摘　要

本文以第七届全国大学生机器人大赛为背景，介绍基于多传感器的机器人电气控制智能系统设计及实现的相关内容。本论文首先在引言中对自主移动机器人的发展以及课题的背景意义做了简单的介绍，后面章节分别从总体方案确定、硬件电路设计、软件程序编写以及整体效果等方面进行了介绍。其中着重介绍了本文作者在本次大赛中负责和参与的几项技术攻关任务——基于多传感器的坐标更新的研究、底层路径控制算法的设计和实现、主脑层的自主路径规划的实现，以及基于计算机视觉的避障行为和白块目标检测和定位。需要指出的是，上述研究都是基于机器人能通过各种传感器感知机器人内部和外界信息而实现的。

多传感器定位方面，利用陀螺仪和码盘的信息实时计算机器人的坐标的同时，利用场地上白线的信息加了光纤传感器帮助进行坐标纠偏，使机器人在真实对抗中，坐标误差一般在 30mm 以内。

机器人底层移动路径跟踪方面，把整条路径分解成直线和圆弧的组合，圆弧路径的跟踪使用了圆弧半径和行进角度的闭环控制，直线路径的跟踪使用了位置和角度的闭环控制。并在终点精确到达的控制中加入了闭环控制。

路径规划方面，本文作者基于启发式深度搜索和曲线拟合的技术研发了专用于本次比赛的路径算法，它实现了各个自动机器人路径的自主规划，后又加入了机器人的障碍检测并重新规划无障碍路径，达到了一定的避障效果。

视觉方面，本文作者用自己实践中发现的稳定的色彩空间变换方法并结合目标尺寸信息在复杂的背景中正确地定位了机器人的目标——白块，并用类似的方法对场地中的白线进行计数，辅助机器人定位；另外，用 canny 算子提取边缘，定位图象边缘较多的复杂区域，对障碍进行检测定位。

本文可在一定程度上和某些特定方面作为机器人电气控制方法和实现的参考，也可以作为以后机器人大赛的技术积累。

关键词：自主移动机器人，路径规划及避障，计算机视觉，多传感器信息融合，轨迹跟踪

Research on the Control System Based on Multi-sensor for Double take-off and Landing Flipping Autonomous Mobile Robots

Abstract

Under the background of the seventh National robot competition for college students, this article introducs the design and realization of the intelligent control system for robots which is based on multi-sensor.

This article first gives a brief introduction of the autonomous mobile robot development and significance of the background. From the back section,it refers to the overall program,hardware circuit design,software program and the overall effect. Specially,the paper focuses on several technology research tasks: location updates based on multi-sensors; the trajectory control algorithm design and implementation; the path planning algorithm as well as obstacle avoidance based target detection; and the target detection based on computer vision.What is needed to point out is that all the research is based on the interaction of the robot with environment by different sensors.

Based on multi-sensors includeing gyroscopes, odometer and fiber sensors,we developed a new method to get the robot's accurate location whose error is usually below 30mm in the competition.

Based on PID control theory, the whole path is divided into straight line and arc. Radius and angle PID loop control is used to make robot move in arc,and position and angle PID loop control is used to make robot move in straight line.And loop control is used for the robot's accurate arrival.

Based on heuristic depth-first search and curve fitting technology，the author developed a path planing algorithm exclusively for this competition.It greatly reduces the debugging of the preparation of a fixed line workload, another advantage is to save storage fixed path of the enormous memory consumption. the author joined the robot obstacle detection and barrier-free path planning to achieve a certain degree of obstacle avoidance effect.

Based on computer vision technology, the author found an effective color space transform method to detect the target in a complex background, and to count the white line to help the robot to realize the second localization;in addition, inspired by the depth of field methods to detect obstacles, by using canny operator, the author regards those complex regions that have more edges as the location of obstacles, thus we realize the obstacles detection.

This article can be provided as the referrence on some specific aspects to some extent,may also as technique data for future robot contest.

Key Words：Autonomous Mobile Robots ,Path Planning and Barrier Avoidance, Computer Vision, Multi-sensor information fusion, Path Tracing

目　　录

摘　　要 1

Abstract 2

引　　言 1

1 绪论 2

1.1 对自主移动机器人及其体系结构研究的发展 2

1.2 自主移动机器人几种典型的体系结构 3

1.3 自主移动机器人中的研究热点 6

1.4 机器人与自动化的关系 9

1.5 本章小结 10

2 课题背景 11

2.1 全国机器人大赛 11

2.1.1 全国机器人大赛的发展 11

2.1.2 我校参赛情况 11

2.1.3 双升降翻转型自主移动机器人主体机械结构 12

2.2 课题背景及开展研究的意义 14

2.2.1 机器人竞赛的出现和发展 14

2.2.2 研究机器人竞赛的意义 14

2.3 本文主要的课题内容 14

2.3.1 机器人各自主功能模块 15

2.3.2 机器人硬件体系 16

2.3.3 机器人软件体系 17

2.4 本章小结 19

3 自主移动机器人硬件系统设计 20

3.1 硬件系统总体结构 20
3.2 机器人系统核心处理器-ARM7 处理器 21
3.2.1 ARM7--LPC213x 系列微处理器 21
3.2.2 LPC213x 系列微处理器开发环境介绍 23
3.3 电源模块硬件设计 24
3.3.1 电池选择 24
3.3.2 电源模块电路设计 26
3.4 主控模块硬件设计 26
3.4.1 主控模块功能分析 26
3.4.2 RS-485 总线研究 26
3.4.3 主控模块电路 28
3.5 导航模块硬件设计 29
3.5.1 导航模块总体结构 29
3.5.2 SPI 串行通信 29
3.5.3 陀螺角度采集 31
3.5.4 码盘距离采集 33
3.5.5 光纤传感器 35
3.6 视觉模块 35
3.6.1 华硕 EPC 笔记本 35
3.6.2 摄像头和镜头的选择 36
3.6.3 视觉模块和主控的通信 36
3.7 本章小结 37
4 基于多传感器的自主移动机器人全场定位算法及实现 38
4.1 机器人定位理论和方法 38

4.2 比赛机器人的全场定位方法 39
4.2.1 比赛机器人的全场定位的传感器数据采集 39
4.2.2 比赛机器人的全场定位的算法实现 40
4.3 多传感器融合的全场定位算法的研究 42
4.3.1 陀螺仪和里程计定位的特点及其局限 42
4.3.2 光纤沿白线定位的特点及其局限 43
4.3.3 陀螺仪里程计结合光纤定位的特点及优势 43
4.3.4 单一依据的辅助定位算法的原理与实现 44
4.3.5 三角更新算法的原理与实现 46
4.4 本章小结 50
5 自主移动机器人轨迹跟踪控制算法及实现 51
5.1 移动机器人的控制问题综述 51
5.2 机器人轨迹跟踪算法 54
5.2.1 经典 PID 算法 54
5.2.2 直线跟踪算法 55
5.2.3 圆弧跟踪算法 57
5.2.4 终点准确定位算法 59
5.3 机器人轨迹跟踪实际问题及解决办法 59
5.4 机器人轨迹跟踪控制的实验效果 60
5.5 本章小结 60
6 自主移动机器人主脑层路径规划及避障算法设计与实现 61
6.1 自主移动机器人路径规划概述 61
6.1.1 自动机器人路径自主规划的意义 61
6.1.2 常用的路径规划方法 61

6.2 自动机器人避障处理.....63
6.2.1 自动机器人避障的意义.....63
6.2.2 障碍点的检测.....63
6.2.3 障碍点的描述.....64
6.3 本届大赛自动机器人路径规划及避障的设计.....64
6.3.1 本届大赛场地具体分析和描述.....65
6.3.2 算法的思路与设计.....67
6.3.3 算法关键函数与流程图.....72
6.3.4 算法在单片机的移植和修改.....75
6.4 自主移动机器人路径规划的实际应用.....76
6.4.1 机器人重试时的路径规划.....76
6.4.2 机器人遇障时的路径规划.....78
6.5 本章小结.....78
7 基于计算机视觉的自主移动机器人目标跟踪和二次定位.....79
7.1 计算机视觉简介.....79
7.1.1 Marr 理论简介.....79
7.1.2 计算机视觉应用系统的组成.....79
7.1.3 计算机视觉应用系统的应用领域.....80
7.1.4 开放源代码的视觉函数库 OpenCV 简介.....80
7.2 计算机视觉在参赛机器人上的应用.....81
7.2.1 利用计算机视觉搜索并定位目标.....81
7.2.2 利用计算机视觉探障.....86
7.2.3 利用计算机视觉辅助二次定位.....88
7.3 计算机视觉程序架构和平台说明.....89

7.3.1 多线程的程序架构 89
7.3.2 单线程的程序 90
7.3.3 linux 平台下的程序实现 91
7.4 本章小结 91
总结和展望 92
参 考 文 献 95
附 录 A 英文资料原文 97
附 录 B 英文资料翻译 108
附 录 C 自动机器人照片 117
附 录 D 自主移动机器人部分程序代码 118
在 学 取 得 成 果 122
致 谢 123

引　言

竞赛机器人作为机器人技术普及与发展的一个重要门类，竞赛机器人的发展已经经历了 20 余年，至今已经有几十种公认的机器人赛事在国际上举办，范围遍及全世界。我国在竞赛机器人的研究方面也已取得了一定的成绩[1]。

2008 年 6 月，中央电视台将继续主办第七届的全国大学生机器人电视大赛，以“竞技牛郎”为主题。本论文主要研究讨论的是一整套基于本届大赛规则的自动机器人电气控制系统。在很多方面我们仍然落后于许多兄弟院校，这就要求我们第七届的队员在总结师兄经验与教训的同时，虚心学习优秀院校的先进技术，全力走进尚未开发的领域进行探索，力求能通过第七届比赛把我校机器人大赛的电控技术进行一次全面的提升，迈上一个新的台阶。

本文将从机器人及其技术的国内外研究现状入手，针对本次电视机器人大赛，详细论述自动机器人的单片机控制系统及其实现，尤其着重论述本文作者在本次大赛中负责和参与的若干项技术——基于多传感器信息全场定位、主脑层自主路径规划、轨迹跟踪算法及基于视觉的辅助定位目标、探障避障行为的实现。

论文的各个部分组织如下：

1） 第一章绪论，简述自主移动机器人技术的国内外发展情况和前沿热点。

2） 第二章介绍本课题背景与意义，简述本机器人系统的软硬件设计框架。

3） 第三章论述自动机器人整体硬件设计和选型。

4） 第四章论述自动机器人利用多传感器信息融合进行全场定位的方法与实现。

5） 第五章论述的自动机器人轨迹跟踪和终点停靠闭环控制的方法与实现。

6） 第六章论述自动机器人主脑层路径规划及和避障的方法与实现。

7） 第七章论述自动机器人基于视觉的避障、定位白块目标以及辅助定位的方法与实现。

1 绪论

1.1 对自主移动机器人及其体系结构研究的发展

移动机器人的研究始于 20 世纪 60 年代末期，斯坦福研究院的 Nils Nilssen 和 Charles Rosen 等人，在 1966 年至 1972 年中研制出了取名 Shakey 的自主移动机器人[1]，目的是研究应用人工智能技术，在复杂环境下系统的自主推理、规划和控制。通常认为，自主式移动机器人 AMR（Autonomous Mobile Robot）是具有高度的自规划、自组织、自适应能力，适合于在复杂的环境中工作的一种智能机器人，具有模型不确定性、系统的高度非线性和控制的复杂性[2] [3]。

随着智能机器人研究水平的不断深入和提高，各种各样的新型传感器被采用，信息融合、全局规划，运动学和动力学计算等模块的技术水平也不断提高，使机器人整体智能能力不断增强，同时也使系统结构变得复杂。在这个结构中，功能如何分解，时间关系如何确定，空间资源如何分配等问题，都直接影响整个系统智能能力。同时为了保证智能系统的扩展，技术的更新和各种新算法的采用，要求系统的结构具有一定的开放性，从而保证机器人智能能力不断增强，采用高新技术的传感器和各种先进的算法，可以实现机器人功能上的任意添加，这使得机器人的体系结构本身变成一个需要研究和解决的复杂问题[4]。

移动机器人体系结构以研究移动机器人系统结构中各模块之间的相互关系和功能分配为对象，即研究多 CPU 系统的功能划分和各层的逻辑结构，以求确定一个或多个智能机器人系统的智能结构和逻辑关系。对于一个具体的移动机器人而言，体系结构可以说就是这个机器人信息处理和控制系统的总体结构。李佳宁等人对移动机器人的体系结构定义为：移动机器人的体系结构是指如何把感知、建模、规划、决策、行动等多种模块有机地结合起来，从而在动态环境中，完成目标任务的一个或多个机器人的结构框架。体系结构是整个机器人系统的基础，它决定着系统的整体行为和整体性能，体系结构设计的合理与否是整个机器人系统高效运行的关键[5]。

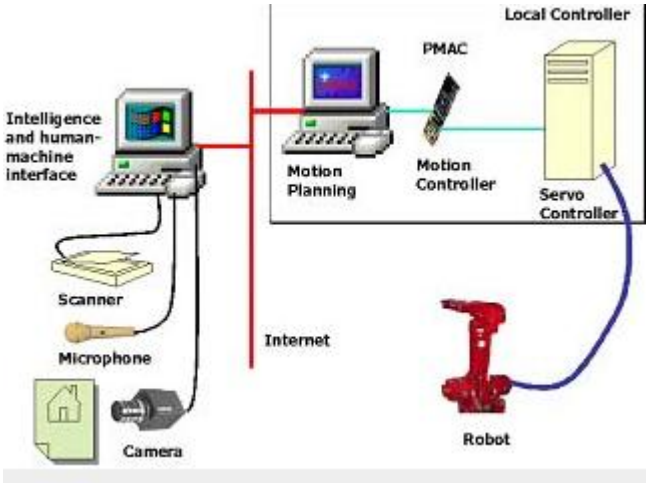


图 1.1 机器人控制系统

1.2 自主移动机器人几种典型的体系结构

从某种程度上来说，自主移动机器人的核心问题是机器人的体系结构的问题[5]。近年来，许多学者致力于解决体系结构中的各种问题，并使结构思想具有一定的普遍指导意义，其中最典型的两种体系结构体系是分层式体系结构和包容式体系结构。

1） 分层式体系结构（Hierarchical Architecture）

分层式体系结构是基于认知的人工智能（Artificial Intelligence， AI）模型，因此也称之为基于知识的体系结构。在 AI 模型中，智能任务由运行于模型之上的推理过程来实现，它强调带有环境模型的中央规划器，它是机器人智能不可缺少的组成部分，而且该模型必须准确、一致。分层式体系结构是把各种模块分成若干层次，使不同层次上的模块具有不同的工作性能和操作方式。

分层式体系结构中最有代表性的是 20 世纪 80 年代智能控制领域著名学者 Saridis 提出的三层模型。Saridis 认为随着控制精度的增加，智能能力减弱，即层次向上智能增加，但是精度降低，层次向下则相反。按照这一原则，他把整个结构按功能分为三个层次，即执行级、协调级和组织级。其中，组织级是系统的“头脑”，它以人工智能实现在任务组织中的认知、表达、规划和决策；协调级是上层和下层的智能接口，它以人工智能和运筹学实现对下一层的协调，确定执行的序列和条件；执行级是以控制理论为理论基础，实现高精度的控制要求，执行确定的运动。需要指出的是，这仅仅是一个概念模型，实际的物理结构可多于或少于三级，无论多少级，从功能上来说由上到下一般均可

分为这三个层次。信息流程是从低层传感器开始，经过内外状态的形势评估、归纳，逐层向上，且在高层进行总体决策；高层接受总体任务，根据信息系统提供的信息进行规划，确定总体策略，形成宏观命令，再经协调级的规划设计，形成若干子命令和工作序列，分配给各个控制器加以执行。

在分层式体系结构中，最广泛遵循的原则是依据时间和功能来划分体系结构中的层次和模块。其中，最有代表性的是美国航天航空局（NASA）和美国国家标准局（NBS）提出的 NASREM 的结构。其出发点之一是考虑到一个智能机器人可能有作业手、通讯、声纳等多个被控分系统，而这样的机器人可能组成一个组或组合到更高级的系统中，相互协调工作；出发点之二是考虑已有的单元技术和正在研究的技术可以应用到这一系统中来，包括现代控制技术和人工智能技术等。整个系统横向上分成信息处理、环境建模和任务分解三列，纵向上分为坐标变换与伺服控制、动力学计算、基本运动、单体任务、成组任务和总任务六层，所有模块共享一个全局数据库，如图 1.2 所示。

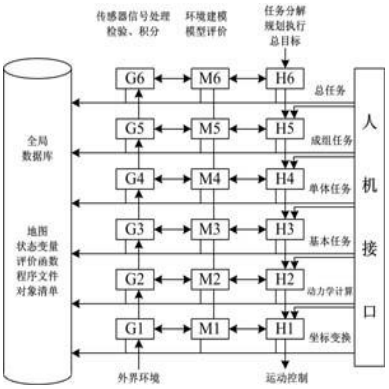


图 1.2 NASREM 的分层体系结构图

NASREM 结构的各模块功能和关系非常清楚，有利于系统的构成和各模块内算法的添加和更换。它具有全局规划和推理的能力，对复杂的环境可以做出合理的反应，适合于一个或一组机器人的控制。但同其它的分层式体系结构一样，NASREM 的问题在于输入环境的信息必须通过信息处理列的所有模块。结果往往是将简单问题复杂化，影响了机器人对环境变化的响应速度，而对机器人非常重要的一点就是对环境变化、意外事件的发生等要求作出迅速反应。

2） 包容式体系结构（Subsumption Architecture）

分层式结构能够较好地解决智能和控制精度的关系，创造一种良好的自主式控制方式。然而由于环境模型的误差、传感器的误差、环境的不确定性、机器人控制系统的复杂性等，使得分层式体系结构在灵活性、实时性和适应性方面经常存有缺陷。

针对上述缺点，美国麻省理工学院的 R.Brooks 从研究移动机器人控制系统结构的角度出发，提出了基于行为的体系结构—包容式体系结构（Subsumption Architecture）。与分层式体系结构把系统分解成功能模块，并按感知—规划—行动（Sense-Planning-Action，SPA）过程进行构造的串行结构不同（如图 1.3 所示）；包容式体系结构是一种完全的反应式体系结构，是基于感知与行为（Sense-Action，SA）之间映射关系的并行结构（如图 1.4 所示）。在包容式结构中，上层行为包含了所有的下层行为，上层只有在下层的辅助下才能完成自己的任务；另一方面下层并不依赖于上层，虽然上层有时可以利用或制约下层，然而下层的内部控制与上层无关，增减上层不会影响下层。

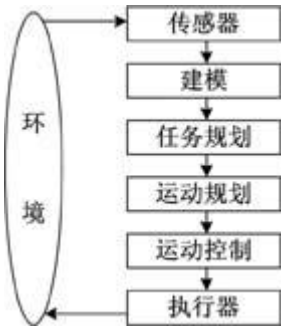


图 1.3 按功能划分的串行结构

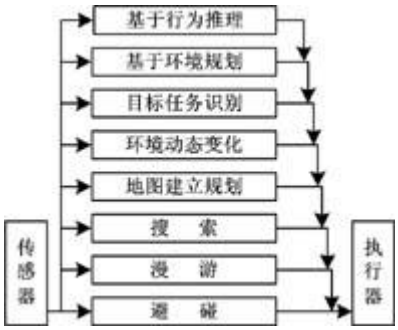


图 1.4 按行为划分的并行结构

包容式体系结构被成功应用于机器人的导航避障。Herbert 机器人实现室内环境中寻找返回苏打罐；清华大学计算机系自主开发的 THMR-Ⅲ型机器人的控制系统，都是包容式体系结构应用的典型例子。

包容式体系结构强调模块的独立、平行工作，但缺乏全局性的指导和协调，虽然在局部行动上可显示出灵活的反应能力和鲁棒性，但是对于长远的全局性目标跟踪显得缺少主动性，目的性较差。

3） 混合式体系结构

随着移动机器人研究的不断深入，要求移动机器人完成的任务越来越复杂和精确，运动的实时性和跟随性也要求越来越高，且对多机器人协作提出更高要求。由于单纯的基于知识或者基于行为结构的机器人已无法满足机器人发展和应用的要求，结合两者优点，国内外学者纷纷提出基于自己的混合式体系结构方案。混合式体系结构成为了机器人体系结构研究的热点。

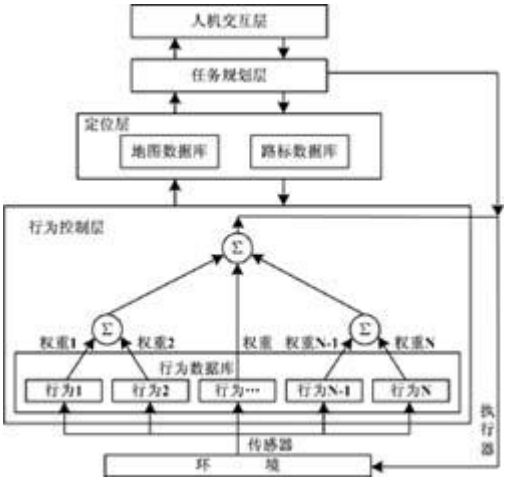


图 1.5 CASIA-Ⅰ的混合分层控制体系结构图

Gat 提出了一种混合式的三层体系结构，分别是：反应式的反馈控制（Controller），反应式的规划—执行层（Sequencer）和规划层（Deliberator）。中科院自动化研究所设计的室内移动机器人 CASIA-Ⅰ采用基于行为的混合分层控制体系结构，该结构包括人机交互层、任务规划层、定位层和行为控制层四个层次，如图 1.5 所示。

1.3 自主移动机器人中的研究热点

近年来，由于机器人相关学科的发展和人们对机器人体系结构研究的重视，体系结构的研究成果卓著，其研究主要集中在以下几方面：

1） Agent 技术在体系结构中的应用

从广义上理解，智能体（Agent）的概念涵盖了许多不同的计算实体，这些实体能够感知环境并作用于环境，它可以是物理实体，也可以是软件代码，还可以与其它智能体建立通讯，系统协调这些智能体的行为，以求共同的动作和问题求解。由于智能体与移

