相对运动速度大小无关。

从微观的角度可以更好地理解这一点。同样一个木块，接触面积较大的那一面，每单位面积所接触的力就会变小，因为接触点的力变小的关系，它的接触点的面积就会比较少，它的摩擦力就会变小，但是接触地面的总面积比较大；假如把它竖起来，单位面积所受的力就会变大，接触点的面积就会加大，它的摩擦力就会变大，但是接触地面的总面积比较小。所以当乘上与地面接触的总面积，无论接触面的面积，摩擦力的大小是相同的。

4.3.2 轮胎的选择

论文通过实验选择适合的轮胎胎皮，这里利用一块木板，将迷宫机器人横放在木板上，迷宫机器人的质量为m，重心加速度为g，木板的倾角为θ，如果迷宫机器人静止在平面上，它的加速度为零，根据牛顿第二运动定律，此时的所有作用力相加等于零，反作用力N为：

$$N = mg\cos\theta \qquad \text{式（4-16）}$$

只要迷宫机器人仍是静止，木板的倾角θ增加时，静摩擦力的大小也增加，因为$\sin\theta$随木板的倾角θ的增加而增加；最后当木板的倾角够大时，迷宫机器人接近于滑动状态，这时最大静摩擦力$fs_{\max}$为

$$fs_{\max} = mg\sin\theta \qquad \text{式（4-17）}$$

这时候的摩擦力系数μ为：

$$\mu \equiv fs_{\max} = mg\sin\theta = \tan\theta mg\cos\theta \qquad \text{式（4-18）}$$

论文方案选用的轮胎胎皮，摩擦系数μ约为0.7，所以迷宫机器人的加速度a不可以超过：

$$a = \mu * g = 0.7 * 9.8m/s^2 = 6.68m/s^2 \qquad \text{式（4-19）}$$

否则迷宫机器人的轮胎对于地面，会因为摩擦力不足，而产生滑动的现象。

迷宫机器人在转弯时实际是以R为半径，当迷宫机器人的左右轮设定不一样的速度时，速度与半径R的实际测量值关系如表4.4所示：

表4.4 左右轮速度与半径关系图

速度 / 半径	左轮：0 右轮：5cm/s	左轮：5cm/s 右轮：10cm/s	左轮：10cm/s 右轮：15cm/s
实际值	25.6cm	46.8cm	61.4cm

通过迷宫机器人左右轮速度和半径之间的关系，可以分析出其在转弯时的运动特性，对分析和设计转弯时的运动轨迹有很大帮助。

4.3.3 连续转弯程序设计

```
void mouseTurnright(void)
{
    while (__GmLeft.cState    != __MOTORSTOP);
    while (__GmRight.cState != __MOTORSTOP);
    /* 开始右转*/
   __GucMouseState     = __TURNRIGHT;
    __GmRight.cDir       = __MOTORGOBACK;
    __GmRight.uiPulse = 41;
    __GmLeft.cDir         = __MOTORGOAHEAD;
    __GmLeft.uiPulse    = 61;
    __GmRight.cState    = __MOTORRUN;
    __GmLeft.cState     = __MOTORRUN;
    GucMouseDir         = (GucMouseDir + 1) % 4;
    while (__GmLeft.cState    != __MOTORSTOP);
    while (__GmRight.cState != __MOTORSTOP);
    __mazeInfDebug();
    __delay(100000);
}
```

4.4 直线冲刺

利用等加速度运动方程式计算加速的距离，加速度为 a，初速度在时间 t_1 时为 v_1，末速度在时间 t_2 时为 v_2，时间的变化量为 $\Delta t = t_1 - t_2$，加速度计算为 $a = (v_2 - v_1)/\Delta t$，也可以写成 $\Delta t = (v_2 - v_1)/a$ [42]，那麼加速的距离就是

$$d = \frac{v_2 + v_1}{2} * \Delta t = \frac{v_2 + v_1}{2} * \frac{v_2 - v_1}{a} = \frac{v_2^2 - v_1^2}{2a} \qquad \text{式（4-20）}$$