动机器人内部功能模块性质相似，并且基于智能体的方法具有自治性、适应性、稳健性，且易于实现，这对那些结构性不好或者定义不明确的任务有很大优势，所以学者们纷纷将多智能体系统（MAS）理论引入移动机器人系统的体系结构设计，提出了分布式体系结构，并取得了很多成绩。

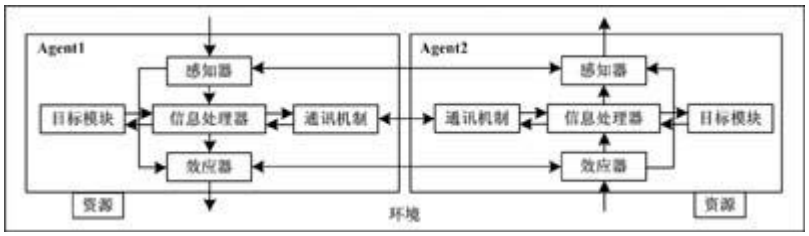


图 1.6 智能体基本结构图

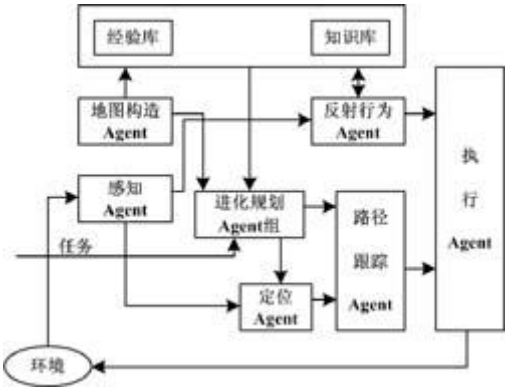


图 1.7.基于 Agent 的移动机器人进化控制体系结构图

夏幼明等人认为智能体是一个既具有信息处理能力的主动实体，又具有与外界交互的感知器、通讯机制，对信息进行存储加工的信息处理器、记忆库；同时还具有根据共同目标和自己的职责所产生的目标模块以及反作用于外部环境的效应器 ，其结构如图 1.6 所示。陈志华等人在分析移动机器人系统工作原理的基础上，构建了一种基于多智能体的投篮机器人（SBMR）体系结构，并对智能体的功能，智能体的工作过程及其特征等问题进行了讨论。中南工业大学将进化控制技术和 Agent 技术相结合，提出了基于 Agent 的移动机器人进化控制体系结构。该系统兼顾了行为的智能性和反应的快速性，且具有开放性的特点，便于维护和改进，其体系结构如图 1.7 所示。王海等人提出了一种基于行为的分布式多智能体结构，在此基础上设计并建立了一个多自主移动机器人系统实验平台，允许系统中的各移动机器人分布式通讯、规划和控制。

2） 多移动机器人系统的体系结构研究

随着机器人应用领域的不断扩展，多机器人系统已经引起了学者们的普遍重视，而构造多移动机器人系统一个重要的因素是体系结构设计，因为系统性能的优劣很大程度上取决于结构是否合理。个体机器人的体系结构是多移动机器人系统的基本组成单位，因此，研究个体机器人的体系结构是研究多移动机器人很重要的研究方向。

李智军等人面向分布式 Agent 的多移动机器人系统，根据机器人系统任务级的协作实现过程和移动机器人应当具备的能力，结合基于知识和基于行为的控制体系结构的特点，设计了一种分层式移动机器人个体控制体系结构，如图 1.8 所示，其控制体系结构包括状态观测层、决策规划层、协调控制层和行为控制层四个层次。

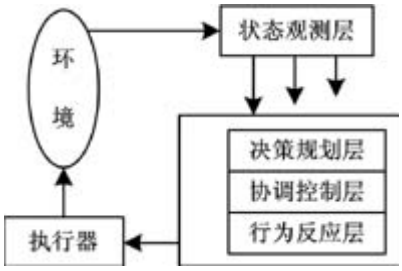


图 1.8 分层式移动机器人个体控制体系结构图

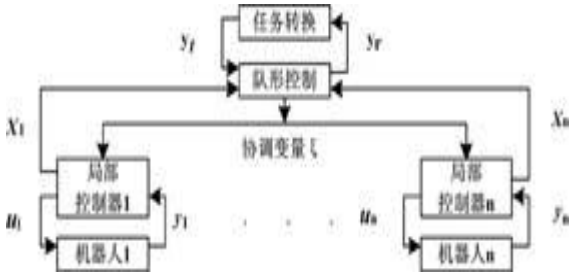


图 1.9 一种新的队形控制体系结构图

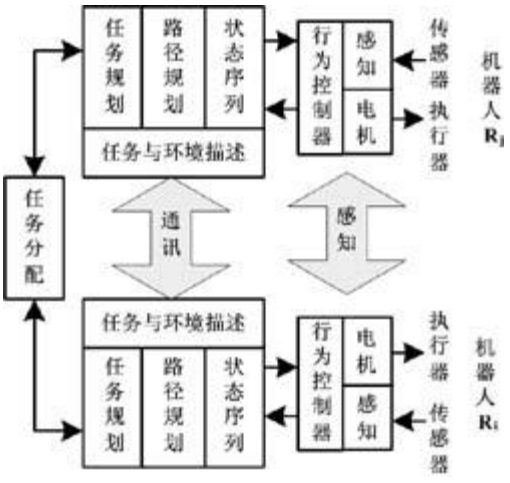


图 1.10 多机器人协同作战系统的结构图

将多机器人系统作为一个整体考虑，多机器人系统的体系结构可以分为集中式（Centralized）和分散式（Decentralized），而分散式结构进一步分为分层式（Hierarchical）和分布式（Distributed）两种[5]。多机器人体系结构就是研究如何根据任

务类型，机器人个体能力等确定机器人群体模型以及恰当的相互关系。Parker. L.E.指出以往对多机器人系统协作的研究大部分只关注效率，而忽视了机器人的故障容忍和适应能力。针对这一问题，他提出了一种名为 LALLIANCE 的新体系结构。该结构将任务、导向行为和选择机制应用到基于行为的体系结构中，不但提高了机器人团队协作效率，而且还满足了实际应用中机器人系统故障容忍和适应能力的要求[5]。苏治宝等人从队形控制基本思想和各种方法的优缺点等方面论述了目前研究队形控制的三种方法：跟随领航者法（leader-following），基于行为法（behavior-based），虚拟结构法（virtual structure），并介绍了一种弥补各种方法缺点的新的队形控制体系结构，如图 1.9 所示。董慧颖等人则将多机器人系统应用在军事上，针对战场上外部环境复杂，系统需对周围环境进行及时准确的判断的特点，采用了任务协调层和行为控制层的混合协调结构作为多机器人协同作战系统的结构，其结构如图 1.10 所示。

3） 基于 CAN 总线移动机器人系统设计研究

在移动机器人体系结构的硬件实现方面，由于 CAN 总线采用了许多新技术和独特设计，与一般通信总线相比，其数据通信具有突出的可靠性，实时性和灵活性等特点。它是一种有效支持分布式控制和实时控制的通讯网络，用它来实现机器人控制结构比较理想。

1.4 机器人与自动化的关系

机器人集成了运动学与动力学、机械设计与制造、计算机硬件与软件、控制与传感器、模式识别与人工智能等学科领域的先进理论与技术。同时，它又是一类典型的自动化机器，是专用自动机器、数控机器的延伸与发展。当前，社会需求和技术进步都对机器人向智能化发展提出了新的要求。

机器人是多学科交叉的产物，但随着机器人应用环境和任务的复杂化，在非结构复杂环境下的信息综合与处理、针对复杂任务的规划和协调的难度和影响变得突出，需要采用信息反馈、优化控制、协调集成的理论、方法与技术去解决，控制学科在系统优化和综合集成方面的优势，将越来越在智能机器人中发挥主导作用。而智能机器人作为一种自动化系统，无论在理论与技术的覆盖面与前沿性、与各种先进信息技术的结合以及物理实现的多样性方面都是其它任何一类自动化系统所不能比拟的。因此，机器人在自

动化科学技术中的代表性和地位将随着其应用范围的拓宽、所采用信息技术的更新和智能程度的提高，得到进一步的认可。

自动化专业是一个由综合性、交叉性的学科所构成的专业，除了控制领域的专业知识，同时融合了传感器、信息、计算机和通信专业的技术。自动化专业从硬件领域来看，包括 EDA 设计，集成电路设计，各种驱动电路设计，芯片设计，单片机应用等，从软件领域看，信息系统包括控制理论，模式识别，人工智能，神经网络等。机器人的设计，制作与研究过程，既包括了控制理论和人工智能的应用又包括了信息的处理与传送。因此，可以说机器人是自动化领域开展各种研究非常合适的一个平台。

图 1.11 自主移动机器人--“勇气号“火星车

1.5 本章小结

本章介绍当前自主移动机器人的最新发展方向和研究热点，以及和自动化专业的紧密联系。

2 课题背景

2.1 全国机器人大赛

2.1.1 全国机器人大赛的发展

我国也曾经多次举办过机器人足球赛等机器人比赛，但是从未举办过类似于 NHK 国际机器人比赛，需要参赛者完成机器人机械结构以及硬件和软件的全部设计制作的赛事。中央电视台举办的首届全国大学生机器人电视大赛是国内首次举行的这类比赛。冠军队将有机会参加亚洲广播电视联合会主办的亚太地区机器人比赛。大赛的宗旨是充分体现“科技是第一生产力”思想，为我国大学生提供一个充分展示其创造力的舞台，进一步激发广大青少年的科技创造热情，为培养大批具有创新精神的高科技人才奠定基础。全国大学生机器人电视大赛已举行了六届：第一届的主题是“抢攀珠穆朗玛峰”，第二届的主题是“太空征服者”，第三届的主题是“鹊桥相会”，第四届的主题是“登长城，点圣火”，第五届的主题是“修建双字高塔”， 第六届的主题是“华夏之光--黄帝建造指南车”。

2.1.2 我校参赛情况

2002 年首届比赛全国共有 27 所大学的 32 支参赛队参加。我校代表队设计制作的机器人代表北京科技大学参加了这项赛事，最终进入了 4 强。在 2003 年第二届的比赛中，我校代表队取得了全国冠军的好成绩。2004 年的第三届比赛中我校代表队也取得了亚军的好成绩。通过比赛，我们得到了很多经验，但跟其他一些学校相比，我们还是有一定的差距的，我们的水平有待继续提高，特别是涉及到高新科技的技术，像机器人视觉，机器人智能控制等等。2005 年的第四届比赛主题是“登长城，点圣火”，我校代表队不负众望，又一次夺得了全国冠军，并在国际比赛中获亚军。2006 年的第五届和 2007 年的第六届大赛中我校代表队再次披荆斩棘，连续两次勇夺全国亚军。

今年，中央电视台又将举办以“竞技牛郎”为主题的第七届全国大学生机器人电视大赛，冠军队将参加 2008 年亚太大学生机器人大赛。

本论文就是以第七届全国大学生机器人电视大赛为背景，主要研究一套基于本届大赛的自动机器人智能系统。

2.1.3 双升降翻转型自主移动机器人主体机械结构

双升降翻转型自动机器人在赛场上有以下几个主要任务：

1） 与另一自动机器人——桥车配合取黄油块；

2） 与手动机器人配合取黄油块；

3） 取白色奶酪。

4） 依靠视觉信息定位掉到地上的白色奶酪块并用爪子夹取。

因为场地中间的黄油块价值 12 分，是比赛双方争夺的中心，经过策略讨论，确定了双提升翻转车的设计思路是要做一个高度灵活的智能小车，所以要能够准确得将块夹起，提升到指定的位置，并能够稳定快速的翻转到指定的位置。将翻转车分为四个部分，即行走机构，翻转机构，提升机构和夹持机构。图 2.1 是双提升翻转车的整体照片。

1） 提升机构设计与制作。

备选方案主要以下几种：

（1） 汽缸提升。优点是提升速度快，结构简单，但对管路的气密性要求很高，不具有重复性，不能利用重试机会。

（2） 利用蓄能，如弹簧。优点是可以采用机械触发，结构简单，但完成动作时，常常比较猛烈，容易造成比赛设施的损毁。

（3） 利用电机加线轮。此种方案的形式有很多种，比较灵活，靠电机带动绕线轮提升至指定高度，在比赛过程中根据需要升降，充分利用重试机会。但提升速度较慢，提升过程平稳。

（4） 综合方式。利用电机将块束缚在出发位置，同时弹簧蓄力，电机在给电后释放约束后靠弹簧提升。

根据比赛规则，第一和第二种方案被否决了，因为规则限定比赛上场之前弹簧不能蓄能，汽缸也不能提前蓄汽。因此采用了第三种方案，即采用电机加线轮方式设计了提升倍增机构，即二级提升，二级提升能把提升机构的速度相对提升电机来说快一倍。

2） 翻转机构设计与制作。

翻转机构是以舵机作为动力源，跟舵机相连接的是一根轴，夹子与轴相连接，当舵机转动时，夹持机构跟着转动。将翻转机构简化可得如图 2.1 所示，建立一个理想力学模型。

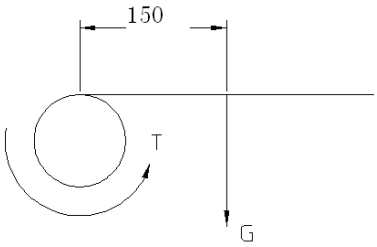


图 2.1 翻转机构受力图

图 2.1 中，G 是夹取机构的重力加上黄油的重力 G=G1+G2=6.6+1.4=8N（数值恰当放大）,T 是舵机输出的力矩，150mm 是总的重力到轴的轴距为 L。

则总的重力对轴的力矩是：

$$T1=G\times L=8\times 15=120N/cm \qquad (2.1)$$

舵机正常输出力矩 T=180N/cm，故 T>T1。舵机提供的动力能正常驱动翻转机构。

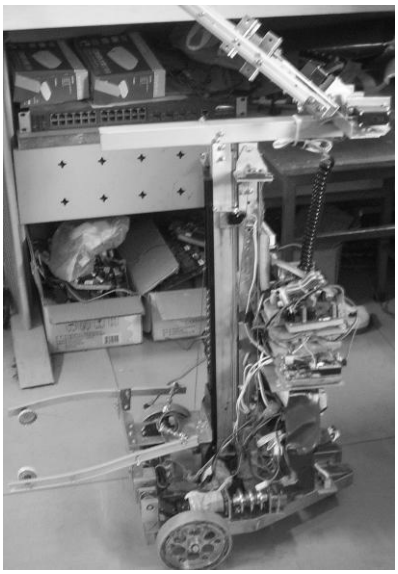


图 2.2 双提升翻转机器人整车照片

2.2 课题背景及开展研究的意义

2.2.1 机器人竞赛的出现和发展

在世界科技和经济的发展的情况下，为了普及机器人知识，促进机器人技术的研究发展，世界上很多国家和地区纷纷开展了各种类型的机器人比赛。比如国际机器人足球联合会（FIRA）组织的微机器人世界杯 Microsoft，国际人工智能协会组织的机器人世界杯 RobotCup，NHK 国际机器人比赛，机器人相扑比赛等等。机器人比赛除了需要采用比较先进的技术外，还有几个特点，一是对抗性强，比赛时间短，有很强的观赏性；二是能够充分的发挥参赛者的创造性思维；三是比赛策略和临场发挥非常重要；四是比赛规则和题目变化较大。尤其是 NHK 国际机器人比赛需要参赛者完成从机械设计制作到控制操作的全部工作，参赛机器人的结构功能以及采用的技术五花八门，参赛者的综合能力可以在参赛过程中得到全面的提高。

2.2.2 研究机器人竞赛的意义

机器人竞赛所包含的关键技术有机器视觉技术、无线电通信技术、机电一体化技术、智能控制技术、仿真技术、软/硬集成技术、人工生命学技术、人机接口技术等；包含着 21 世纪我们要开发的信息、自动化、通信、机电一体化等综合技术。这些关键技术无疑对我国传统产业的技术改造具有实际意义。

为了使人工智能与机器人技术能在更广泛、更深入的层面展开研究,并使其研究成果尽快转化为生产力,在机器人足球成为人工智能与机器人学的标准问题并被广泛开展的同时,近年来,国内外开展了多种形式、多个层面的机器人比赛。把这些竞赛机器人中涉及到的一些共同问题进行深入研究,无疑对学术研究和生产应用都有很强的意义。

2.3 本文主要的课题内容

本文着重介绍了本文作者在本次大赛中负责和参与的几项技术攻关任务——辅助定位的研究，底层路径控制算法的设计和实现，主脑层的自主路径规划的实现，白块目标检测，以及基于计算机视觉的避障行为。需要指出的是，上述研究都是基于机器人能通过各种传感器感知机器人内部和外界信息而实现的。而坐标的准确计算和纠正则是自主

移动机器人体系的核心，是所有上层算法实现的基础。在实现机器人当前坐标精确的前提下，我们解决了机器人直线和圆弧路径精确行走的问题，在底层实现精确行走的前提下，上层的自主路径规划和视觉的目标定位得到了最大限度的发挥。

2.3.1 机器人各自主功能模块

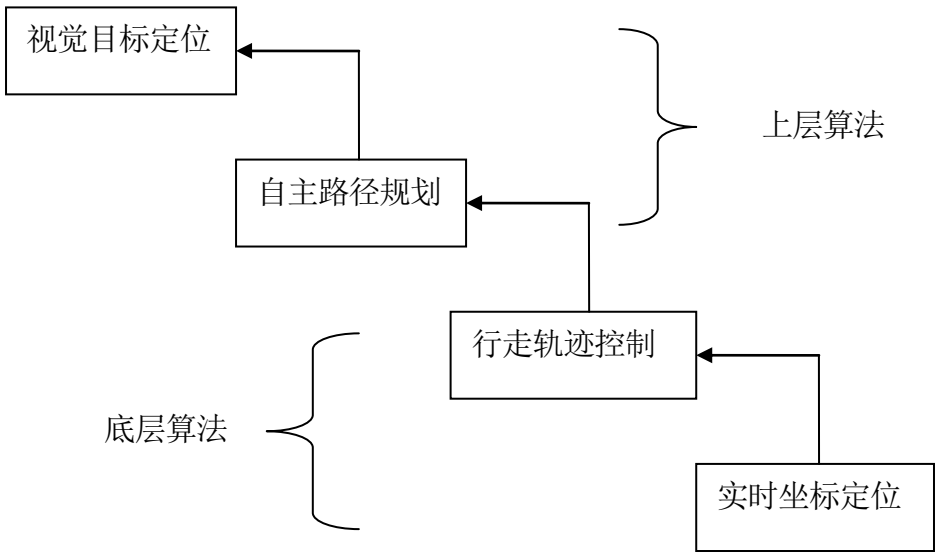


图 2.3 各自主功能模块间的递进关系

辅助定位方面，本文作者利用陀螺仪和码盘的信息实时计算机器人的坐标的同时，利用场地上白线的信息额外加了光纤传感器帮助进行坐标纠偏，克服码盘误差的积累和机器人高速情况下坐标更新周期相对较慢，坐标计算算法产生的系统误差。

机器人底层移动轨迹跟踪方面，把整条路径分解成直线和圆弧的组合，控制机器人按圆弧或者直线行走，圆弧路径的跟踪使用了圆弧半径和行进角度的闭环控制，直线路径的跟踪使用了位置和角度的闭环控制。

路径规划方面，本文作者基于启发式深度搜索和曲线拟合的技术研发了专用于本次比赛的路径算法，它实现了各个自动机器人路径的自主规划，在调试中大大减少了编写一条条固定路线的工作量，另外一个好处是节省了存储固定路径带来的巨大的内存消耗。后又加入了机器人的障碍检测并重新规划无障碍路径，达到了一定的避障效果。

视觉方面，本文作者用自己实践中发现的稳定的色彩空间变换方法并结合目标尺寸信息在复杂的背景中正确地定位了机器人的目标——白块，并用类似的方法对场地中的

白线进行计数，辅助机器人定位；另外，通过景深探测障碍方法的启发，用 canny 算子提取边缘，定位图象边缘较多的复杂区域，对障碍进行检测定位。

2.3.2 机器人硬件体系

机器人硬件系统基于 NXP 公司生产的 ARM7 内核微控制器 LPC2138，是由多个处理器构成的一个一主两从的处理器系统。整个系统分为两个层次，上层系统是一个主控模块，处理上层的路径规划算法、行走控制、机构动作执行作以及智能决策处理，和机构模块之间用 RS-485 总线连接。下层系统为三从系统，负责多个传感器的数据采集及处理任务，为上层系统的算法和控制提供实时的传感器数据，是整个系统的感知模块。下层数据采集和主控间分别采用 SSP,RS-485 以及简单串口通信协议进行实时数据传递。整个系统结构示意图如图 2.4 所示。

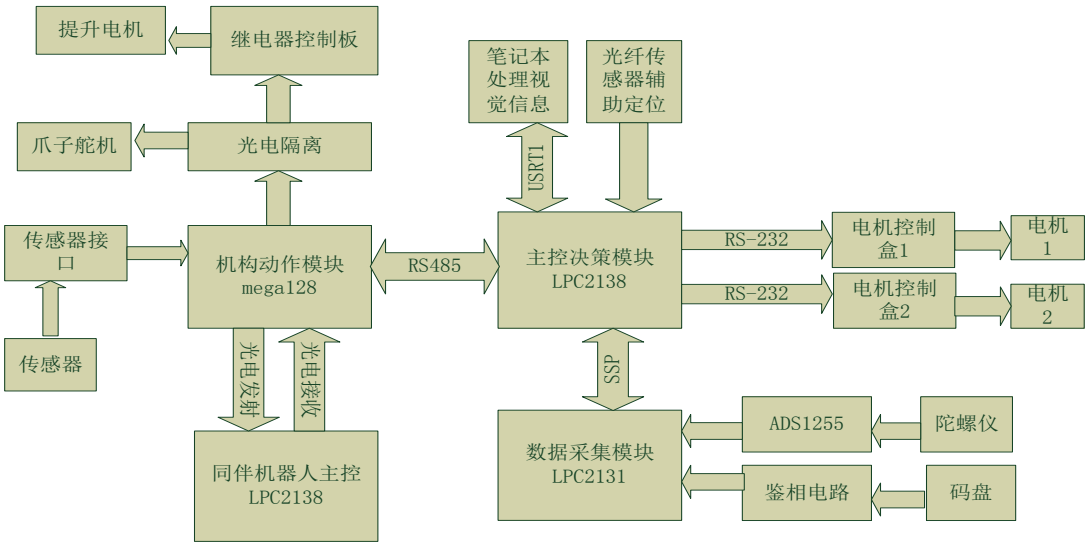


图 2.4 双升降翻转型机器人硬件系统

整个系统由底层到上层协同工作，完成陀螺仪的模拟信号 AD 转换、AD 芯片转换结果数据采集、陀螺仪数据处理得出机器人角度；码盘脉冲信号采集和处理得出机器人的行走距离；动作模块控制提升电机和爪子机构舵机；主控决策模块根据导航模块和动作模块的任务完成情况选择不同的策略，并协调导航模块和动作模块的动作，主控模块还负责在两个机器人配合取中央黄色黄油时与另一个自动机器人的通信。

2.3.3 机器人软件体系

为了实现机器人的智能性，软件的框架设计非常重要。软件框架的基本思想是尽量去除由于不同的地图，不同的机器人，不同的策略带来的各个代码模块间的耦合，实现机器人的软件基本框架可以用一下的几个核心数据结构来表示。

AI_TASK 任务是机器人任务调度的基本单位。可以工作在'强制任务'和'调度任务'两种工作模式下。这两种模式可以相互转换。

1） 强制任务

强制任务就是所谓的顺序任务。在该任务下，机器人 具有最小的智能和非常有限的异常处理能力。任务按照存储顺序依次被执行。任务的制定者必须规定机器人的行走路径（机器人也具有有限的路径规划能力）动作与动作之间的先后关系是固定的，任何一个任务执行失败都将导致整个强制任务执行序列的中断，任务调度模式会自动转入'调度任务'状态。在强制任务模式下，如果任务执行出现异常，机器人会自动重试规定次数（或者在规定的时间内一直重试）。

2） 调度任务

调度任务是一种规定动作、执行范围、执行有效时间允许机器人根据以上信息综合优先级、距离远近自动决策实现任务执行的工作模式。各任务之间不存在依附关系和先后关系，他们只有优先级和执行时间的区别。系统按照时间顺序从高优先的任务中选取目标点最近的任务来执行。一旦任务执行错误就自动放弃，直接进行下一轮任务调度。如果获得执行的任务存在后续的强制任务，机器人就会自动切换到强制任务模式去执行。该模式具有很强的自由度。需要精心安排来产生一个高智能、高效率、高应变能力的策略。

```
struct AITask//智能任务
{
    volatile POINT      StartPoint;          //机器人出发点的坐标
    volatile POINT      EndPoint;            //机器人最终目标点的坐标
    volatile POINT      BlockPoint;          //白块的信息
    volatile POINT      StopPoint;           //障碍的信息
    volatile uint8      WorkRetryTimes;      //动作重试次数
```

```
    volatile uint8          cError;                     //出错代码
    volatile uint8          RunningMode;                //运行模式
    volatile uint8          MoveRetryTimes;
    volatile uint8          NeedReachStatus;            //是否需要接近状态来查询传感器
    volatile uint8          PathNum;                    //路径段数
    volatile uint8          Strategy;                   //策略
    PATH（*pFixedPath）[SUBPATH_NUM_OF_ONE_TASK]; //一个策略的子任务路径集合
    AI_TASK_POINTER   pNext;                            //后一个任务
    AI_TASK_POINTER   pPre;                             //前一个任务
    volatile uint8          taskPos;                    //一个策略的任务段
    volatile uint8          stopIsFound;                //障碍被探测到
    volatile uint8        FirstSecondPos;               //摆放位置
};
```

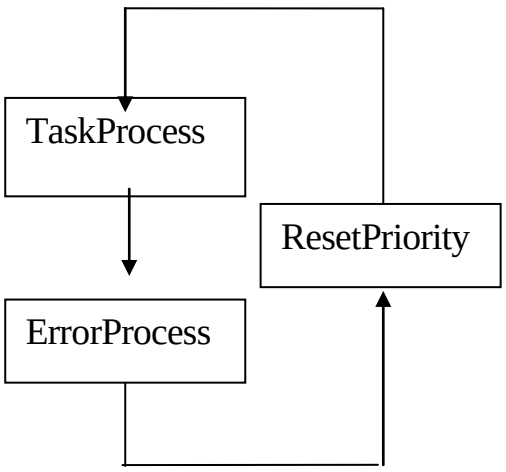


图 2.5 机器人软件的基本框架

软件框架的基本流程如图 2.5 所示，机器人在不停反复地执行这个流程，首先开始执行任务的处理函数 TaskProcess()，当任务顺利完成或者出现某种问题失败后，跳出TaskProcess()，进入错误处理函数 errorProcess(),完成善后处理，ResetPriority()更新各个任务的优先级，生成一个新的任务，再自动进入下一个任务的处理函数 TaskProcess()，不断循环这个流程。具体的代码实现如下所示。

```
/*----------------------------------------------------任务调度----------------------------------------------------*/
 do{
```

```
AITaskInit（pTask）;
  for（i = 1;i<7;i++）{//运行状态，固定模式
    if（0 == （*pFunArray[i]）（pTask））
        i = ErrorProcess（pTask）;
    else{
        if（i == 6）{
            pTask->cError = TASK_DONE;
            pTask   = Go2NextTask（pTask,FIX_MODE）;
        }
    }
  }//for
}//do
while（（pTask!=NULL））;//cError 代表没有错误，顺利完成子任务
```

在错误处理函数中，如果重试次数大于零，我们将对当前子任务的目标重新规划一条新的路径，否则，则转入下个子任务，并规划出到下个子任务目标点的路径。

结合实际硬件来说，主控单片机 LPC2138 通过外接口控制、读取底层数据采集、负责视觉的笔记本和机构控制三个模块的信息，进行情况判断决策和路径规划，并结合轨迹闭环跟踪算法计算出两底盘电机的目标速度，再将速度命令发送给两个电机控制器从而进行行走控制，到达既定位置后向机构模块发送机构动作的命令。底层数据采集单元根据从动轮码盘和陀螺仪的信息得到机器人在场地上的里程和绝对角度，并将这些信息发送给主控单元。机构控制模块等候主控单元的调遣，接受到指定的动作命令时执行相应的动作，接受到查询状态的命令时，向主控单元返回相应的状态。

在时序上，底层数据采集单元通过查询定时器实现 5ms 一次循环，更新角度和里程信息，通过 5ms 一次外部中断向主控单元发送角度和里程信息。

2.4 本章小结

本章论述本课题背景与意义，简单分析了这次比赛对自主移动机器人的功能要求和机器人相应的的双升降翻转型的机械设计，介绍当前自主移动机器人的各智能功能模块，简述本机器人系统的软硬件设计框架。这样，我们就可以根据这个软硬件系统构架图进一步在软件和硬件上搭建系统，为后面的工作打好基础。

3 自主移动机器人硬件系统设计

3.1 硬件系统总体结构

本章将详细介绍 V 型双提升自动机器人的硬件系统。该硬件系统由 NXP 公司生产分别作为主控板的 ARM7 内核微控制器 LPC2138、作为导航板的 LPC2131 和作为机构板 mega128 多个处理器以及作为视觉模块的笔记本构成的一个一主三从的处理器系统。主脑层负责路径规划算法、行走控制、以及智能决策处理，用 RS-485 总线和机构板连接。导航模块负责多个传感器的数据采集及处理任务，用 SSP 和主脑通信，为上层系统的算法和控制提供实时的传感器数据，是整个系统得感知模块。主脑和视觉模块之间则采用简单协议的串口通信方式。整个系统结构示意图如图 3.1 所示。

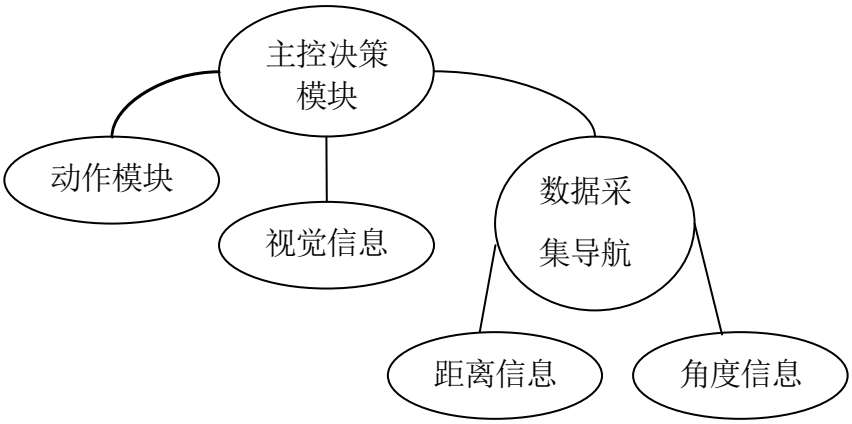


图 3.1 硬件系统结构示意图

整个系统由底层到上层协同工作，完成陀螺仪的模拟信号 AD 转换、AD 芯片转换结果数据采集、陀螺仪数据处理得出机器人角度；码盘脉冲信号采集和处理得出机器人的行走距离；导航模块微控制器通过 SPI 串行通信接口从陀螺板读取角度值，并通过鉴相电路对码盘数据距离信息，通过 SSP 将数据传给主控模块；动作模块控制提升电机和爪子机构舵机。考虑到 mrga128 单片机在 IO 口资源上更有优势，并且在控制机构上的表现更稳定，所以使用 mega128 单片机作为机构动作模块；主控模块根据导航模块和动作模块和视觉模块的任务完成情况选择不同的策略，并协调导航模块和动作模块的动作，主控模块还负责在两个机器人配合取中央黄色黄油时与另一个自动机器人的通信硬件系统的详细结构如图 3.2 所示。

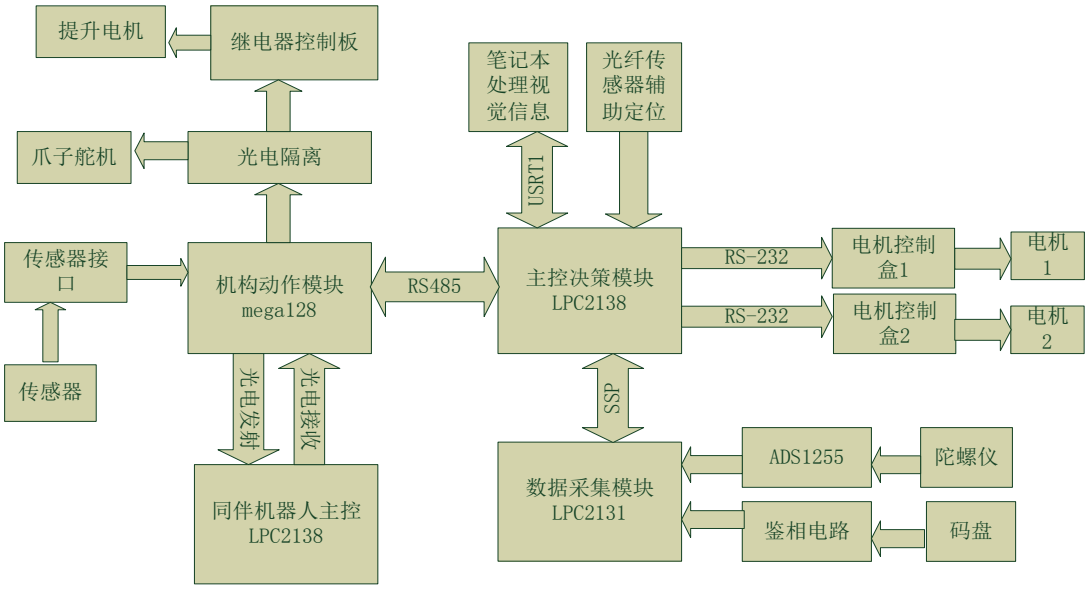


图 3.2 双升降翻转型机器人硬件系统

3.2 机器人系统核心处理器-ARM7 处理器

在我们的自主移动机器人上，由于条件的限制，并没有使用 FPGA 和 CPLD 等多种类型的芯片，而是使用了 ARM7 微处理器来实现包括数据采集，任务调度在内的所有功能。

3.2.1 ARM7--LPC213x 系列微处理器

ARM7TDMI-S 核是通用的 32 位微处理器内核，采用冯·诺依曼结构，它具有高性能和低功耗的特性。ARM 结构是基于精简指令集计算机（RISC，Reduced Instruction Set Computer）原理而设计的，指令集和相关的译码机制比复杂指令集计算机要简单得多，由此可见使用一个小的、廉价的处理器核就非常容易实现很高的指令吞吐量和实时的中断响应。

ARM7TDMI-S 处理器使用了一个被称为 Thumb 的独特结构化策略，它非常适用于那些对存储器有限制或者需要较高代码密度的大批量产品的应用。

基于 Thumb 的一个关键的概念就是“超精简指令集”。基本上，ARM7TDMI-S 处理器具有两个指令集：

- 标准 32 位 ARM 指令集；
- 16 位 Thumb 指令集。

Thumb 指令集的 16 位指令长度使其可以达到标准 ARM 代码两倍的密度，却仍然保持 ARM 的大多数性能上的优势，这些优势是使用 16 位寄存器的 16 位处理器所不具备的。因为 Thumb 代码和 ARM 代码一样，在相同的 32 位寄存器上进行操作。Thumb 代码仅为 ARM 代码规模的 65%，但其性能却相当于连接到 16 位存储器系统的相同 ARM 处理器性能的 160%。关于 ARM7TDMI-S 处理器的详细内容请参阅 ARM 官方网站上的 ARM7TDMI-S 数据手册。

LPC2131/2132/2134/2136/2138 是基于一个支持实时仿真和跟踪的 16/32 位 ARM7TDMI-S™ CPU 的微控制器，并带有 32/64/128/256/512 K 字节嵌入的高速 Flash 存储器。128 位宽度的存储器接口和独特的加速结构使 32 位代码能够在最大时钟速率下运行。对代码规模有严格控制的应用可使用 16 位 Thumb 模式将代码规模降低超过 30%，而性能的损失却很小。

LPC2131/2132/2134/2136/2138 特性

1） 小型 LQFP64 封装的 16/32 位 ARM7TDMI-S 微控制器。

2） 8/16/16/32/32KB 片内静态 RAM。

3） 32/64/128/256//512KB 片内 Flash 程序存储器。128 位宽度接口/加速器实现高达 60MHz 的操作频率。

4） 片内 Boot 装载程序实现在系统编程（ISP）和在应用中编程（IAP）。Flash 编程时间：1ms 可编程 256 字节，扇区擦除或整片擦除只需 400ms。

5） EmbeddedICE-RT 和嵌入式跟踪接口可实时调试（利用片内 RealMonitor 软件）和高速跟踪执行代码。

6）1 个（LPC2131/2132）或 2 个（LPC2134/2136/2138）8 路 10 位 A/D 转换器共包含 8/16 个模拟输入，每个通道的转换时间低至 2.44us。

7） D/A 转换器可产生不同的模拟输出（仅适用于 LPC2132/2134/2136/2138）。

8） 2 个 32 位定时器（带 4 路捕获和 4 路比较通道）、PWM 单元（6 路输出）和看门狗。

9） 实时时钟具有独立的电源和时钟源，在节电模式下极大地降低了功耗。

10） 多个串行接口，包括 2 个 16C550 工业标准 UART、2 个高速 I2C 接

口（400 Kbit/s）、SPI 和具有缓冲作用和数据长度可变功能的 SSP。

11） 向量中断控制器。可配置优先级和向量地址。

12） 多达 47 个可承受 5V 的通用 I/O 口（LQFP64 封装）。

13） 多达 9 个边沿或电平触发的外部中断引脚。

14） 通过片内 PLL 可实现最大为 60MHz 的 CPU 操作频率。

15） 片内晶振频率范围：1~30 MHz。

16） 两个低功耗模式：空闲和掉电。

17） 可通过个别使能/禁止外部功能和降低外部时钟来优化功耗。

18） 通过外部中断将处理器从掉电模式中唤醒。

19） 单个电源供电，含有上电复位（POR）和掉电检测（BOD）电路：CPU 操作电压范围：3.0~3.6 V （3.3 V±10%）

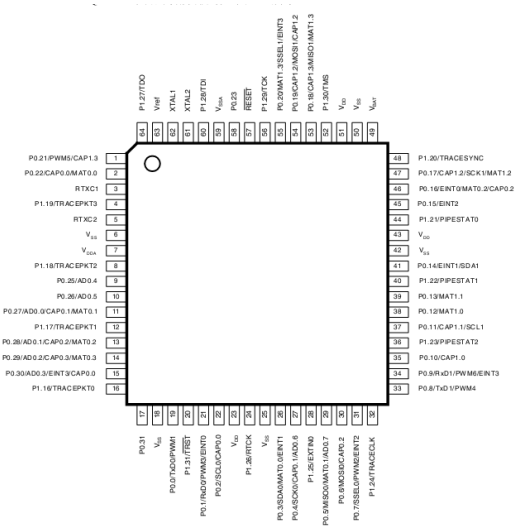


图 3.3 LPC2131 的 64 脚封装

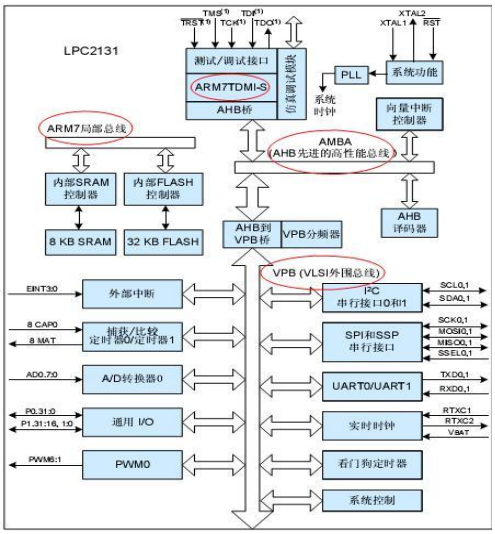


图 3.4 LPC2131 方框图

在我们的机器人中，使用了 2138 和 2131ARM 处理器。

3.2.2 LPC213x 系列微处理器开发环境介绍

ADS 集成开发环境是 ARM 公司推出的 ARM 核微控制器集成开发工具，英文全称为 ARM Developer Suite，成熟版本为 ADS1.2。ADS1.2 支持 ARM10 之前的所有 ARM 系列微控制器，支持软件调试及 JTAG 硬件仿真调试，支持汇编、C、C++源程序，具有编译效率高、系统库功能强等特点，可以在 Windows98、

Windows XP、Windows2000 以及 RedHat Linux 上运行[6] [7]。

这里将简单介绍使用 ADS1.2 建立工程，编译连接设置，调试操作等等。最后还介绍了基于 LPC2131 系列 ARM7 微控制器的工程模板的使用。 由于用户一般直接操作的是 CodeWarrior IDE 集成开发环境和 AXD 调试器，所以这一章我们只介绍这两部分软件在 Windows 下的使用，其它部分的详细说明参考 ADS 1.2 的在线帮助文档或相关资料。

1） CodeWarrior IDE 简介

ADS 1.2 使用了 CodeWarrior IDE 集成开发环境，并集成了 ARM 汇编器、ARM 的 C/C++编译器、Thumb 的 C/C++编译器、ARM 连接器，包含工程管理器、代码生成接口、语法敏感（对关键字以不同颜色显示）编辑器、源文件和类浏览器等等。CodeWarrior IDE 主窗口如 图 3.5 所示。

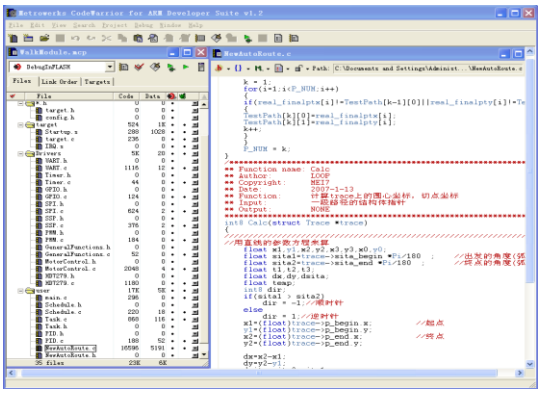


图 3.5 CodeWarrior IDE 主窗口

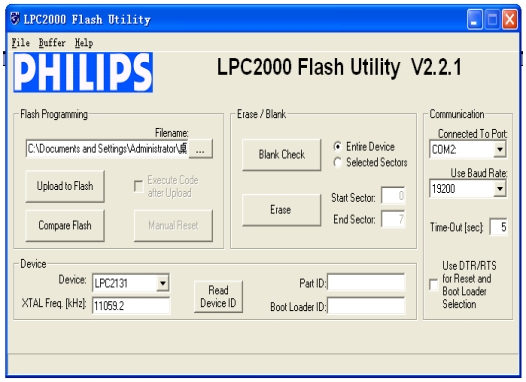


图 3.6 程序下载工具主窗口

2） AXD 调试器简介

AXD 调试器为 ARM 扩展调试器（即 ARM eXtended Debugger），包括 ADW/ADU 的所有特性，支持硬件仿真和软件仿真（ARMulator）。AXD 能够装载映像文件到目标内存，具有单步、全速和断点等调试功能，可以观察变量、寄存器和内存的数据等等。AXD 调试器主窗口如 图 3.6 所示。

3.3 电源模块硬件设计

3.3.1 电池选择

电路工作必须有稳定可靠的电源供给，我们采用可充电型电池。目前应用较广的可充电电池主要有四类，即小型密封式维护铅酸蓄电池、镍镉电池、镍氢电池和锂电

池。铅酸电池具有电压平稳，安全性好，维护简便等特点，它功率特性优良，自放电低，一般充电后搁置 4 个月容量损失不超过 10%，但体积和重量较大。镍镉电池体积小、容量大，具有可快速充电、大电流放电、免于维护等特点，循环使用寿命长（2000 次），为铅酸电池的 2 倍以上，但镍镉电池具有记忆效应，同时镉具有强污染性。镍氢电池具有能量密度高、可快速充放电、循环寿命长、无记忆效应、无污染、免维护等特点，镍氢电池比能量是镍镉电池的 1.5~2 倍，具有良好的耐过充、过放电特性。

锂电池是近几年发展最为迅猛的一类动力电池，它具有重量轻、体积小、内阻小，自放电小、放电电压平稳、无污染、无记忆效应等特点。锂电池的能量是大容量镍镉电池的 1.5 倍，密度是其 2 倍，平均比能量是镉/镍电池的 2.6 倍，可在-20~+60°C 范围内稳定工作，虽然锂电需要特殊充电器进行充电，但综合性能高于前三种。