在做直线冲刺时，需要考虑加速距离与减速距离，(4-20)式只有计算加速的距离，如果加速度与减速度相同，迷宫方块一格固定为 18cm，所以加减速的距离

都是$d=9*n$，n 为可以加减速的迷宫方格，因为迷宫的方块都是固定 18cm，一般的直走冲刺末速度v_2就是

$$v_2=\sqrt{2*9*n*a+v_1^2},n=1,2,3....14 \qquad \text{式（4-21）}$$

迷宫机器人直线冲刺时，通过记忆下的直线迷宫格数算出加速距离减速距离和末速度。

算法流程图如图 4. 9 所示：

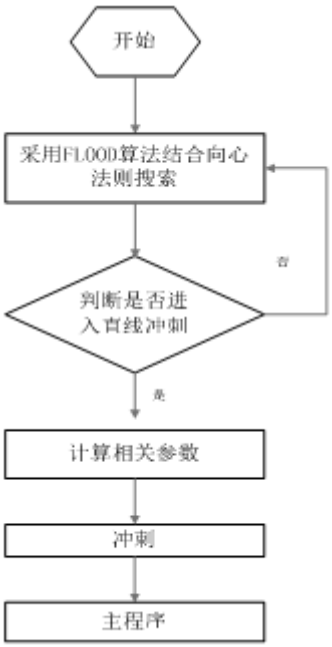


图 4.9 直线冲刺算法流程图

4.5 小结

本章主要从矫正行走姿态，连续转弯，直线冲刺等方面进行论述，解决了迷宫机器人行走迷宫时准确和高效的问题

5 总结与展望

随着世界各国对智能移动机器人的深入研究和在生产生活中的普遍应用，智能移动机器人这种包含了数学、电子学、信息技术、物理学、生物学等多学科的技术产品也将越来越受到重视。

论文以迷宫机器人路径规划算法研究为中心。首先搭建的迷宫机器人的硬件平台，主要通过仿真对比四种路径规划算法确定了路径规划策略方案。由于仿真和现实之间存在误差，最后对运动控制进行优化，使其能在实际的迷宫搜索中更为准确有效的完成任务。本论文主要完成了以下工作：

（1）硬件方案设计。通过阅读大量资料确定了迷宫机器人各个模块的选型，完成了各模块的电路设计，应用 Protel DXP 绘制系统原理图，完成了迷宫机器人的硬件设计。

（2）算法方案设计。根据 IEEE 全面全面电脑鼠走迷宫的规则，比对其它算法，并在此基础上提出算法的优化结合，确定了迷宫机器人的路径规划算法。确保了高效的完成迷宫搜索任务。

（3）运动控制方案设计。考虑到在实际运行中的种种问题，提出了对迷宫机器人的行走姿态修正和电机在加减速时的控制，使其能准确、稳定的行走。最后对与迷宫搜索的优化和直线冲刺，保证了在更短的时间内完成迷宫搜索。

论文虽然取得了一些成绩，但也存在不足之处：

（1）首先是硬件壁垒，在控制模块上没有加入陀螺仪等先进的技术产品，使得迷宫机器人在运动控制上未能更为精确。

（2）迷宫机器人未能实现远程监督控制。在算法优化，运动控制的实验中不能及时的反馈数据，使得查找问题和解决问题投入了大量的时间，造成了很多的不便。

迷宫机器人的相关技术都是相关学科的前沿技术，需要研究者有很高的理论知识和实践能力。只有运用理论知识到实践中去才用验证其是否是真正可行的，是否更为高效，才用研究出真正有用的正果。

致　谢

本论文得以完成，首先感谢指导老师段中兴教授、郑普亮老师，在作者研究的期间给予耐心指导与鼓励，在此致上最诚挚的谢意。也感谢在研究生学习期间，受到每一位老师于课堂上的教导与关怀。由衷的感谢我的父母亲及家人无时无刻给予我温暖与关怀，因为有父母多年来的辛劳养育，我才能的顺利的取得硕士学位。感谢段中兴老师给予我参加 IEEE 电脑鼠走迷宫竞赛的机会和经费上的帮助，让我所参加的比赛可以顺利完成并取得三等奖。最后，感谢我所待的实验室，在这里除了学习我的专业技能以外，还有学习待人处事的道理。感谢学长们，提供诸多的援助，在此向他们表达特别感激之意。感谢刘浩洋、叶之祺同学在迷宫机器人的硬件设计、软件编写上给予的帮助。感谢和我朝夕相处的同学们，使我日常生活多了许多欢笑，使得研究生活增添许多美好及愉快的回忆，谢谢你们陪我渡过这段岁月。祝愿大家身体健康，工作顺利，祝愿人工智能与机器人研究的明天更美好。

参考文献

[1] N J Nilsson.A mobile automation:An Application of Artificial Intelligence Techniques[M]. Proc.of UCAI,reprinted in Autonomo Mobile Robots:Control,Planning and ArchiteOnre. 1969,2(1):233—239.

[2] Borenstein,Y Koran.Real—time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments[M].Robotics and Automation,1990.Proceedings.1990 IEEE International Conference on,13-18 May 1990,10:572-577.

[3] Koren Y Botenstein J.Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation[M].Robotics and Automation,199lProceedings.199l IEEE International Conference on,9-ll Apr1991.2:1398—1404.

[4] 李磊，叶涛，谭民，陈细军．移动机器人技术研究现状与未来[J].机器人，2002 年 9 月，24 卷第 5 期：475.

[5] 机器人发展简史[J].世界科学.2007,03.