考虑到本届自动机器人具有体积小、重量轻、选用电机功率大的特点，我们选用锂电池进行电源供电，经测试发现其放电电压平稳，一组 24V 锂电连续使用 48 小时后仍能平稳供电，效果优越。

图 3.7 锂电池平衡充电器

但是需要强调的是，过度充电与放电会使锂电池报废。过度充电和过度放电，将对锂离子电池的正、负极造成永久的损坏。这与镍氢、镍镉电池的做法截然不同。另外，由于锂离子电池的电记忆效应小得可以忽略不计，所以半途充电对它的损伤并不是很大，而它最怕的是过度放电（完全放电）。下面是我们使用的平衡充电器简介：

输入电压：9~15V DC，分压式平衡充电，充电电流：0.5/1.0/1.5/2.0 A （1~4 CH），1~4 节锂离子/锂聚合物电池 ,如图 3.7 所示。

3.3.2 电源模块电路设计

整个系统的供电需求为：模拟电路部分提升电机和底盘电机控制盒需要 24V 供电，由于不需要稳压，直接用 24V 锂电池供电。数字电路部分 LPC2131 需要 3.3V 供电，码盘需要 5V 供电，陀螺仪需要 5V 供电。陀螺板内部包含 5V 和 3.3V 稳压芯片分别给陀螺仪和负责陀螺仪数据采集的 LPC2131 以及模数转换芯片 ADS1255 供电（ADS1255 模拟供电需要 5V，数字供电需要 3.3V）。给陀螺板的供电为 6V。由于主板上对稳压芯片的发热量要求不高，主板的 5V 和 6V 电压用 78 系列线性稳压芯片 7805 和 7806 由电池的 12V 电压稳压得到。给 LPC2131 微控制器的 3.3V 供电由线性稳压芯片 LD1117 从 5V 稳压得到。在陀螺板内部，由于对要求稳压芯片必须发热量小，避免发热量影响陀螺仪的精度。所以 5V 电压由 TPS7350 从 6V 稳压得到，3.3V 电压由 TPS7333 从 6V 稳压得到。

根据以上需求分析，电源模块电路如图 3.8 和图 3.9 所示。

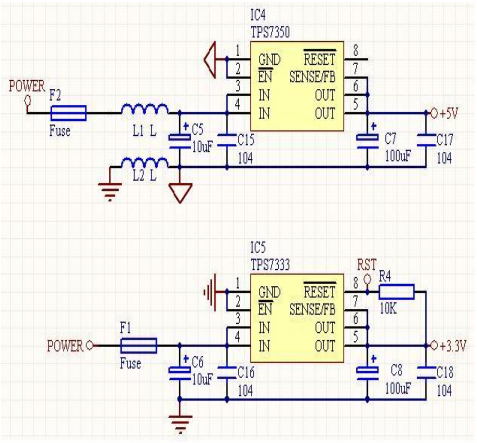


图 3.8 陀螺板电源部分

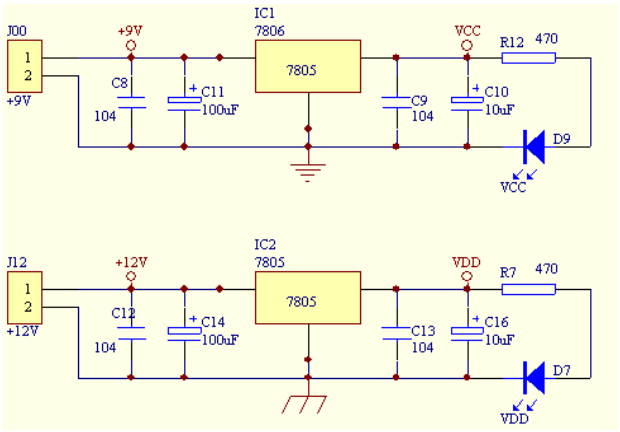


图 3.9 主板电源部分

3.4 主控模块硬件设计

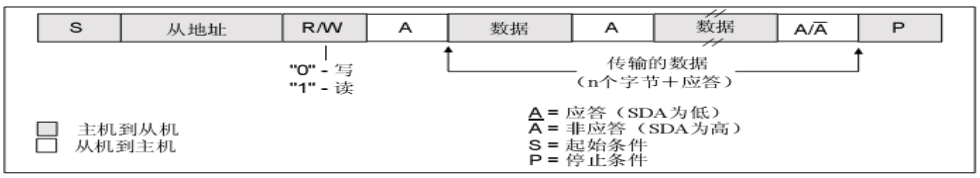
3.4.1 主控模块功能分析

主控负责任务调度，各种异常处理，协调机构、定位、行走、视觉等各模块，使机器人系统顺畅运行，由于主要是和其余模块进行通信，主控模块硬件上比较简单，几乎就是单片机的最小系统板，由于要和机构进行板级间通信，我们采用了 RS-485 总线。

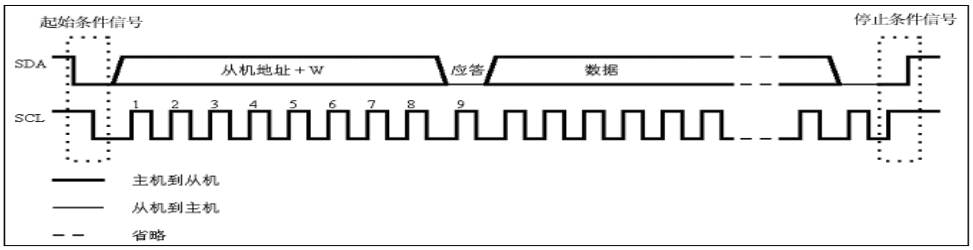
3.4.2 RS-485 总线研究

1） 通信要求

模块之间的通信要求稳定性好，抗干扰能力强，速度在 100Kbps 以上。稳定型和抗干扰能力强是为了保证机器人在场上的稳定发挥，如果通信出现紊乱会导致整个系统的崩溃。速度要求在 100K 以上是为了使通信不至于占用微控制器太多的资源，为程序的其它任务处理让出了时间。初期使用过 I^2C 总线，速度能达到 100Kbps，在高速 I^2C 模式下能达到 400Kbps。实际使用过程中，I^2C 总线协议和时序复杂，传输一个字节需要开始信号、从机地址、读写信号、应答信号、数据、应答信号、停止信号，每一位的传递基于时钟信号 SCK，如图 3.10 所示。时钟信号 SCK 容易受到干扰，在传输过程中任何一个环节出现错误都会导致通信的失败，所以 I^2C 总线是板级总线，适合于



（a）I2C 主模式数据格式



（b） I2C 时序

图 3.10 I2C 主模式数据格式和 I2C 时序

PCB 板内部芯片的通信。实际使用中容易出现通信失败，所以鉴于对稳定性的要求，采用工业标准的 RS-485 总线。

2） RS-485 标准

在数据通信，计算机网络以及分布式工业控制系统当中，经常需要使用串行通信来实现数据交换。目前，有 RS-232,RS-485,RS-422 几种接口标准用于串行通信。RS-232 是最早的串行接口标准，在短距离（<15M），较低波特率串行通信当中得到了广泛应用。其后针对 RS-232 接口标准的通信距离短，波特率比较低的状况，在 RS-232 接口标准的基础上又提出了 RS-422 接口标准，RS-485 接口标准来克服这些缺陷。

RS-485/422 采用平衡发送和差分接收方式实现通信：发送端将串行口的 TTL 电平信号转换成差分信号 A,B 两路输出，经过线缆传输之后在接收端将差分信号还原成 TTL 电平信号。由于传输线通常使用双绞线，又是差分传输，所以又极强的抗共模干扰的能力，总线收发器灵敏度很高，可以检测到低至 200mV 电压。故传输信号在千米之外都是可以恢复。RS-485/422 最大的通信距离约为 1219 M，最大传输速率为 10Mbps，传输速率与传输距离成反比，在 100Kbps 的传输速率下，才可以达到最大的通信距离，如果需传输更长的距离，需要加 485 中继器。

在实际设计过程中用 LPC2138 的串行通信接口 UART1 作为 RS-485 的接口，P0.10 管脚作为发送器和接收器的使能端。RS-485 总线接口电路如图 3.11 所示。

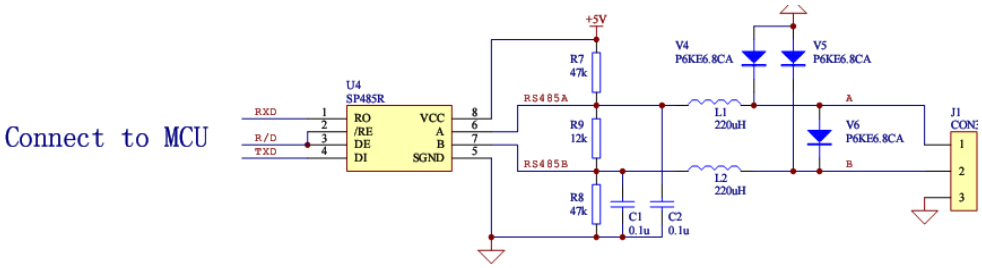


图 3.11 RS-485 总线接口电路

3.4.3 主控模块电路

主控模块由总线接口和与同伴机器人的通信接口组成，电路设计如图 3.12 所示。

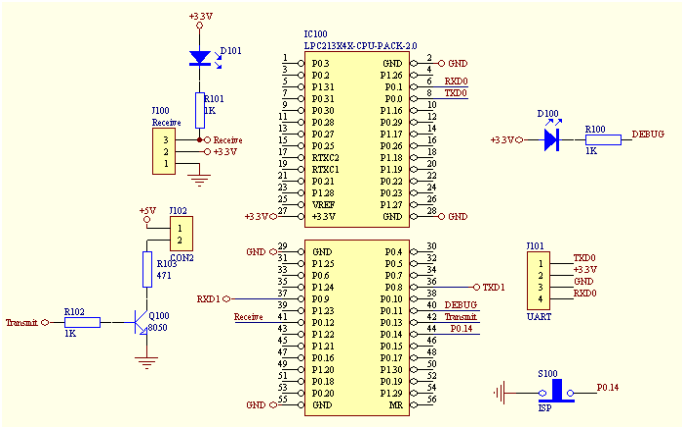


图 3.12 主控模块电路设计

3.5 导航模块硬件设计

3.5.1 导航模块总体结构

导航模块主要负责自动机器人的行走控制，全场使用坐标定位。进行路径的规划和控制电机沿着规划的路径行走。为了完成该任务，必须结合陀螺仪的角度和码盘的距离信息。所以导航模块必须完成陀螺仪数据采集和处理以及码盘脉冲的采集和处理，下层将处理好得到的角度信息和距离信息传递到上层进行信息的融合处理。所以导航模块的主控芯片需要与下层陀螺板进行通信以获得数据，同时对码盘进行采样，导航模块的结构图如图 3.13 所示[8]。

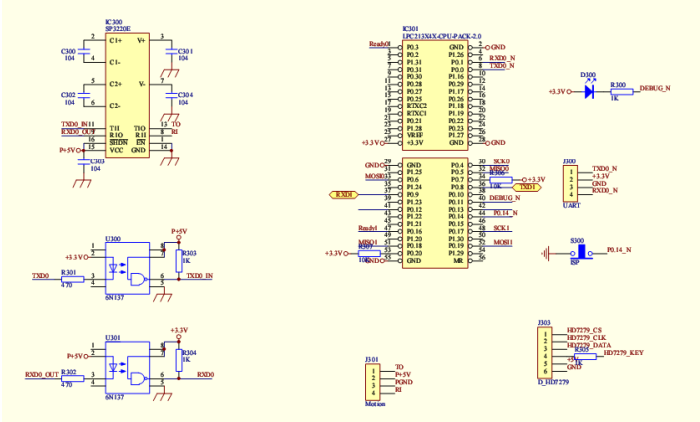


图 3.13 导航模块硬件结构图

3.5.2 SPI 串行通信

在导航模块内部，从码盘得出了机器人的行走距离信息是由 IO 口输入捕获实现的，但是机器人的角度信息从陀螺板中的 ADS1255 A/D 芯片得到的。此时需要一种短距离高速通信方式将信息实时地将角度信息传递到导航模块芯片中。导航模块处理器 LPC2131 具有一个硬件 SPI（SPI，Serial Peripheral Interface）接口，它是一个同步、全双工串行接口，最大数据位速率为时钟速率的 1/8，可以配置为主机或者从机。SPI 的此特性满足了高速串行数据交换的要求。且接口简单，由 MISO、MOSI、SCK、SSEL 四线组成。

SPI 可应用于：

1）串行存储器，如 DataFlash、三线 EEPROM 等；

2）串行外设，如 ADC、DAC、LCD 控制器、CAN 控制器、传感器等；

3）外部协处理器。

SPI 在同一总线上可以有多个主机或者从机，但同一时刻只能有一个主机和一个从机能够进行通信，在一次数据传输过程中，主机向从机发送一字节数据，从机也向主机返回一字节数据。因为 LPC2131 具有一个硬件 SPI0 接口和一个硬件 SSP（兼容 SPI，也成为 SPI1），所以系统设计为一主一从模式。陀螺板与导航模块主控通信采用 SSP（SPI1），陀螺板为从机；码盘板与导航模块主控通信采用 SPI0，码盘板为从机。

SPI 的通信时序图如图 3.14 所示。

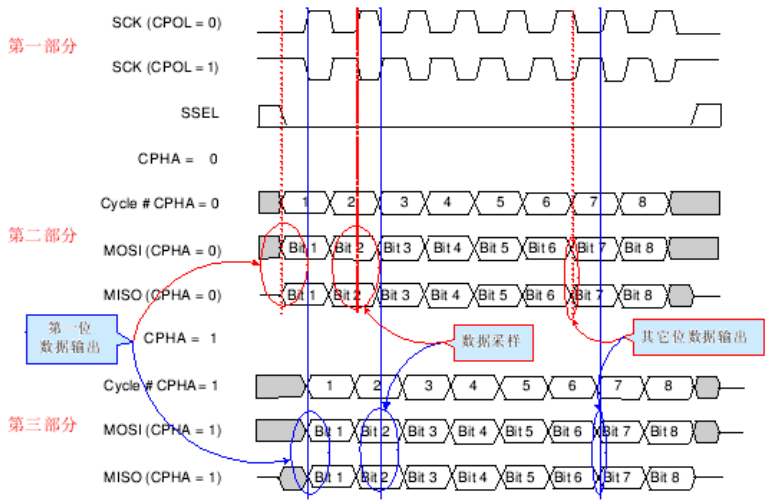


图 3.14 SPI 通信时序图

SPI 具有 4 种不同的传输模式，如表 3.1 所示。

表 3.1 SPI 数据和时钟的相位关系

CPOL 和 CPHA 的设定	驱动的第一个数据	驱动的其它数据	采样的数据*
CPOL=0, CPHA=0	在第一个 SCK 上升沿之前	SCK 下降沿	SCK 上升沿
CPOL=0, CPHA=1	第一个 SCK 上升沿	SCK 上升沿	SCK 下降沿
CPOL=1, CPHA=0	在第一个 SCK 下降沿之前	SCK 上升沿	SCK 下降沿
CPOL=1, CPHA=1	第一个 SCK 下降沿	SCK 下降沿	SCK 上升沿

当器件为主机时，传输的起始由包含发送数据字节的主机来指示。此时，主机可激活时钟并开始传输。当传输的最后一个时钟周期结束时，传输结束。当器件为从机并且 CPHA=0 时，传输在 SSEL 信号激活时开始，并在 SSEL 变为高电平时结束。当器件为从机且 CPHA=1 时，如果该器件被选择，传输从第一个时钟沿开始，并在数据采样

的最后一个时钟沿结束。器件作为主机时，SSEL 口要加上拉电阻，而从机的 SSEL 口则不能加上拉，当主机需要与某个从机进行通信时,需要使用主机的一个普通 I/O 口来置低从机的 SSEL,从而选择该从机。

我们在应用中，设置 SPIclk = Fpclk / SPCCR = 11.0592*4 MHZ / 8， 数据在第一个时钟沿采样。在中断中 SPI 读取数据寄存器之前必须先读 SPSR 寄存器，清零 SPIF 位，才能访问数据寄存器。

3.5.3 陀螺角度采集

陀螺仪采用俄罗斯 FIZOPTIKA 公司的光纤陀螺仪 941-3AM，如图 3.15 所示。

图 3.15 光纤陀螺仪 941-3AM

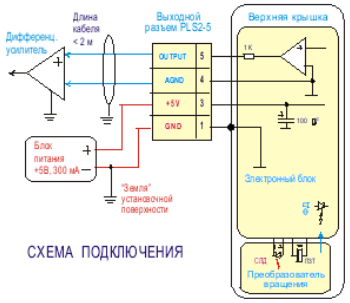


图 3.16 941-3AM 示意图

941-3AM 是一个模拟输出的角速率传感器，它通过电压的形式响应运动物体的角速率，传感器的输出被定义为“Output”与“NGnd” （模拟地）之间的电压值。输出电压的符号依赖于绕敏感轴旋转的方向。输出电压与旋转角速率成比例。测量范围为±600 度/秒。供电采用+5V 供电。输出电压在±5V 之间，如比例因子为 3.6mV/（度/秒），则输入角速度为 600 度/秒。输出电压为 1.8V。941-3AM 的示意图如图 3.16 所示。941-3AM 的零偏变化因子为 0.001deg/s，3 分钟内漂移 0.18 度，完全满足了比赛要求。为了与 941-3AM 的高精度相匹配，采用 24 位 AD 转换芯片。941-3AM 的频率范围为 0~500Hz，根据采样定理，所选用的 AD 芯片的采样速率高于 1KHz 即可，实际设计选用 TI 公司的 24 位∑-Δ 型模数转换芯片 ADS1255。ADS1255 的采样频率可编程，最高可达 30KHz，完全满足了需要。下面简要介绍∑-Δ 型模数转换器。

∑-Δ 转换器又称为过采样转换器。这种转换器由∑-Δ 调制器及连接及其后的数字滤波器构成，如图 3.17 所示。

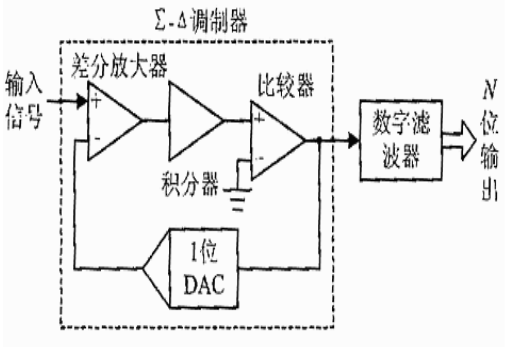


图 3.17 ∑-Δ 转换器示意图

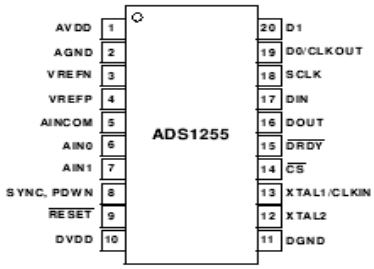


图 3.18 ADS1255 管脚图

调制器的结构近似于双斜率模数转换器，包括 1 个积分器和 1 个比较器，以及含有 1 个 1 位数模转换器的反馈环。这个内置的数模转换器仅仅是一个开关，它将积分器输入切换到一个正或负参考电压。∑-Δ 模数转换器还包括一个时钟单元，为调制和数字滤波器提供适当的定时。窄带信号送入∑-Δ 模数转换器后被以非常低的分辨率（1 位）进行量化，但采样频率却非常高。经过数字滤波处理后，这种过采样被降低到一个比较低的采样率；同时模数转换器的分辨率（即动态范围）被提高到 16 位或更高。

尽管∑-Δ 模数转换器采样速率较低，且限于比较窄的输入带宽，但在模数转换器市场上仍占据了很重要的位置。它具有三个主要优势：

1) 低价格、高性能（高分辨率）；

2) 集成化的数字滤波；

3) 与 DSP 技术兼容，便于实现系统集成。

ADS1255 的输入范围为 0~5V，而 941-3AM 的输出有负电压，所以在输入端需要设计一加减运算电路，将负电压转换为正电压。ADS1255 集成了 SPI 串行通信接口，AD 转换结果通过该接口方便地传递到微控制器，且微控制器通过该接口可对 ADS1255 进行编程设置，例如设置转换速率等。ADS1255 的管脚分布如图 3.18 所示。

陀螺板的整体电路设计如图 3.19 所示。

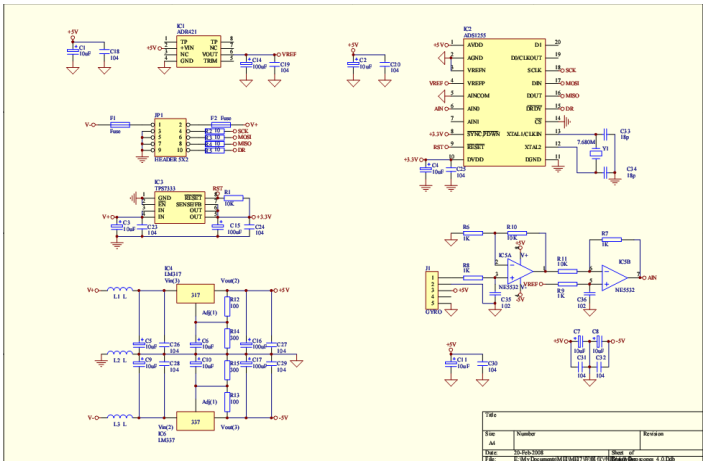


图 3.19 陀螺板电路设计

3.5.4 码盘距离采集

1） 光电编码器基本原理介绍

光电编码器角度检测传感器是一种广泛应用的编码式数字传感器，它将测得的角位移转换为脉冲形式的数字信号输出，如图 3.20 所示。光电编码器角度检测传感器可以分为绝对式光电编码器和增量式光电编码器。

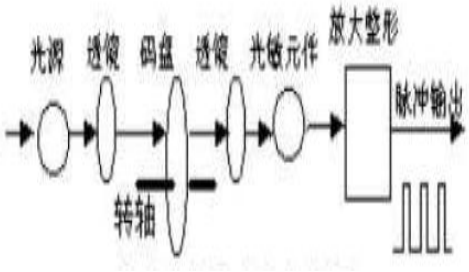


图 3.20 光电编码器原理示意图

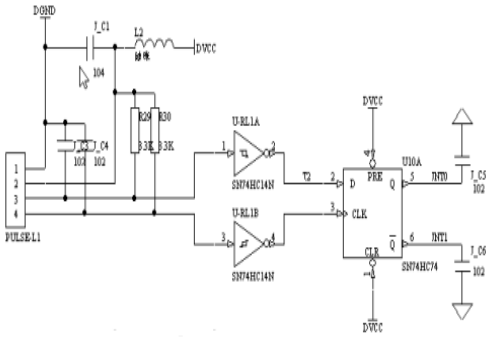


图 3.21 加滤波的码盘鉴相电路

（1） 光学式绝对型旋转编码器

绝对式光码盘是由码盘和光电检测装置组成，码盘采用照相腐蚀工艺，在一块圆形光学玻璃上刻出透光和不透光的编码。编码盘分为若干个扇区，代表若干个角位置，每

个扇区分成 n 条码道，代表 n 位二进制数。为了保证低位码的精度，一般把最外码道作为编码的最低位。其光电检测原理如下：

光源位于编码盘的一侧，n 只光敏三极管位于另一侧，光电三极管接收到光信号时输出低电平“0”，未接收到时输出高电平“1”，将“0”、“1”信号按照由外向里的顺序输出 n 位二进制数即为所测得的角位置。

（2） 光学式增量型旋转编码器

增量式光电编码盘是在一个码盘上开出 3 条码道，由内向外依次为 A、B、C，在 A、B 码道上等距离开有透光的缝隙，2 条码道上相邻的缝隙互相错开半个缝宽，第 3 条码道 C 只开出一个缝隙，用来表示码盘的零位。在码盘的两侧分别安装光源和光敏元件，当码盘转动时，光源经过透光和不透光的区域，就有一系列脉冲从敏感元件输出。码道上有多少缝隙，就会有多少个脉冲输出。

（3） 鉴于机器人安装空间的限制，我们选用 OMRON 体积较小的 200 线码盘。

2） 码盘鉴相电路

光码盘反馈的脉冲经过施密特反向器 SN74HC14 后整形为图中的 T2，之后输入给 D 触发器 SN74HC74，产生方向信号 INT0、INT1。这三路信号将电机的速度信息反馈回了底层数据采集单片机，从而实现闭环调速。在实际应用时，由于电机对码盘的脉冲输出存在电磁干扰、机械震动干扰，从而导致码盘鉴相电路输出的方向信号出现了毛刺，这些毛刺直接影响了单片机对电机旋转方向的判断，从而导致电机在恒速过程中，出现了偶然的抖动，这个极大地影响了电机速度的调节。因此，我们在鉴相电路的 A、B 项输入端加上了滤波电容，虑除码盘反馈信号的尖峰干扰；在鉴相电路的方向输出端加上了电容，虑除了方向信号上的瞬时尖峰干扰；在码盘的电源输入端，加上 LC 吸波电路，防止电源干扰。改进后的码盘鉴相电路如图 3.21 所示。改进后的码盘鉴相电路能较好地虑除掉码盘反馈信号上的干扰。

但是，该电路也存在很大的不足，码盘 A、B 项输出信号上的滤波电容对码盘输出脉冲的频率有限制作用。此处的电容相当于一个低通电路，经过该电路后，反馈脉冲的上升沿和下降沿变长，当码盘转速增大时，反馈脉冲的波形将发生较大的变形，在高速下，码盘鉴相电路对变形后的 A、B 项信号不能进行识别，从而导致鉴相失败，不能形

成闭环反馈。由于时间有限，我们没能在码盘鉴相电路上再作较为深入的研究和改进，直接沿用上一届的鉴相电路。实际测试机器人的行走距离精度满足要求。

3.5.5 光纤传感器

图 3.22 autonics 光纤传感器 BF4R

该传感器具有以下特征：

- 高应答速度 0.5ms 以下；
- 自动感度设定（按钮设定）；
- 自动可选择 Light On/Dark On 模式
- 适合小型物体的检测

该传感器比一般的光电相比，便于标定，可靠性更高；它的缺点时体积较大，不便于布线。我们在机器人底盘底下安装了三个光点传感器，该三个传感器用来检测场地上的白线，进行坐标更新，是机器人基于多传感器进行全场定位的关键。

3.6 视觉模块

3.6.1 华硕 EPC 笔记本

机器人的视觉功能依赖于一款非常小巧的笔记本—华硕 EPC.。该笔记本的特点是轻巧廉价，体积和一本教科书差不多大，下面是详细的性能介绍：

1） CPU 类型 赛扬-M

2） 最高主频 900MHz

3） 主板描述 Intel 855GME

4） 笔记本重量 920g

5） 内存大小 512MB

6） 内存类型 DDRII

7） 硬盘描述 SDD 闪存硬盘

8） 标准接口 3 个 USB、显示输出、耳机、SD 卡（兼容 SDHC），Kensington 锁等接口或（13） 插槽

9） 电能规格电池类型 4 芯 5200 毫安电池

10） 供电时间 3.5 小时

图 3.23 视觉模块使用的笔记本电脑

图 3.24 视觉模块使用的摄像头

3.6.2 摄像头和镜头的选择

比赛中，机器人的视野越大越好，作者在选择镜头的过程中发现，广角镜头不仅价格贵出不少，而且会使摄像头色彩失真，导致后续的图像处理效果变差。在找不到更好的摄像头和镜头的条件下，作者只好以色彩失真小为首要考虑因素，放弃广角镜头，在摄像头的选择上，我们没有很高的要求，选择了清华紫光的普通网络摄像头。

3.6.3 视觉模块和主控的通信

出于方便的考虑，视觉模块通过串口不停地向主控发送一个以 0xff 开头的包，包含了白块和障碍的位置信息，以及二次定位的信息。主控在进入视觉模式时，读取视觉模块传送的数据，一般模式下不会处理视觉信息。毕竟视觉信息可靠性无法完全保证。

3.7 本章小结

本章论述自动机器人整体硬件设计和各种元气件选型，包括电源、硬件上各个模块的相互通信、底层数据的采集。根据设计需要规划了单片机内部资源的分配和相应外围电路。在整个系统中，陀螺仪是最精密的环节，对 A/D 芯片的选型、外围运放和电阻的选型都有很高的要求，经过大家的充分论证，合理地对硬件电路进行了设计，使系统在稳定性和高效性方面有了提升。

4 基于多传感器的自主移动机器人全场定位算法及实现

4.1 机器人定位理论和方法

1） 基于信任度的定位

在信任度表示系统的中，最重要的分支是单架设和多假设信任度系统。位置的单假设表示方法的主要优点是给定唯一的信任度，位置没有任意性。但是，实际上由于传感器的噪声，位置更新过程总是产生位置的单个假设常常是不可能的。而多假设信任状态的策略是将信任度建模成数学分布，比如高斯分布。这个表示方法特别适合数学上所定义的跟踪函数，比如卡尔曼滤波器。如果机器人仅从它的传感器和执行器获得关于位置的信息，则其信息可以合并为信任度。该方法对于一类定位和导航求解是关键的，在这类问题中，机器人不仅要对达到特定目标进行推理，还要对自己信任度状态的未来路径进行推理。譬如，机器人要选择一条使它未来位置不确定度最小的路径。但这也需要多假设表示方法带来的决策问题。

2） 基于概率地图的定位[9] [10]

（1） 马尔可夫定位

该方法对所有可能的机器人位置使用明确指定的概率分布，考虑了从任何未知位置开始的定位，因而可以恢复不明确的地方，因为机器人可以完全跟踪多个完全不同的位置。然而，要在任何时刻，在整个状态空间内更新所有位置的概率，就需要空间的离散表示，比如几何栅格或者拓补，所以，其所要求的存储器和计算功能可能限制了精度和地图的尺寸[10]。

（2） 卡尔曼滤波器定位

该方法从起始的已知位置开始跟踪机器人，具有固定和精确的有效性质，特别是在连续的环境表示。然而，如果机器人的不确定度变得非常大（比如碰撞），因而造成非真正的单模，卡尔曼滤波器就不能捕获多个可能的位置，造成无法挽回的损失[10]。

4.2 比赛机器人的全场定位方法

4.2.1 比赛机器人的全场定位的传感器数据采集

底层数据采集单元负责采集陀螺仪和从动码盘的信息，并处理成角度和里程，再定时向主控单元发送，定时采用定时器查询方式。底层数据采集单元的流程如图 4.1 所示：

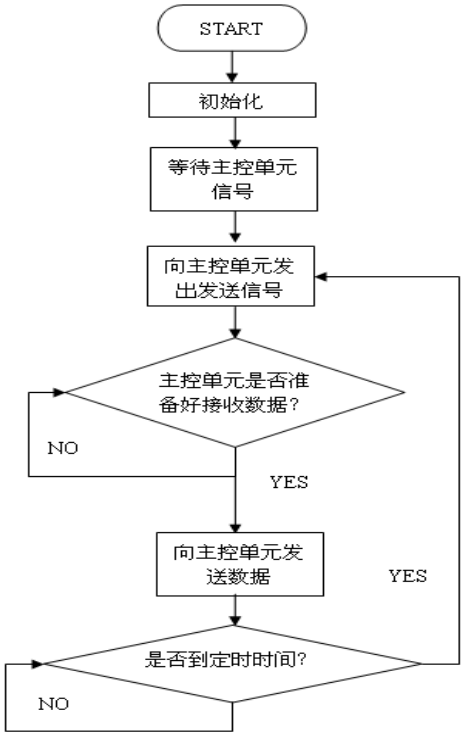


图 4.1 底层数据采集单元流程图

底层采集用两个计数器来记录两个从动码盘的脉冲，并将其经过鉴相电路得到移动方向，将方向接到底层 LPC2131 的外部中断，当码盘的移动方向变化时会产生中断，记录下当前计数器的值并将计数器清零，从而保证不错记码盘脉冲。

底层采集单元的 LPC2131 初始化完成后，会等待主控单元发送主控单元准备完毕的信号。收到信号后进入检测计数和发送数据的循环。在一次循环中，底层的 LPC2131 先将连接到主控 LPC2138 外部中断的输出口置高，使主控单元产生中断接收底层发送的数据，包括角度值和里程值。发送完成后再将连接到主控 LPC2138 外部中断的输出口置零，等待计时器到时间进行下一次传送。

4.2.2 比赛机器人的全场定位的算法实现

全场定位系统的原理和实现 FLS（全场定位）需要陀螺仪、码盘、接触开关以及视觉多个传感器联合工作。陀螺仪和编码盘的定位虽然比较准确，但陀螺仪的漂移和轮径及安装方面的几何尺寸误差等原因，使得累积误差难以消除，这样就会导致机器人行走时间越长，坐标的误差就越大，到后来就不能依靠这个坐标了；所以利用本届桥柱较多，完全可以使用这些已经存在的固定点来校正有陀螺仪和码盘得到的坐标。校正方法虽然可以精确确定机器人与桥柱中心的距离关系，但是不能辨别机器人当前的角度信息。所以要让定位系统非常准确，就必须利用信息融合技术，把两者结合应用，依靠优势互补，在陀螺仪和编码盘得到的连续位置信息基础上加以接触开关的校正，使得系统能精确、实时的定位。但该定位方法是以陀螺仪角度信息是准确的作为基础条件的。

定位的目标是得出机器人在比赛场地这个平面上的二维坐标。使用微元累加的思想，把机器人的行动曲线看成是很多段微小的直线组成的，就有可能解决定位问题。

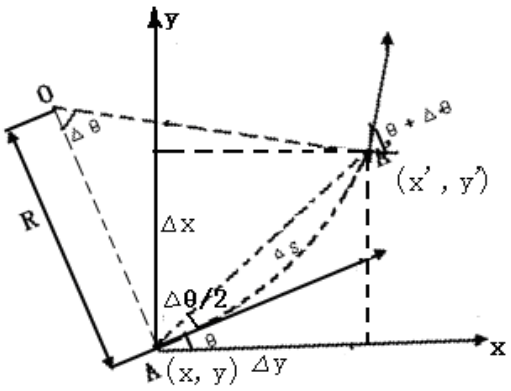


图 4.2　机器人运动位置坐标的计算

图4.2 表示机器人沿弧线从A（ x, y, θ ）点走到A′（ $x+\Delta x, y+\Delta y, \theta+\Delta\theta$ ）的坐标变化。 $\Delta x, \Delta y, \Delta\theta$ 分别表示在Δt时间内机器人的横、纵坐标和角度的增加量， (x, y) 是图4-1中建立的直角坐标系中的机器人坐标， θ 则表示以横轴为起始位置并以逆时针方向为正的方向角。ΔS表示A 点到A ′点的弧长，R表示圆弧半径。于是 $\Delta x, \Delta y$ 可由如下的公式计算得到：

$$\Delta x = AA' \bullet \cos(\theta + \frac{\Delta\theta}{2})$$

$$= 2R \bullet \sin(\frac{\Delta\theta}{2}) \bullet \cos(\theta + \frac{\Delta\theta}{2})$$

$$= \Delta S \bullet [\sin(\frac{\Delta\theta}{2}) \Big/ \frac{\Delta\theta}{2}] \bullet \cos(\theta + \frac{\Delta\theta}{2}) \tag{4.1}$$

$$\Delta y = AA' \bullet \sin(\theta + \frac{\Delta\theta}{2})$$

$$= 2R \bullet \sin(\frac{\Delta\theta}{2}) \bullet \sin(\theta + \frac{\Delta\theta}{2})$$

$$= \Delta S \bullet [\sin(\frac{\Delta\theta}{2}) \Big/ \frac{\Delta\theta}{2}] \bullet \sin(\theta + \frac{\Delta\theta}{2}) \tag{4.2}$$

当机器人走直线时$\Delta\theta$=0，$\sin(\frac{\Delta\theta}{2}) / \frac{\Delta\theta}{2} \to 1$，则式（4.1）,（4.2）可写成：

$$\Delta x = \Delta S \bullet \cos(\theta + \frac{\Delta\theta}{2}) \tag{4.3}$$

$$\Delta y = \Delta S \bullet \sin(\theta + \frac{\Delta\theta}{2}) \tag{4.4}$$

由于时间间隔Δt很短，线段AA’的长度近似等于圆弧长度，设Δt时间内编码盘走过的距离为delta_len，则$\Delta S \approx$ delta_len。设last_angle为上次采样的角度， rob_angle为当前的角度，通过计算Δt时间内机器人的位置变化量$\Delta x, \Delta y$，进行累加，从而求出机器人在整个赛场上的位置。

$$x' = x + delta\_len \bullet \cos(\theta + \frac{\Delta\theta}{2})$$

$$= x + delta\_len \bullet \cos(\frac{\theta + (\theta + \Delta\theta)}{2})$$

$$= x + delta\_len \bullet \cos(\frac{last\_angle + rob\_angle}{2}) \tag{4.5}$$

$$y' = y + delta\_len \bullet \sin(\theta + \frac{\Delta\theta}{2})$$

$$= y + delta\_len \bullet \sin(\frac{\theta + (\theta + \Delta\theta)}{2})$$

$$= y + delta\_len \bullet \sin(\frac{last\_angle + rob\_angle}{2}) \tag{4.6}$$

这样，在初始坐标（x,y）的基础上，每个采样周期（程序循环一次的时间，大约6ms）都计算一次更新坐标(x', y')，这个(x', y')相当于下一个周期的初始坐标（x,y），如此循环，就可以时刻知道机器人所在的位置坐标。

经过测试，实际运行情况与该理想模型十分接近。但是随着运行路程的增长，累计误差越来越大，因此需要用其他的手段进行位置校正。由于章节安排的关系，本文在视觉部分会详细论述借助视觉传感器进行二次定位的过程。

4.3 多传感器融合的全场定位算法的研究

在往届机器人大赛中，各队全场定位的方法主要分为两种，一种就是陀螺仪和里程计的定位方法，还有一种就是我校原来一直在用的循白线定位方法。下面具体分析一下这两种方法的特点和局限。

4.3.1 陀螺仪和里程计定位的特点及其局限

陀螺仪和里程计能够实时直接返回从上电开始机器人已走的里程和机器人的角速度，经过计算可以得到每个坐标更新周期机器人坐标的相对偏移量，从而得到机器人的坐标信息。

经过反复的测试和电路以及程序的不断改进，陀螺仪返回的角度是非常精确的，可以认为角度没有误差。但是，码盘所计的里程信息则受限于码盘安装精度和机器人的移动方式，无法始终保持里程信息的精确；另一个引起定位不准的主要原因是由于目前机

器人软硬件系统的限制，机器人坐标的更新周期为 10ms，周期太长，导致现有的数学模型不能很好地估计出一个周期内机器人的坐标偏移。

上面这些原因，导致了基本的陀螺仪加里程计实现的全场定位算法的局限性——定位精度随着机器人移动距离和速度的增加误差累积越来越达，最后到了不可接受的地步500mm。

4.3.2 光纤沿白线定位的特点及其局限

当机器人的移动路径都是沿着场地上的白线时（见图 4.3），机器人的定位会方便很多，每个交叉点是机器人坐标的最小单位，依靠白线信息，机器人就可以实现低分辨率（白线交点为最小单位）的定位。而白线信息的检测，可依靠光电传感器轻松实现。

尽管这种定位方法十分简单可靠，但是，机器人不得不被限制只能在白线上移动，活动空间大大受限，使得陀螺仪不能充分发挥其巨大威力。

4.3.3 陀螺仪里程计结合光纤定位的特点及优势

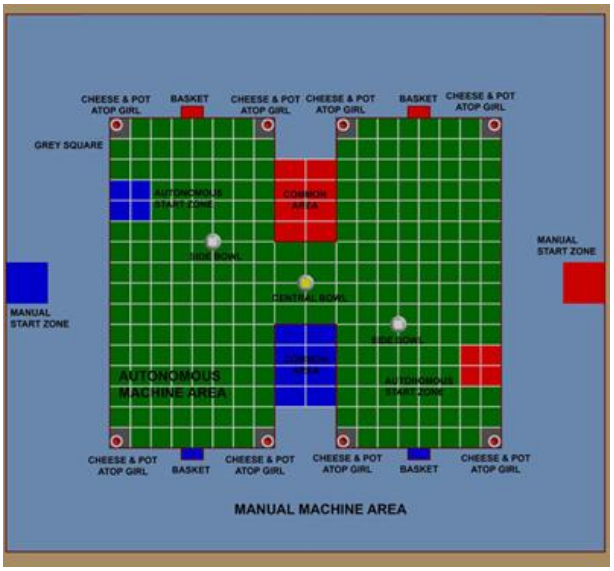


图 4.3 第七届机器人大赛比赛场地图

由于场地上的白线是非常好的坐标参照系（见图 4.3），每条白线间的间距是 500 mm。机器人在每到一条白线上时，完全可以在原有当前坐标的基础上对坐标进行进一步更新。比如说，原来计算出来的坐标是（2595,-1581）,但此时机器人恰好经过白线的交叉点，则我们可以推断出该交叉点坐标必定是（2500,-1500）,从而反过来更新纠正陀

螺仪里程计计算出的坐标。作者将根据比赛场地是白线组成的方格的特点，开发辅助定位的算法。其基本思想是结合陀螺仪和里程计实时计算出的机器人坐标并利用场地上白线的位置信息，加一个或多个额外的光电传感器，实现多传感器融合的定位纠偏算法。本文作者和另一名队员一起研究了坐标更新的方法，介绍如下。

4.3.4 单一依据的辅助定位算法的原理与实现

最初，我们的校正的基本思想很简单，检测地上的白线，经过横线时更新 Y 坐标；经过竖线时，更新 X 坐标。有了这个基本思路后，接下来就是如何在算法上实现了。一开始，作者在从动轮码盘附近安装了一个光电传感器（因为从动轮码盘是机器人的旋转中心，坐标也是以它的来定的），这样，从动轮经过白线时就能够被传感器检测到。先做一个大胆的假设，假设陀螺仪加码盘的坐标定位在一定范围内偏差不会很大。当经过白线时，判断当前横纵坐标，并由当前坐标分别判断离当前坐标较近的横线和竖线的坐标，分别计算当前横纵坐标到横线竖线的距离，哪个距离近（说明当前白线就是近的那根线），就更新哪个坐标。例如，当前坐标为（1985，2350），显然离当前点近的白线应该是 X= 2000，所以，将机器人的坐标更新为（2000，2350）。这对于非交叉点的白线是可行的。考虑到以上算法在循线过程中，一直更新自己所循的线，而没有更新与之相垂直的那个坐标。而正确的做法应该是在交叉点更新另外一个坐标。因而，引入了“小圈”的概念。小圈的定义很简单，当横纵坐标离横竖线的距离都在一定范围内（取 3CM），就可以认为检测到的白线为交叉点，这时强制更新与车当前角度相垂直的坐标。这样，又引入了一个“小圈死循环”。假设循线走进入小圈后，更新了垂直方向的坐标，这样每次更新完，到下一周期时，还在小圈范围内，又被更新回小圈中心，进入了一个死循环，坐标必然错误。因而在小圈时进行了特殊处理，对进入小圈采用边沿触发方式，即进入一次白线，只更新一次坐标。一旦出了小圈范围，则更新近的坐标。

按照以上算法，作者和另一名队员进行了测试。测试结果见表 4.2。可以看出，X 值普遍存在减小的趋势，这是由于场地白线不精确，对初始摆放角度造成偏差，导致 X 一直在减小。以上的做法理论上应该没有什么问题了，然而在进行第 6 组测试时，出现了错误。考虑机器人与线平行走，但不在线上，这时，理论上只会更新垂直方向上的坐标。实际上是错误的。简单的说，因为垂直方向的坐标比当前方向的坐标离白线近，所

以一开始几次肯定是更新垂直方向的坐标。然而，平行方向的坐标误差越积越大，超出了正常范围，进入了另外一条白线，这时计算出来的距离可能会变的很小，以至于将横竖线混淆，进行了错误的更新。为了解决这个问题，考虑在车上装 3 个传感器，每个传感器都按以上算法计算自己的坐标，并映射到车的旋转中心处。一旦一个坐标与两外两个坐标发生了分歧，那么就以另外两个坐标为准，更新出错的坐标。这样能在一定的程度上减小这个错误发生的概率。但毕竟机器人宽度有限，不可能保证一直都与横线、竖线有交叉，所以这种错误还是有可能发生的。由于检测到一次白线 X 和 Y 都有可能更新，因而容错量为 125mm。