[6] 袁牧．基于路径规划的迷宫移动机器人[J].中国测量技术，2008,06.

[7] 张钹．智能移动机器人的现状及发展[J].科学导报.1992,6:42-46.

[8] 孟伟，洪炳，韩学东．一种多机器人协作控制方法[J].机器人，2004,12.

[9] 曲道奎，杜振军，徐殿国，徐方．移动机器人路径规划方法研究[J].机器人，2008,02.

[10] 贾润亮．多机器人系统路径规划系统研究[D].太原：太原理工大学，2008.

[11] 朱力．目前各国机器人发展情况[J].China youth s&t.2003,11:38-39.

[12] 朱德良．迷宫机器人路径规划算法研究[D].上海：东华大学，2007.

[13] 戴博，肖晓明，蔡自兴．控制工程[J].移动机器人路径规划技术的研究现状与展望 2005.

[14] 周立功等．IEEE 电脑鼠开发指南-基于 MicroMouse615 迷宫智能鼠[M].广州致远电子有线公司.2008,1.

[15] LM3S615 微控制器数据手册[M].2007,03:15-17.

[16] 实验教程 EasyARM615 [J].广州致远电子有线公司.2008.

[17] Pere. Automatic Planning of Manipulator Movements.IEEE TRANS on Sys Man and Cyb[M].1981,11(11):681-698.

[18] 西原主计等[日]，牛连强、赵文珍译．机器人 C 语言机电一体化接口[M].北京：科学出

版社，2002.

[19] 调制[EB/OL]http://baike.baidu.com/view/265301.htm.

[20] 微控制器数据手册 LM3S615 [M].2007,03:31-40.

[21] 齿轮技术资料 KHK Kohara Gear Industy, [EB/OL].
http://www.khkgears.co.jp/tw/gear_technology/guide_info.html.

[22] 印刷电路板布局指导原则[EB/OL].
http://www.haifeng.idv.tw/leo/cgi-bin/attachment.cgi?forum=6&topic=426&postno=2&name=emilayoutnctu_1145007001&type=.doc/

[23] 蒋新松．水下机器人[M].辽宁：辽宁科学技术出版社，2002.

[24] Holland J H.Genetic Algorithms and the Optimal Allocations of Trails SIAM Journalof Computing[D].1993,2:82-105.

[25] IEEE 国际电工和电子工程学会电脑鼠走迷宫竞赛规则[M].2011,3.

[26] 祝尚臻，于宏涛，李丽霞．迷宫机器人的回溯深度优先算法应用[J].科技信息，2009,06.

[27] 孙巧榆，潘荫荣，胡幼华，孙强．计算机工程[J].八方向走迷宫算法，2004,1.

[28] 贾明，严世贤．Linux 下的 C 编程[M].北京：人民邮电出版社，2001.

[29] 谭浩强．C语言程序设计（第3版）[M].北京：清华大学出版社，2005：91-111.

[30] 严蔚敏，吴伟民．数据结构(c语言版)[M].北京：清华大学出版社，1996:167—169.

[31] 严蔚敏，吴伟民．数据结构(c语言版)[M].北京：清华大学出版社，1996:169—170.

[32] 周文辉，唐健，谢培泰．计算机与网络[J].OSPF中的flood算法改进，1999,12.

[33] 王小忠，孟正大．机器人运动规划方法的研究[J].控制工程，2004,03.

[34] 贺少波，孙克辉．计算机系统应用[J].基于向心法则的电脑鼠走迷宫算法设计与优化，2012,9.

[35] 周立功．ARM 嵌入式系统基础教程[M].北京：北京航空航天大学出版社，2005.

[36] 刘保柱，苏彦华，张宏林．MATLAB7.0从入门到精通[M].北京：人民邮电出版社，2010：266-372.

[37] 刘旭．智能移动机器人路径规划及仿真[D].南京：南京理工大学，2004.

[38] 林俊，谷兵，杨晨，蔡婷婷．计算机应用研究[J].自适应泛洪的迷宫路径优化算法研究，2012,12.

[39] 薛定宇，陈阳泉．基于MATLAB/Simulink的系统仿真技术与应用（第2版）[M].北京：清华大学出版社，2011：1-412.

[40] Margrit Betke,Leonid Gurvits．Mobile Robot Localization Using Landmarks[M].IEEE Trans.on

Robotics and Automation,1997,13(2):247-275.

[41] 上海交通大学物理教研室．大学物理学（第4版）[M].上海：上海交通大学出版社，2011：75-77.

[42] 上海交通大学物理教研室．大学物理学（第4版）[M].上海：上海交通大学出版社，2011：90-92.

附录 硕士研究生学习阶段发表论文

1.刘世泽.基于人工神经网络的迷宫路径策略问题研究，工业控制计算机，拟 2013 年第六期