表 4.1 单一依据的坐标纠偏的算法思想阐述

区域	坐标更新方式	一次更新造成的负面误差（最坏的情况）单位：mm
Out 区域	按距离更新	< 200（ReloadDistance）
Mid 区域 1（上次坐标在以白线交汇点为圆心的小圈内）	强制更新，与前次方向平行交替更新，垂直则按原方向更新	<60（OutsideR）
Mid 区域 2（上次坐标在以白线交汇点为圆心的小圈外）	按距离更新	<60（OutsideR）
In 区域 1（上次也是在小圈内部）	不要更新坐标，避免更新方向的坐标被始终不停更新为同一个值	0
In 区域 2（上次不是在小圈内部）	表示第一次进入小圈，在小圈内需要且仅需要更新一次； 在小圈内的状态就是车经过交汇点，我们目前按照车的角度，更新与车方向垂直方向的坐标（当车的角度不平行或者垂直的时候，就不更新了）	<30 或者 0
Noisy 区域	不更新坐标（去躁）	0

表 4.2 单一依据的辅助定位算法的第一次测试的 6 组数据

白线坐标	经过更新的机器人坐标（单位:mm）					
	1	2	3	4	5	6
（250,4250）	（248,4250）	（247,4250）	（253,4250）	（254,4250）	（251,4250）	（249,4250）
（250,3750）	（246,3750）	（246,3750）	（246,3750）	（256,3750）	（251,3750）	（244,3750）
（250,3250）	（244,3250）	（244,3250）	（240,3250）	（250,3250）	（244,3250）	（240,3250）
（250,2750）	（244,2750）	（243,2750）	（243,2750）	（239,2750）	（243,2750）	（241,2750）
（250,2250）	（242,2250）	（241,2250）	（240,2250）	（243,2250）	（241,2250）	（236,2250）
（250,1750）	（238,1750）	（241,1750）	（236,1750）	（236,1750）	（236,1750）	（230,1750）
（250,1250）	（235,1250）	（239,1250）	（235,1250）	（234,1250）	（235,1250）	（228,1250）
（250,750）	（229,750）	（237,750）	（227,750）	（234,750）	（234,750）	（226,750）
（250, 0）	（227, 0）	（238, 0）	（226, 0）	（233, 0）	（232, 0）	（225, 0）
（250,750）	（223,750）	（235,750）	（224,750）	（229,750）	（229,750）	（223,750）
（250,1250）	（222,1250）	（230,1250）	（228, 250）	（231,1250）	（229,1250）	（219,1250）
（250,1750）	（215,1750）	（228,1750）	（226, 750）	（229,1750）	（226,1750）	（215,1750）
（250,2250）	（215,2250）	（231,2250）	（223, 250）	（226,2250）	（228,2250）	（211,2250）
（250,2750）	（213,2750）	（226,2750）	（220,2750）	（223,2750）	（225,2750）	（250,2733）
（250,3250）	（211,3250）	（228,3250）	（218,3250）	（221,3250）	（223,3250）	（250,3124）
（250,3750）	（211,3750）	（223,3750）	（223,3750）	（220,3750）	（223,3750）	（250,3697）
（250,4250）	（210,4250）	（220,4250）	（216,4250）	（223,4250）	（223,4250）	（250,4013）

4.3.5 三角更新算法的原理与实现

考虑到单一依据的辅助定位算法理论上的固有缺陷和偏低的容错量，原来与作者合作的另一名队员后期提出了三角更新算法，由于性能有了质的飞跃，并且作者也参与了部分的工作，有必要作进一步的介绍。

三角更新算法最大的特点，也是最巧妙的地方，在于当机器人检测到白线后，能够准确地分辨所检测到的白线是横线、竖线，还是横竖线的交叉点。如此，机器人若检测到横线，则更新 Y 坐标；检测到竖线，则更新 X 坐标；检测到交叉点，则更新 X、Y 坐标。如此，容错量可以达到 250mm。

1） 三角更新算法的原理

如图 4.4 所示，等边三角形的三个顶点分别代表三个光纤传感器，1 代表车头所在方位，2 代表左轮所在方位，3 代表右轮所在方位。将[0，360）以 30 °为一个区间分为 12 个区间，表 5.6 列出了机器人以不同的方向前进时进入横、竖线的不同情况。

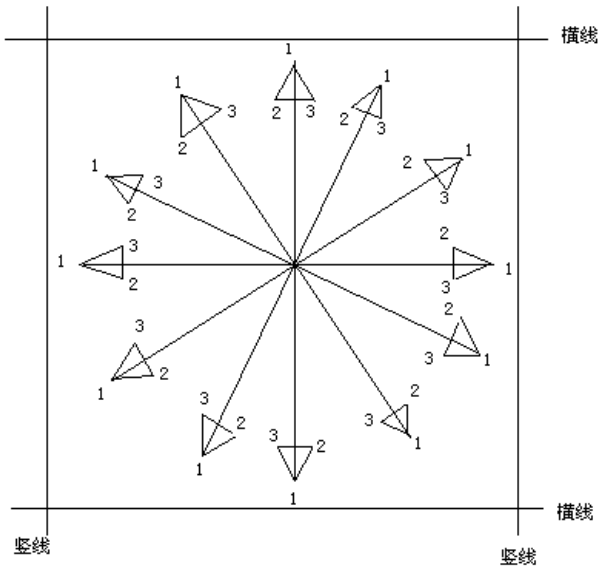


图 4.4　三角更新传感器示意图

在判断三角形传感器进入横线时，根据当前机器人角度，判断三角形中与横线夹角最小的边上的两个传感器是否在白线上，来确定是否进入横线，只要两个传感器中有一个不在线，则认为不在横线上；同理，在判断三角形传感器进入竖线时，根据当前机器人角度，判断三角形中与竖线夹角最小的边上的两个传感器是否在白线上，来确定是否进入竖线，只要两个传感器中有一个不在线，则认为不在竖线上。

由此，我们得出了三角更新中两个最关键的节点表：

```
uint16   horizontalTable[12] = {12, 12, 23, 23, 13, 13, 12, 12, 23, 23, 13, 13};
uint16   verticalTable[12] = { 23, 13, 13, 12, 12, 23, 23, 13, 13, 12, 12, 23};
```

我们首先将机器人当前角度规范到[0,360）范围内。那么，只要判断当前传感器状态与当前角度映射到表里的边之间的关系，就能够判断机器人当前是否在线。每当判断出机器人进入横线、竖线，或者交叉点，依据当前机器人的坐标，再找到离当前坐标最近的横线、竖线，或者交叉点的坐标，并更新当前坐标。这就是三角更新的具体实现过程。

表 4.3 三角更新传感器进线顺序表

角度\进线顺序	横线	竖线
[0,30）	2,1	3,2
[30,60）	1,2	1,3
[60,90）	2,3	3,1
[90,120）	3,2	2,1
[120,150）	1,3	1,2
[150,180）	3,1	2,3
[180,210）	2,1	3,2
[210,240）	1,2	1,3
[240,270）	2,3	3,1
[270,300）	3,2	2,1
[300,330）	1,3	1,2
[330,360）	3,1	2,3

在如何判断当前传感器状态与表中数据的关系上，引入了两个加权表和一个边表：

```
uint16  horizontalNodTable[12] = { 3,  3,  6,  6,  5,  5,  3,  3,  6,  6,  5,  5};
uint16  verticalNodTable[12]   = { 6,  5,  5,  3,  3,  6,  6,  5,  5,  3,  3,  6};
uint16  edgeTable[8] = {300, 300, 300, 12, 300, 13, 23, 123};
```

将三个传感器进行加权，1、2、3 传感器的权值对应为 1、2、4（符合二进制规范），那么，就能根据节点表得到加权表，再依据加权表中的数值，映射到边表中，空余的为 300（任意给的，只需大于 123），对应的数据为对应边，就得到了边表。

2） 三角形传感器边长的确定

依据三角传感器的工作原理，在宽度为 D 的直线上（节点除外），三个传感器中，至多有两个传感器同时在线，因而对边长有最小限制；同时，必须保证三角形在 30°范围内，其中的两个传感器都能在线，因而对边长 L 有最大限制（见图 4.5）。

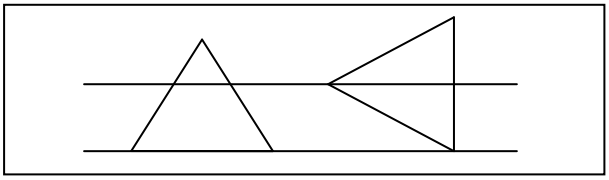


图 4.5 三角更新传感器尺寸限制示意图

$\frac{\sqrt{3}}{2}L \geq D$　Lmin = 34.64mm

$\frac{1}{2}L \leq D$　Lmax = 60mm

$\bar{L}$ = （ Lmin + Lmax ） / 2 = 47.32mm

理论上最合适的边长应该是最大、最小限制的平均值 47.32mm，但实际应用中，由于传感器具有发散性，经过试验，当边长为 50mm 时，效果最优。

3） 三角更新算法的试验及结果

为了保证这项技术的可靠性，作者和另一名队员做了大量的实验测试，实现结果见表 4.4，可以看到，三角更新效果非常明显，横纵坐标误差都在 30mm 以内。而如果单独靠陀螺仪和码盘而不进行坐标更新，误差往往会随速度的提升越来越大，最大误差达到 485mm。

表 4.4 三角更新试验数据表

	运行速度	运行时间	X 偏差	Y 偏差
1	0.5m/s	1min	9mm	7mm
2	1m/s	1min	8mm	16mm
3	1.5m/s	1min	12mm	13mm
4	2m/s	1min	14mm	9mm
5	2.5m/s	1min	18mm	11mm
6	3m/s	1min	18mm	19mm

表 4.5 坐标不进行更新的试验数据表

	运行速度	运行时间	X 偏差	Y 偏差
1	0.5m/s	1min	79mm	57mm
2	1m/s	1min	78mm	96mm
3	1.5m/s	1min	132mm	141mm
4	2m/s	1min	245mm	149mm
5	2.5m/s	1min	358mm	371mm
6	3m/s	1min	388mm	485mm

4.4 本章小结

本章着重论述自动机器人利用多传感器信息融合进行全场定位的方法与实现。并通过大量的试验和各种方案的比较，最终找到了一种很好的坐标更新方法，突破了机器人的全场定位这一关键技术，为以后的各种任务调度和上层算法以及行走控制奠定了坚实的基础。

5 自主移动机器人轨迹跟踪控制算法及实现

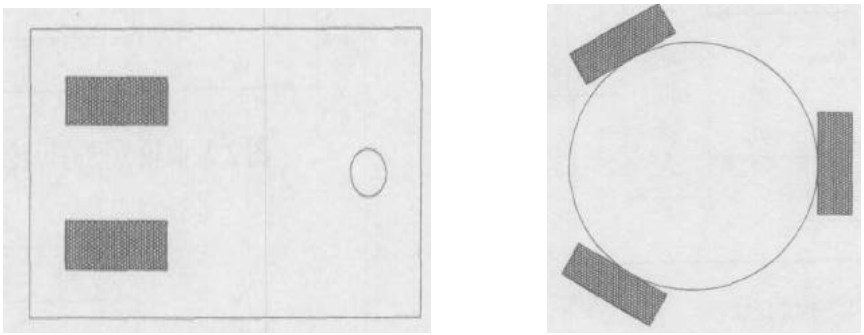
5.1 移动机器人的控制问题综述

1） 非完整约束和非完整约束机器人

完整约束只限制受控对象的空间位置，或者同时限制空间位置及运动速度但经过积分后可转化为只对空间位置的约束，因此称其为几何约束。具有完整约束的系统称为完整系统。由于可以通过积分和非线性变换从约束条件中解出若干个状态变量[11]，进而可将原系统转化为一个低维系统，所以此类系统的分析与综合问题与无约束系统相比而言没有太大的困难，在理论与应用研究方面已取得满意的进展[12][13] 。

非完整约束则是同时限制空间位置和运动速度，并且不能通过积分转化为空间位置的约束，简单地说就是不可积约束或运动约束，如轮子和地面之间的滚动约束、某些关节无驱动或者驱动器失灵等情形。相应地我们称具有非完整约束的系统为非完整系统。对于该类系统，由于约束是不可积的，不存在相应的降维变换，使系统呈现一些复杂特性，如不能实现输入一状态线性化、不能采用光滑非线性反馈实现渐近稳定等，非线性控制中的一些有效方法也不再适用于该类系统，因而非完整性使机器人的控制问题变得相当困难。但是，非完整性使机器人本身的结构具有更高的灵活性和可靠性，它允许机器人的某些关节无驱动，可以大大降低机器人的制造成本、重量和能耗，提高了机器人的灵活性。因此，非完整机器人在国防和工业上具有很高的应用价值，近年来已受到国内外学术界、研究机构和工业界的广泛重视。

从 1980 年代末起, 由于机器人及车辆控制的需要, 使得国外开始对非完整系统的控制问题进行深入研究。由于非完整约束是对系统广义坐标导数的约束, 它不减少系统的位形自由度, 这使得系统的独立控制个数少于系统的位形自由度, 给其控制设计带来很大困难。另外，利用非线性控制系统理论的微分几何方法已证明: 非完整系统不能用连续的状态反馈镇定。因此以研究连续状态反馈为主的现代控制理论中大量成熟的结果无法直接用于非完整系统的镇定控制研究, 使得非完整控制系统研究成为当今控制领域最具挑战性的难题之一[14][15]。



（a）　　（b）

图 5.1 两种不同底盘结构的轮式机器人

图 5.1（a）是两轮差动驱动的移动机器人的结构模型，在模型中左右两轮由两个电机分别驱动，另一个是可以自由转动的角轮，该结构受到非完整约束，故属于非完整移动机器人[16]。图 5.1（b）是全向移动机器人的结构模型，在模型中三个车轮在移动机器人底座上均匀分布，互成 120 度，分别由三个电机驱动,该结构机器人由于受到完整约束，故属于完整移动机器人[17]。

按照上面的说明，我们比赛机器人属于非完整移动机器人。

2） 相关的控制问题和方法

镇定非完整系统的非线性方法，如非连续镇定控制律、时变镇定控制律、混合镇定控制律等。但这些方法是在全局坐标系下研究非完整约束系统的，也有学者在极坐标系下讨论和研究非完整约束系统的镇定问题。在极坐标系下研究轮式移动机器人的特点是纯滚动无滑动这个约束不显式存在，可得一种全局渐近镇定控制律，从而使平衡点的渐近镇定可以通过光滑定常状态的反馈控制律来实现。滑模控制不需要知道被控对象精确的数学模型。当外界扰动和参数摄动满足匹配条件时系统滑模运动对它们有完全的自适应性或称为不变性。所以可以将具有优良的鲁棒性的滑模控制应用于移动机器人的控制中。智能控制由于模糊算法及神经网络算法所具有的学习能力和非线性映射能力，为解决机器人控制提供了新的手段。

移动机器人的控制问题包括以下三个方面即点镇定问题、轨迹跟踪问题和路径跟踪问题 [18]。

（1） 点镇定问题要求移动机器人从任意的初始状态镇定到任意的终止状态，其目的是获得一个反馈控制律，该控制律可使整个闭环系统的一个平衡点是渐近稳定的。轮式移动机器人由于做纯滚动无滑动运动，因此受到非完整约束，移动特性受到限制，并且非完整控制系统不具有孤立的平衡点而导致系统的平衡点不能被局部渐近镇定。

（2） 轨迹跟踪也称动态跟踪。要求移动机器人跟踪与时间呈函数关系的路径。

（3） 路径跟踪，要求移动机器人以给定的速度或加速度沿着给定的与时间无关的路径运动。在移动机器人的路径跟随问题上，一般地，会选取两个输入量中的前进量为任意的常量或时变量，而另一个输入量用作控制量[18]。

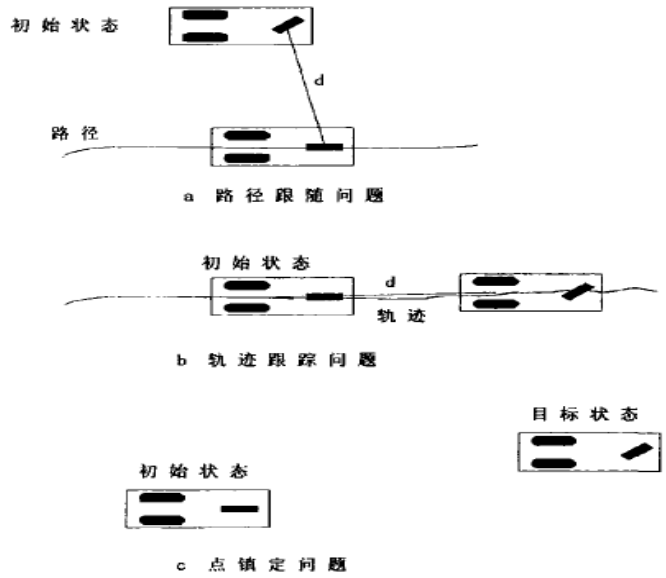


图 5.2 三种机器人系统的控制问题

点镇定问题实际上是一个输入输出的问题，它用 m=2 个输入去控制 n=3 个被控状态，由于它试图由两个输入去控制多个独立的变量，因此点镇定问题是最难解决的。路径跟随问题是用 m=1 个输入去控制 p=1 个输出，而轨迹跟踪问题是由 m=2 个输入去控制 p=2 个输出，这两个问题都是输入一输出的，它们对系统状态的控制是间接的，并且控制输入和被控制变量的个数是一致的。

按照上面的分类，我们比赛中的控制问题应该属于第 2 种，轨迹跟踪问题。

5.2 机器人轨迹跟踪算法

5.2.1 经典 PID 算法

由于在实际的程序实现中，我们使用的是经典 PID 控制，所以首先对经典 PID 算法进行简单的介绍。

在机器人的控制中，由于要求机器人对自己能很有效的控制，对直流电机的控制的转速和位置，有较高的要求，有高精度的电机控制，机器人才可以知道自己的位置，知道自己的行走路线。

直流电机是一个典型的滞后时间很小的二阶惯性环节，有时也可以近似的看作一阶惯性环节。理论和实践均证明了，对于这种被控对象，PID 是一种较好的控制算法。

PID 的基本算法是：控制器的输出是

1） 与控制器的输入（误差）成正比。

2） 与输入的积分成正比。

3） 与输入的导数成正比的三分量的和。连续系统的 PID 控制的表达式为：

$$U = Kp(e + \frac{1}{T_I}\int edt + T_D\frac{de}{dt}) \tag{5.1}$$

式(5.1)中：

e——测量值与给定值的偏差

T_D——微分时间

T_I——积分之间

Kp——调节器的放大系数，$\delta = \frac{1}{Kp}$ 又叫调节器的比例带

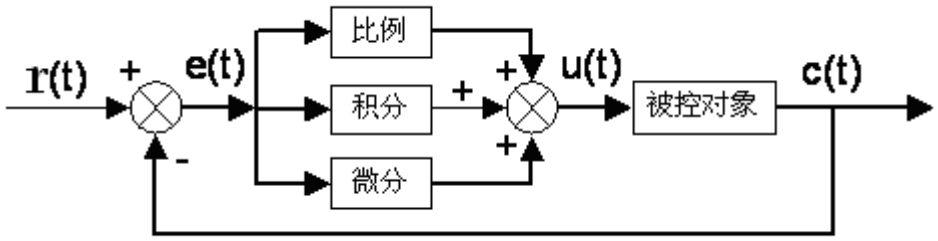


图 5.3 经典的连续系统 PID 控制器

PID 控制也称比例——积分——微分控制。一般来说，比例环节是用来放大误差，来纠正偏差的；积分环节用来消除系统的稳态误差；微分环节用来提高系统的快速性。PID 调节器的性能就取决于 Kp ， T_I ， T_D 这三个系数，设计和调试者的任务也就是这三个参数。

在连续系统中的积分项和微分项很难用计算机上实现，要用计算机实现，就要将这样的系统转化为差分方程，由此实现数字 PID 调节器。

1） 经典位置式 PID 算法

用矩形法数值积分代替上式中的积分项，对导数项用后向差分逼进，得到数字 PID 控制器的基本算式（位置算式）：

$$u_n = K_P(e_{n-1} + \frac{1}{T_I}\sum_{k=1}^{n} e_k T + T_D \frac{e_n - e_{n-1}}{T}) \tag{5.2}$$

其中 T 是采样时间。

2） 经典增量式 PID 算法

对位置式算式加以变换，可以得到 PID 调节算法的另一种实用形式（增量算式）：

$$\Delta u_n = u_n - u_{n-1} = K_P[(e_n - e_{n-1}) + \frac{1}{T_I}e_n + \frac{T_D}{T}(e_n - 2e_{n-1} + e_{n-2})] \tag{5.3}$$

$A = K_P$， $B = K_P\frac{1}{T_I}$， $C = K_P\frac{T_D}{T}$,可以得到一个方程，这个方程，经常用来在计算机上做逻辑运算。

这种算法用来控制步进电机特别方便，对直流电机的控制也可以采用。

正如我们的控制问题是控制机器人跟踪一条规划出来的固定路径，我们可以设定机器人的移动速度为比赛需要的速度，而将两轮的差速作为机器人跟踪控制的输入量。

5.2.2 直线跟踪算法

由动作的起始点坐标和目标点坐标可以确定坐标平面上的一条直线，任意两点间的直线行走函数设计的目标，就是通过 PID 校正使机器人从动作起始时的位置，沿着由起始点指向目标点的直线，走到目标点。对于直线行走的 PID 校正，误差的形式有两种：角度误差和中心位置误差。如图 5.4 所示：

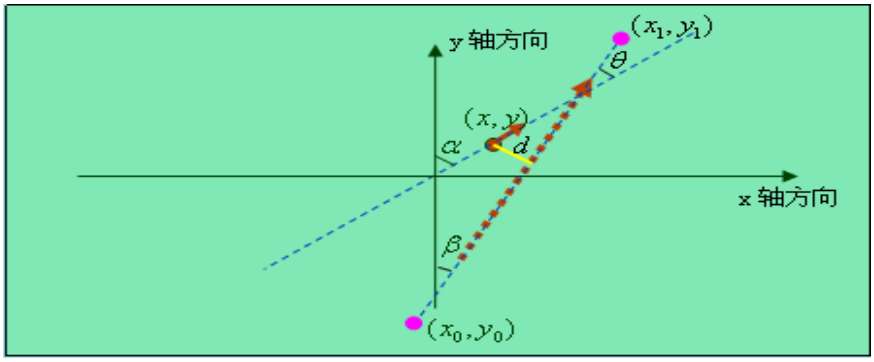


图 5.4　直线动作角度误差与中心位置误差的计算

其中，α 表示机器人当前的方向；β 表示由动作起始点到动作目标点的直线在坐标系中的角度，其角度约定与 FLS 角度约定相同；θ 表示角度误差；d 表示机器人中心位置误差，在白线左边为负，在白线右边为正。由平面几何关系即可求得 θ 与 d 如下：

$$\theta = \alpha - \beta \tag{5.4}$$

$$d = (x - x_0) \times \cos\beta - (y - y_0) \times \sin\beta \tag{5.5}$$

这是两个相对独立的误差，可以通过加权将它们合成一个误差，再用一套 PID 参数对其进行校正，如下式：

$$e = d + k \bullet \theta \tag{5.6}$$

加权系数 k 反映了机器人角度误差与位置误差相消或相涨的程度，它实际上包含了一定的微分的意义。

对于综合后的误差 e，我们采用增量式 PID 算法对其进行校正,函数实现为：

int16 PIDControl（ fp32 CtrlError , uint16 MotorSpeed）

而经过实际测试，当机器人处于开环状态时，在 delta 时间内角度偏差引起的位置偏差。得出角度偏差和位置偏差的加权系数为：3，在实际调试时也印证了这个权值是合适的。

在任意两点间的直线行走函数是在给定起始点，目标点坐标，起始速度，最高速度，末速度，加减速距离后，该函数会自动根据给定的条件选择是否需要首先原地转过

一定的角度，使机器人的方向与欲行走直线的方向相同。然后根据上面所说的增量式PID 校正算法对机器人的速度进行实时地校正。实现沿直线走。

上述直线跟踪算法在程序里由以下函数实现:

int32 GoStraightLine（struct CPoint startPoint,struct CPoint endPoint）

5.2.3 圆弧跟踪算法

为了实现行走过程的流线衔接，在需要转弯时不减速，或者只是将车的速度减慢一点，实现快速转弯，在转弯后，可以以较高的速度进入到下一个行走直线函数。快速转弧函数的思想为：在高速情况下，在需要转过的角度较大时，逐渐增大两轮的差速，并在距离目标角度还有一定的误差时，减小两轮的差速，可以实现当角度接近目标角度时，两轮的速度差很小。减小函数跳转间车的震荡。实现平稳的过渡。

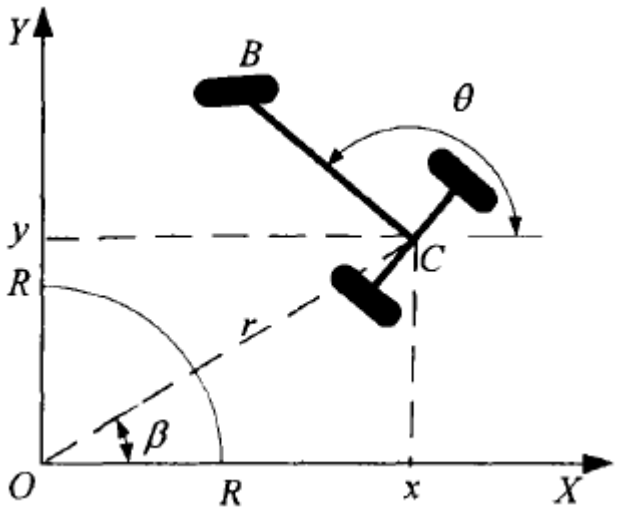


图 5.5 机器人小车跟踪圆的情形

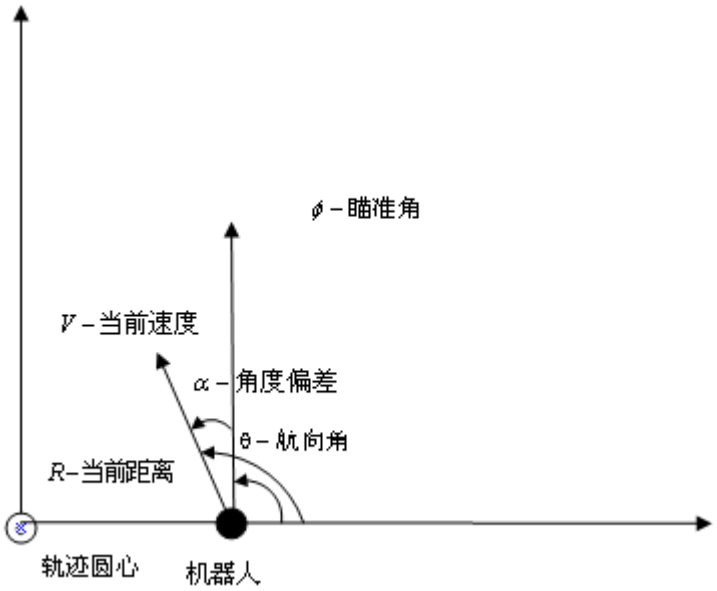


图 5.6 机器人小车以半径 r 作圆弧运动时的状态偏差

若不考虑航向角（机器人当前姿势的角度），只取两个状态变量角度偏差和与圆心的距离偏差，利用 Lyapunov 函数，建立轮式机器人数学模型，可以找到合适的控制律，在不同的极点配置下，实现状态变量角度偏差和距离偏差的镇定，不能镇定机器人的航向角。

在状态变量中同时考虑当前角度偏差，与圆心的距离偏差及航向角，利用 Lyapunov 函数，建立轮式机器人数学模型，可以找到合适的控制律，在不同的极点配置下，实现系统的航向角、距离偏差和角度偏差的镇定。

实际机器人调试中，在圆弧路径 PID 控制里，算法的参考目标是转弯半径 R 和机器人实时的转弯角度α,类似直线跟踪的思想，我们也在这两个参照量和实际测得的机器人状态的偏差之间做加权求出总误差，减少参数个数，以方便参数的调试。其中，机器人当前的转弯半径由路径规划给出的当前机器人所要跟踪圆弧路径得到，机器人相对圆弧圆心的距离和参考半径的差即距离误差。而由全场定位算法得到的机器人实时位置对应所跟踪的圆弧计算出的应转的角度和由陀螺仪反馈回来的自身角度之间的差即角度偏差。

$$e = \Delta R + k * \Delta\theta \tag{5.7}$$

上述圆弧跟踪算法在程序里由以下函数实现:

int32 GoArc（ struct CPoint startPoint,struct CPoint endPoint,struct CPoint arcCentre,int32 deltaDegAngle,int32 arcRadius）

5.2.4 终点准确定位算法

之前的直线和圆弧跟踪算法只是确保了机器人在行走过程中的精确轨迹跟踪，但机器人的准确停靠则是又一个棘手的问题。由于惯性，机器人往往要超过设定的终点才停下来，即使我们在终点附近减速，由于场地的原因，我们很难保证真正比赛时的场地摩擦系数和自己实验场地恰好一致，所以，必须开发更可靠的终点准确定位控制算法。

正如文献所述，点镇定的问题是很难控制的，我们在机器人离终点还有 500mm 左右距离时开始调用精确停车函数，此时也采用闭环 PID 调节，但与之前行走轨迹跟踪控制算法不同，在前面的轨迹跟踪控制中，我们仅仅将左右轮差速作为被控量，车左右轮的平均速度是我们根据实际情况设定的，在终点控制中，车左右轮差速和平均速度都成为了被控量，进行 PID 闭环控制。

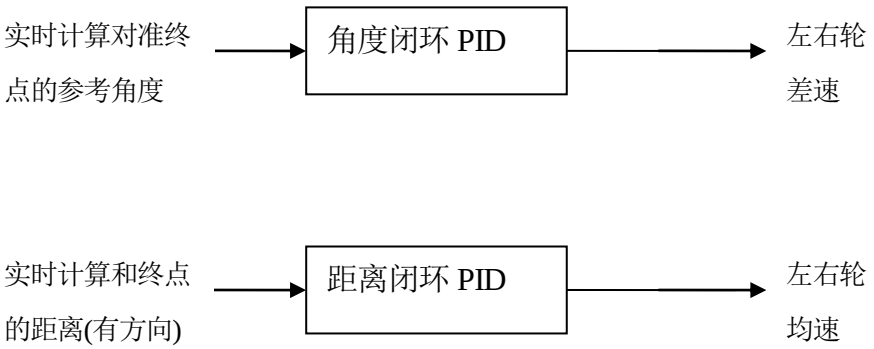


图 5.7 机器人闭环终点控制框架

5.3 机器人轨迹跟踪实际问题及解决办法

实际调试的过程中，一开始受限于行走模块的处理器对两个左右轮电机控制盒串口发送速度命令的时间限制，由于对一个电机控制盒发送速度命令需要 8ms 的时间花费，导致我们在程序中将控制周期设定为 20ms，而实际上真正用于算法数据处理的时间不到 4ms,一方面造成了处理器性能的浪费，更重要的是，使得算法的控制效果的不到充分实现。另外一个问题是左右两个电机控制盒不是被同时设定速度，之间有 8ms 的时间差，如果每次都先给左轮电机控制盒发速度，实际直观的控制效果总是给人有左轮先启

动的感觉。我们用各个控制周期内轮流改变对两个控制盒发送顺序的简单方法改善了控制效果。后期，我们终于找到了比较稳定的对电机控制盒采用 PWM 波控制方式，将控制周期缩短到了 5ms，改善了控制效果。

另外轨迹跟踪算法实际中最大的问题是各个直线和圆弧段之间的衔接，我们为了保证机器人在各段间衔接的流畅性，我们对各衔接点设置了死区，在机器人到达前段终点的附近马上转入对下一段路径的跟踪。

5.4 机器人轨迹跟踪控制的实验效果

平时调试机器人时积累的行走闭环控制数据结果如下表所示。

表 5.1 机器人行走轨迹坐标误差比较（20 次测试的实验结果）

机器人速度 2m/s	走圆弧	走直线
最大误差（单位:mm）	352	245
最小误差（单位:mm）	18	0
平均误差（单位:mm）	55	39

从表 5.1 可以看出，轨迹 控制误差一般都在 100mm 以内，误差基本可以接受。

表 5.2 机器人行走取块得分情况统计

得分点	测试次数	成功次数	成功率
黄块（2.5m/s）	200	185	92.5%
己方白块 （3m/s）	200	188	94.0%
对方白块（2m/s）	200	179	89.5%

从表 5.2 可以看出，应用以上论述的机器人轨迹跟踪控制算法，取块得分成功率在 90%左右，达到了上场车的要求。

5.5 本章小结

本章论述了自动机器人轨迹跟踪和终点停靠闭环控制的方法与实现，论述了机器人直线行走和圆弧行走以及终点准确定位的方法。实现了对机器人行走的精确控制、结合上一章论述的全场定位，使我们的机器人能够出现在场上任意位置，扩大了机器人的活动范围，为机器人做出整套得分、干扰和保护动作奠定了基础。也为下章所述的路径规划提供了物理实现的保证。

6 自主移动机器人主脑层路径规划及避障算法设计与实现

6.1 自主移动机器人路径规划概述

6.1.1 自动机器人路径自主规划的意义

在往几届的比赛中，我们学校机器人行走的路线都是队员们在赛前一步一步写好调好的，这样调试下来，将会有非常大的工作量，上届就有队员调试一辆车写了近万条路线。除了工作量大，还会耗费很多宝贵的调试时间，而且，单片机的内存 RAM 有限，事先不可能无止境地存储任意多的路径。更重要的是，不能实时自主路径，机器人就无法实现真正的机动性和智能性，给程序的架构也会带来问题。鉴于此，我们的讨论结果是，今年的自动机器人必须能够根据起始点和终止点信息自主规划路径并寻线到达，这样不但能极大的减少工作量，而且可以让我们快速地编写出新的策略。而上一章节所介绍的全场定位算法和路径控制算法的实现，则为我们的路径规划奠定了非常坚实的基础。

6.1.2 常用的路径规划方法

由于路径规划一般是基于电子地图的，所以在介绍常用的路径规划方法前， 先介绍一下常用的地图表示方法。

1） 常用的地图表示方法。

（1） 连续地图的表示方法

连续值地图是环境精确分解的一种方法，环境特征的位置可以在连续空间中精确地予以标记。连续地图表示方法的主要优点是：对于环境的配置以及该环境内的机器人的位置，有较准确和高表达性的潜能。其危险是地图计算上的耗费，但这个危险可以用抽象化和至获取最相关的环境特征来缓解。从根本上来说，这个实现现实世界到地图表示的变换是一个滤波器，滤去了所有非直线数据。

（2） 地图的栅格表示方法

在移动机器人中，固定分解的概念很普遍。固定占有栅格法是一种流行的方法。当然也有环境自适应（单元可变）的动态栅格占有法。但栅格占有法也有其缺陷：机器人

储存器中的地图尺寸会随着环境规模的增大而增大。而且，任何固定分解，不管环境细节如何，均需预先加上一个几何栅格[19] [20]。

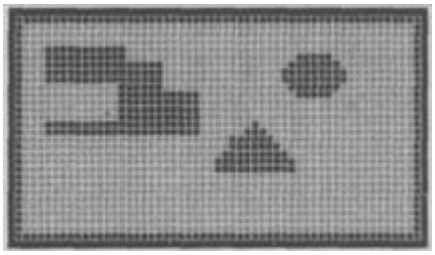


图 6.1 栅格电子地图

（3） 地图的拓补表示方法[21]

拓补分解方法的主要动机是环境可能包括重要的几何特征（与测距无关），避免了对几何环境品质的直接测量，而注重于机器人定位最相关的环境特征。实际上由于机器人定位和导航往往是紧密结合的，常常用拓补图来进一步实现机器人导航。此时机器人必须满足两个约束：它必须有根据拓补点来检测它当前位置的方法，另一点是它必须有在结点间行走的方法[21[22]]。下面介绍常用的路径规划方法。

2） 常用的路径规划方法。

（1） 拓扑法

这种方法的优点在于利用拓扑特征来缩小搜索空间,拓扑法的复杂性仅仅依赖于障碍物的数目,在理论上是完备的。缺点在于表示的复杂性、特殊性。建立拓扑网的过程相当复杂,较难实现。

（2） 人工势场法[23]

该方法的优点是可以使机器人迅速躲开突发障碍物,实时性好。但是经常会因为局部极点而使机器人运动到一个死区,发生震荡现象而导致规划失败。由于其计算简单,因此常被用于局部路径规划。

（3） 遗传算法

遗传算法具有优良的全局寻优能力和隐含的并行计算特性,提高了路径规划问题的求解质量和求解效率。

（4） 神经网络法

神经网络是一个高度并行的分布式系统,处理视觉系统探测到的图像速度高,可以充分利用其非线性处理能力达到环境及坐标辨识的目的,还可以完成机器人内部坐标和全局坐标的快速转换。在有监督的情况下,学习网络的最大缺点在于环境改变后必须重新学习,这在环境信息不完整或环境经常改变的情况下难以应用。

（5） 栅格法

该方法的特点是简单和易于实现,易于扩展到三维环境。它的缺点是对工作区域的大小有一定的要求,如果区域太大,将使栅格的数量急剧增加,使搜索存在组合爆炸的问题。

6.2 自动机器人避障处理

6.2.1 自动机器人避障的意义

在过去的几届比赛中，经常会出现机器人发生相撞从而撞出直线无法完成既定任务的情况。在多数情况下，我们学校都是采用从机械上增加机器人撞击的稳定性出发来解决的。尽量使撞击后我方的机器人不出现出线的情况。但是这并不是解决问题的根本办法，不论机械上做得再好还是避免猛烈碰撞导致的致命后果——机器人迷失自己。这个问题在本届比赛中更是占显得尤为突出，因为今年的比赛有着与以往任何一届都不同的特点：得分点特别多，导向白线路径复杂，自动区较小，自动机器人可能需要在场地的任意位置自主取块。在狭小的自动区内将由六辆自动机器人运动，这样的环境下出现机器人碰撞的可能性必将大大增加，可是既然我们做不到在碰撞丢失之后再重新找回来，那么我们就必须要避免我们的自动机器人与其他机器人的碰撞，保护自己。

6.2.2 障碍点的检测

要避免与其他机器人相撞首先必须能敏锐的检测前方的机器人，也就是必须考虑障碍点的检测。那采取什么方法检测障碍呢？我们依据现有条件采用了以下两种障碍检测方式：

1） 从动轮码盘反馈：

也就是“撞”的方法，这可能是最容易想到的，也是最普遍的办法。机器人在行进的过程中受到障碍物的阻碍，无法顺畅前进，即认为遇障。可通过从动码盘信号检测到。优点是身体力行后“知难而退”；缺点是在高速行进的情况下，可能会因冲撞导致车体出现滑行、旋转等复杂的情况，从而失去对自身状态的定位。

2） 视觉图像处理：

这是比较高层次的障碍检测，也就是智能型比较高的方式。优点是能够直接从图像信息判断；但是缺点是难度较大，并且最大的问题还是可靠性和稳定性。

考虑到我们的实际情况，既要简单可行又要准确率比较高。经过讨论，我们决定采用摄像头视觉和从动轮码盘综合使用检测障碍。利用视觉去主动躲避障碍物，而为了能够准确的判断障碍并且保护底盘驱动电机，我们为机器人加入了防堵转控制程序，这样通过从动轮感知电机堵转的方式既能保护我们贵重且稀缺的驱动电机，而且能比较准确的判断出前方障碍，进行避障。

6.2.3 障碍点的描述

有了障碍点的检测手段，接下来的任务就是如何描述探测到的障碍点。在 6.3.2 中将会讲到我们把场地抽象为一个二维数 能通过者，于是只要检测到障碍点后把相应的场地交叉点的值赋为 1 就可以了。但在实际的应用中，因为障碍点往往是对方或己方的机器人，是可移动的，所以我们还要在一段时间后清除障碍点。恢复电子地图的初始状态。

6.3 本届大赛自动机器人路径规划及避障的设计

如 6.1.1 自动机器人路径自主规划的意义和 6.2.1 自动机器人避障的意义所述，面对本届比赛复杂的场地和繁琐的规则，我们必须提高自动机器人的智能性，首先应该做的就是自动机器人的路径规划及避障。

对于本届比赛来说，我们的自动机器人没有采取寻白线前进为主的技术，故自动机器人的路径规划及避障不是像往届那样基于寻线的方式，而是在全场自由行动，这个特点，增加了路径规划和避障的难度。

第一步要做的是设计自动机器人的路径规划。自动机器人如果要能做到自主的路径规划，他就必须知道整个场地的情况如何以及自己现在什么位置。这两点都依赖于对场地的描述。虽然我们的路径规划不是基于寻线的，但是我们对全局地图的描述是离散的，分辨率有限。后面会提到本文作者首先是用二维数组来描述全场的地形，在数组空间搜索出路径后再通过直线圆弧拟合的方式计算出最后实际的路径。

第二步要做的就是基于自动机器人的路径规划的避障设计。我们已经想到了探测障碍的办法，剩下来的就是对障碍的处理。我们的思路是：判断前方节点为障碍点则退回上一个经过的节点，然后以当前所在节点为起始点，以目标节点为终止点重新进行一次路径规划绕开前方障碍点到达目的地。这个方案最大的困难来自对障碍的定位，具体说就是程序中应该将电子地图中哪几个位置置 1（1 代表障碍）。

在这两个工作里面最重要最基础的是自动机器人的路径规划，下面各节将详细介绍本届大赛中使用的自动机器人路径规划算法。

6.3.1 本届大赛场地具体分析和描述

本届大赛场地图如图 6.1 所示：

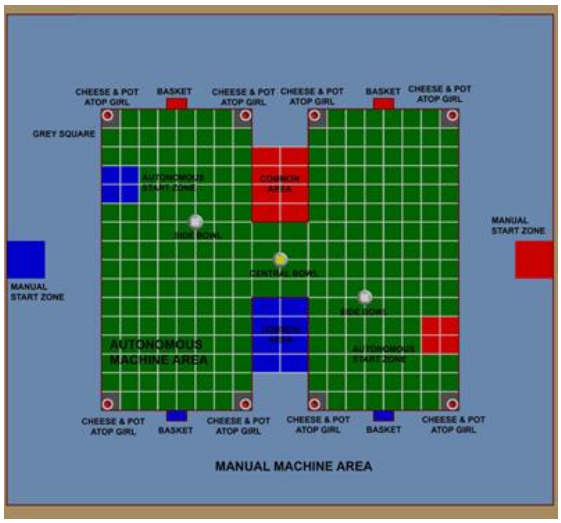
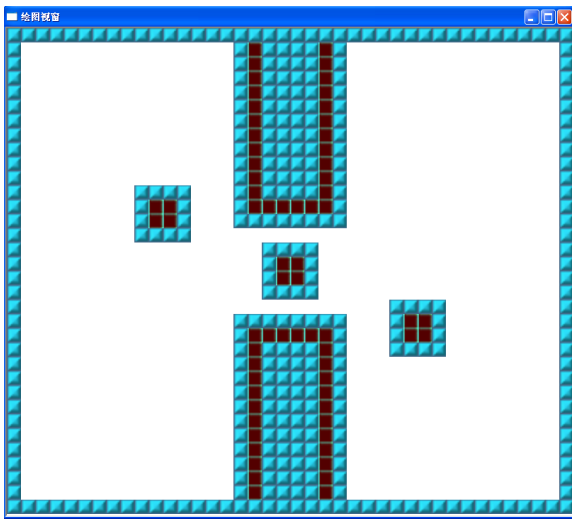


图 6.2 第七届机器人大赛比赛场地图图



6.3 第七届机器人大赛比赛场地栅格地图

如前面所述，我们的自动机器人虽然不是基于寻线的，但是用电子地图来描述场地的分辨率毕竟是有限的，本文作者采取二维数组来直观地表示全场地图，每个点表示的范围是 25cm*25cm 的正方形区域。

我们将场地上的每个交叉点位置通过两个坐标来表达出来：纵坐标和横坐标。考虑场地的四周边界，整个场地可以分为（38+2）*（32+2）的二维区域，于是每个交叉点的纵坐标范围是：1~38，横坐标范围是：1~32。如果将白线的交叉点作为一个二维数组的一个元素，将可以到达的点置位 0，不可到达的点置为 1，则可以得到比赛场地的数组描述如下：

```
signed char map[MAPHEIGHT][MAPWIDTH]=
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,1,5,5,1,0,0,0,1,5,5,5,5,5,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,1,5,5,1,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,1,5,5,1,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,5,5,5,5,5,1,0,0,0,1,5,5,1,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
```

1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,

1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,

1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,

1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,5,1,1,1,1,5,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};

需要注意的是，由于最后的路径规划是在连续二维空间中进行的，为了防止机器人与障碍向切，或者由于拟合的原因导致路经与障碍相交，作者人为地放大了障碍的边缘。如图 6.5 所示，棕色区域是真正的障碍边缘，而地图中将其认为放大到了蓝色边缘，为后面的连续空间路径规划留有了一定裕量。

6.3.2 算法的思路与设计

在已知机器人当前位置和目标位置的情况下，基于已知的全场地图，算法从选定的初始节点出发一步一步向前考察，将当前节点压入栈中，从当前节点位置按一定优先顺序（考察的顺序是算法的关键，由启发函数来确定）考察临近的上下左右四个节点，如果走入死胡同（考察的节点四周都是障碍），或者前面能够到达的节点都已经去过（需要另一张表来标志每个节点是否去过），那么就继续考察下一个临近点，一旦找到临近的节点可以继续前进，则将此节点压入栈中。按照上面的模式进行的搜索是一种“深度优先搜索”，因为在这种搜索中，总是设法向前再，只有到无法继续前进的时候，才回头寻找其他路径，是一种尽量向深处搜索的方式，显然，堆栈适合用来存储探索过程中所遇点的结构，因为点的存储与取出正是先进后出的顺序。对于栈的具体实现方式，本文作者采用了链表的数据结构来实现，在程序里用结构体 list：

```
struct list{
    char m;
    char n;
    struct list *next;
    struct list *back;
};
```

来表示链表的每个单位。另外，正如前段所提到的那样，我们需要另外一张表来标志每个节点是否去过。在程序里用二维数组 pass[HEIGHT][WIDTH]来表示。

在作者的深度搜索算法中，理论上并不能保证能生成两点间的最短路径，事实上算法中的启发函数起了非常重要的作用。如图 6.4 所示，在没有用启发函数的情况下，深度优先算法生成的路线非常不合理，如图 6.5 所示，机器人将要绕个大弯才到达目标点。如图 6.6、图 6.7 所示，在使用启发式搜索的情况下，算法生成的路径有了显著的改进，可以说是最优的了。下面论述作者采用的启发式搜索策略：

1） 半场的启发式搜索策略

在立足当前节点搜索下一个节点时，算法应当设定搜索的优先级，程序中的四个方向搜索的优先级由目标点和起始点的横向纵向距离决定。如图 6.4：当目标点在当前点的右下方，并且纵向距离大于横向距离时，算法中的优先级顺序变为下右左上（注意与最高优先级相对的搜索方向要设为最低优先级，否则在终点和起点在横坐标或纵坐标相同的情况下会产生问题），从图中可以看到，由于一开始下优先级最高，绿色搜索路径先向下延伸，在达到纵横距离相等的时候，出现了向右向下搜索方向交替优先的情况，搜索路径表现为一横一纵沿 45 度交替延伸，以几乎最优的路径到达了目标点。

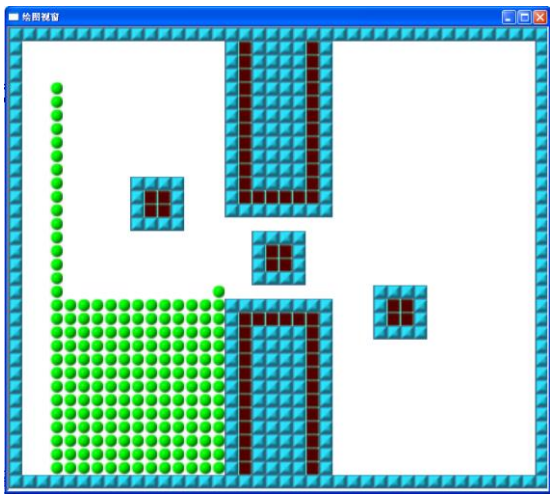


图 6.4 非启发式深度搜索的基本路径（绿色单元）

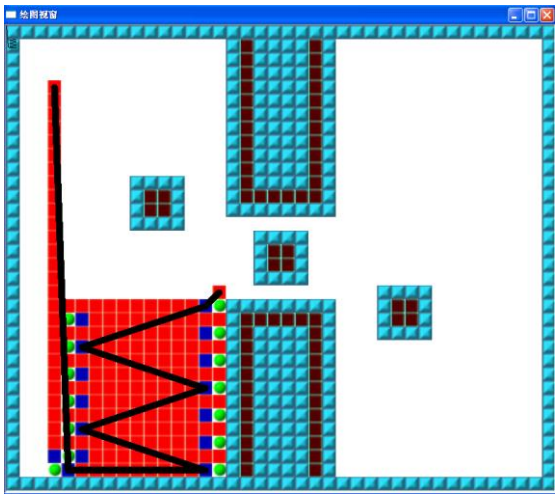


图 6.5 非启发式深度搜索的折线路径（黑线）

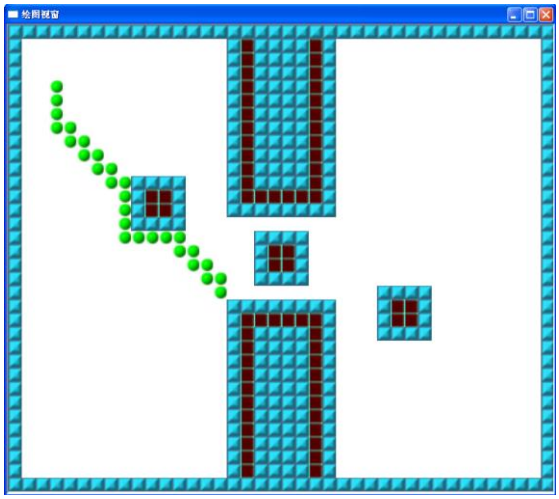


图 6.6 启发式深度搜索的基本路径（绿色单元）

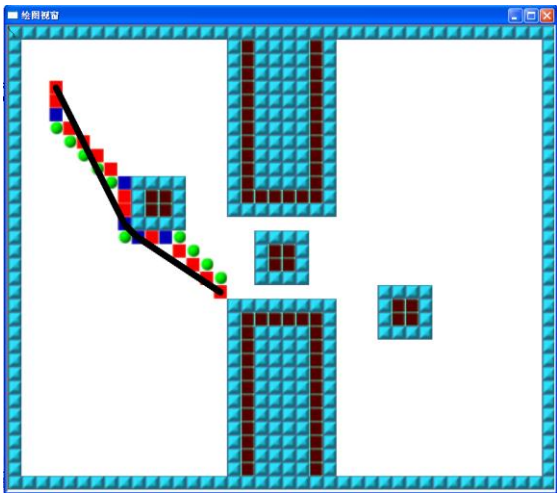


图 6.7 启发式深度搜索的折线路径（黑线）

2） 跨半场的启发式搜索策略

上面的启发式搜索策略在半场取得了很好的效果，但是在跨半场时，由于起点和终点之间往往有公共区的很长的障碍地带，如图 6.8，终点在起点的右上方，如果采取原来的启发式搜索策略，搜索路径一定会向右，结果是撞到中间的障碍地带。所以，必须对跨半场的情况该进启发函数。本文作者的策略是在半场内先暂时设定一个临时的目标点（选择地图中间狭长通道区域），待当前点到达该临时目标点或者附近时，修改回为原来的目标点。如图 6.10，这样的搜索策略，收到了很好的路径生成效果。

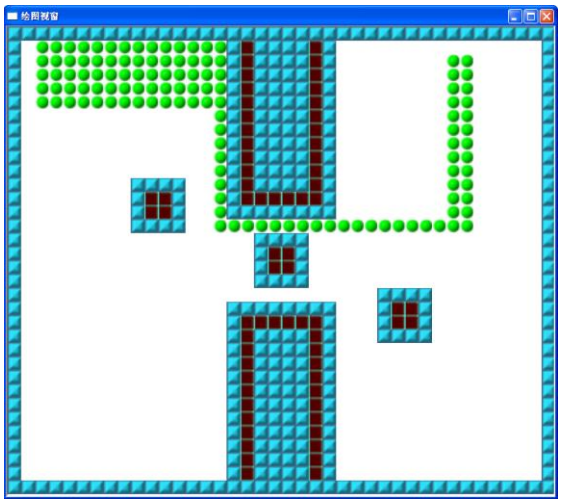


图 6.8 启发式深度搜索的基本路径（绿色单元）

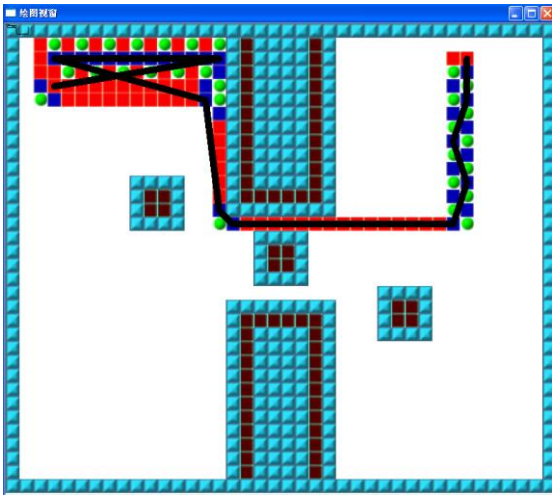


图 6.9 启发式深度搜索的折线路径（黑线）

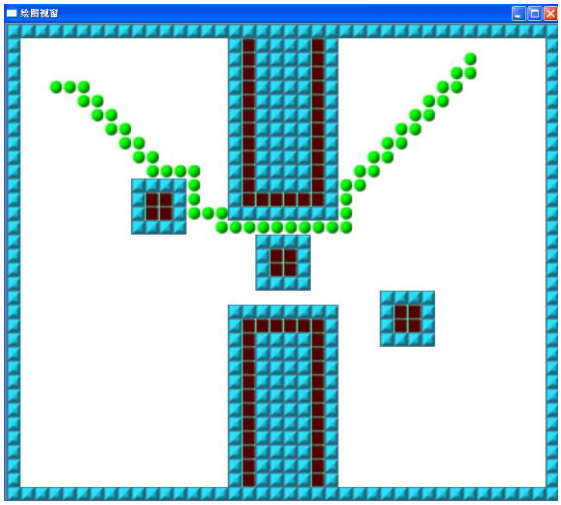


图 6.10 改进后的启发式深度搜索的基本路径

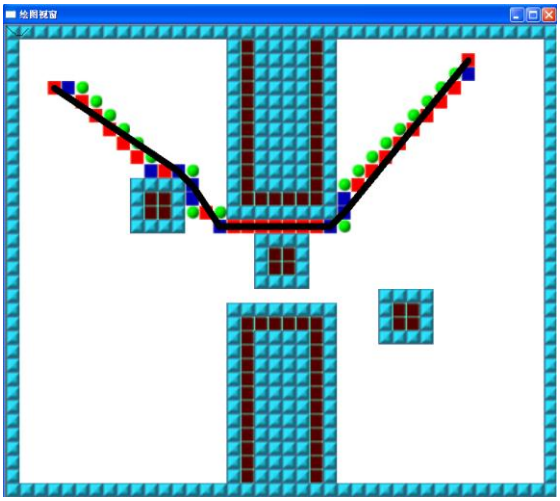


图 6.11 改进后的启发式深度搜索的折线路径

3）离散路径到连续路径的转换方法

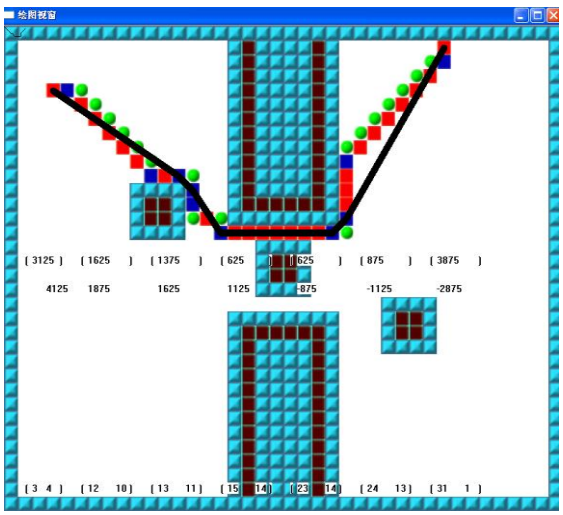
在得到离散空间的路径后，理论上机器人可以按这种折线路径行走，但实际由于底层控制方法的原因，机器人在走这种分段折线时有以下的缺陷：

（1）分段折线加减速控制较难，要求机器人在高速情况下准确定位到线段的终点非常困难；

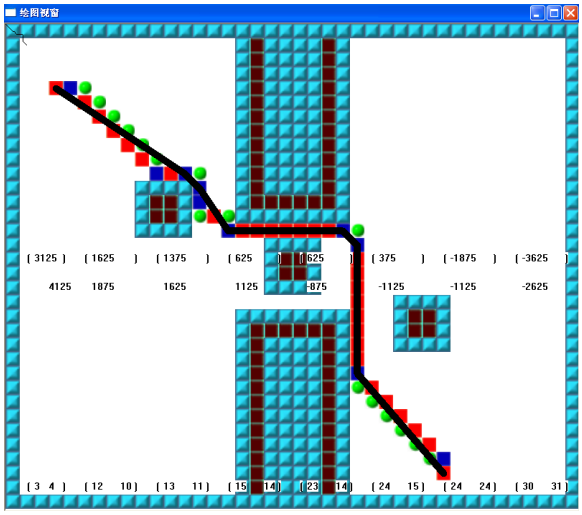
（2）在折线交替阶段，机器人不得不停下来原地转向，这实际上会浪费大量的时间；

所以，我们需要进一步开发连续路径的规划算法，使得机器人能够顺畅行进，衡量路径规划最优的标准不一定是路径最短，而是要考虑机器人行进时的机械运动规律，比如转角。下面介绍把离散空间生成的折线转换到连续路径的方法：

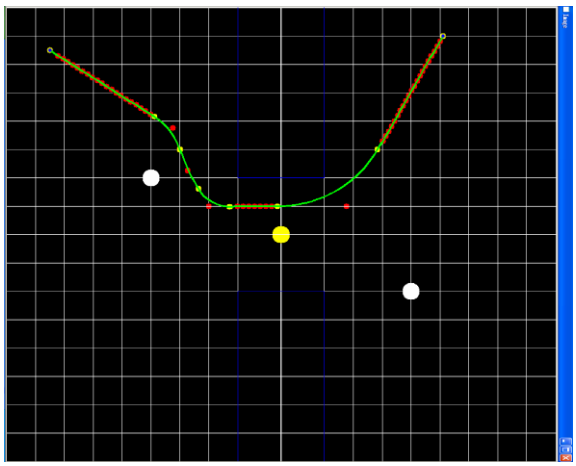
在得到折线路径后，作者使用了各段折线的中点作为拟合的基础点系。相邻两个点之间则用圆弧加直线的方式连接起来，本质上是一种二次曲线路径。以上就是最基本的想法。我们设计了两点之间已知出射角和入射角的连接算法，而出射角和入射角则是两点所在的折线角度，这样的设计，保证了连续路径能够较平滑地逼近折线。作者认为这也是一种对基础点系拟合的方法。如图 6.12 所示：



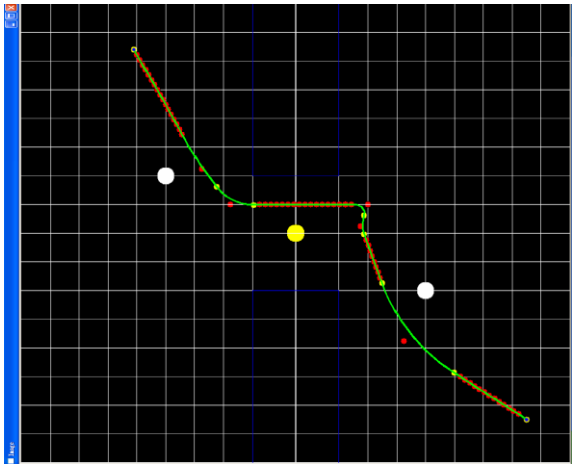
（a）全场路径规划示例 1—栅格路径



（b）全场路径规划示例 2—栅格路径

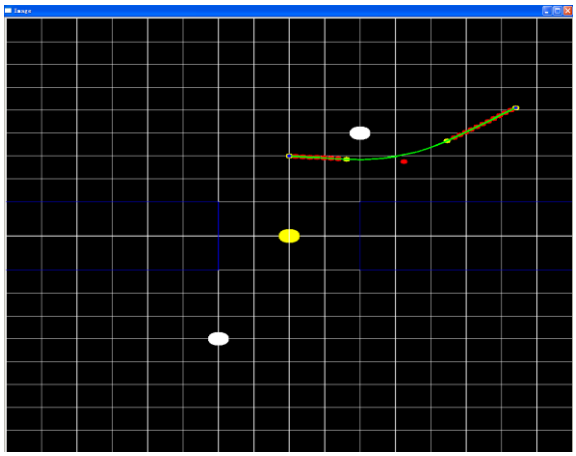


（c）全场路径规划示例 1—连续路径

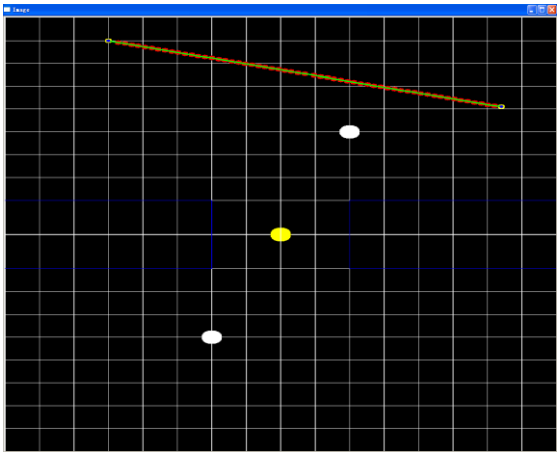


（d）全场路径规划示例 2—连续路径

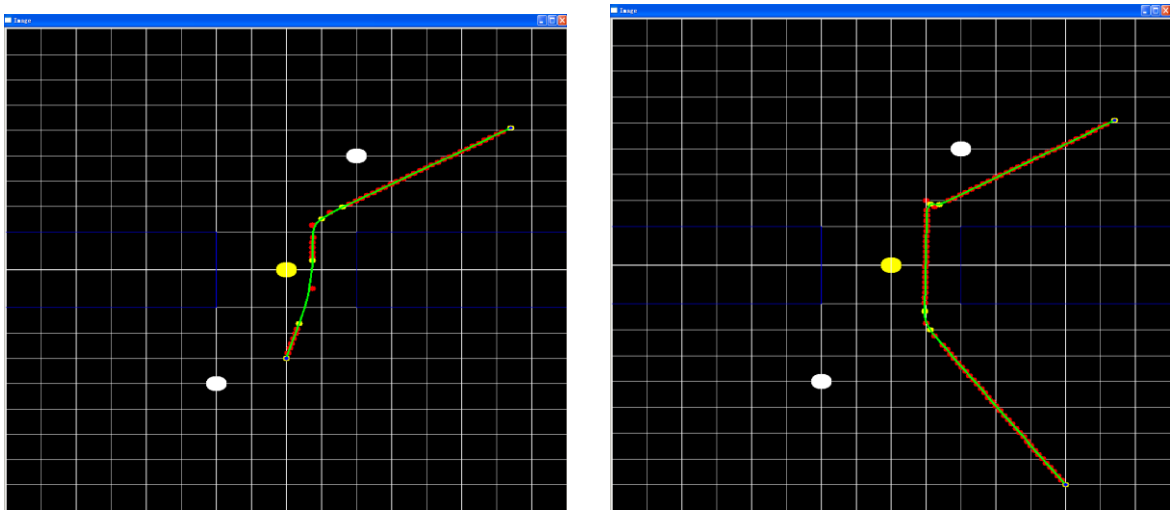
图 6.12　路径规划栅格路径到连续路径仿真示例



（a）



（b）



（c）　　（d）

图 6.13 更多的路径规划仿真示例

由于章节安排的原因，本文将基于视觉的障碍探测定位放到了视觉一章详细论述。

6.3.3 算法关键函数与流程图

从前面的论述分析可以得知，实际上自动车的路径规划系统可以分为两大块：一是知道起始点和终止点后通过理论算法计算出中间的通路；二是知道从起始点到终止点的通路后实际控制底层驱动电机按照通路顺序真正“走”到终止点。于是，与路径规划算法相关的关键函数也分为这两大块。下面分别介绍它们的功能、用法以及函数流程。

1） 和算法相关的函数

（1） void changeBeginEnd （void）

函数功能是必要时交换起点和终点，这样做的原因是两点间路径规划并不是对称的，同样的两个点起点的选择不同会导致路径优化有差别。

（2） void initSmallMap（void）

初始化栅格地图，确认起点和终点，主要对起点和终点的输入坐标作错误处理。

（3） void Search（void）

启发式搜索函数，进行单步搜索，包含在一个 while 循环中。

（4） int32 bestPolicy（void）

进行栅格地图第一次路径优化，得到基本路径，去掉频繁的 45 度折角。上面示意图中绿色单元是启发式搜索后的初始路径，包含许多之字形单元，经过此函数优化后仅保留了红色单元。输出为得到的初步路径的点的个数

（5） int32 optimize（int k）

进行栅格地图第二次路径优化，找到拐点，如图中的蓝色单元就是优化后的结果，是路径中的各个拐点。输入为得到的初步路径的点的个数，输出为得到的一次优化路径的点的个数。

（6） void opagain（int s,int k）

进行栅格地图第二次路径优化,其实可以生成上图中的黑线折线路径。输入为得到的初步路径的点的个数,得到的一次优化路径的点的个数。

（7） void RecordPath（struct Path *path）

记录整个路径。输入为整个路径的结构体

（8） int8 autoRoute（struct Point p1,struct Point p2）

路径规划算法对外提供的接口函数，输入为起点和终点坐标，输出为函数执行标志，成功返回 1，失败为 0。

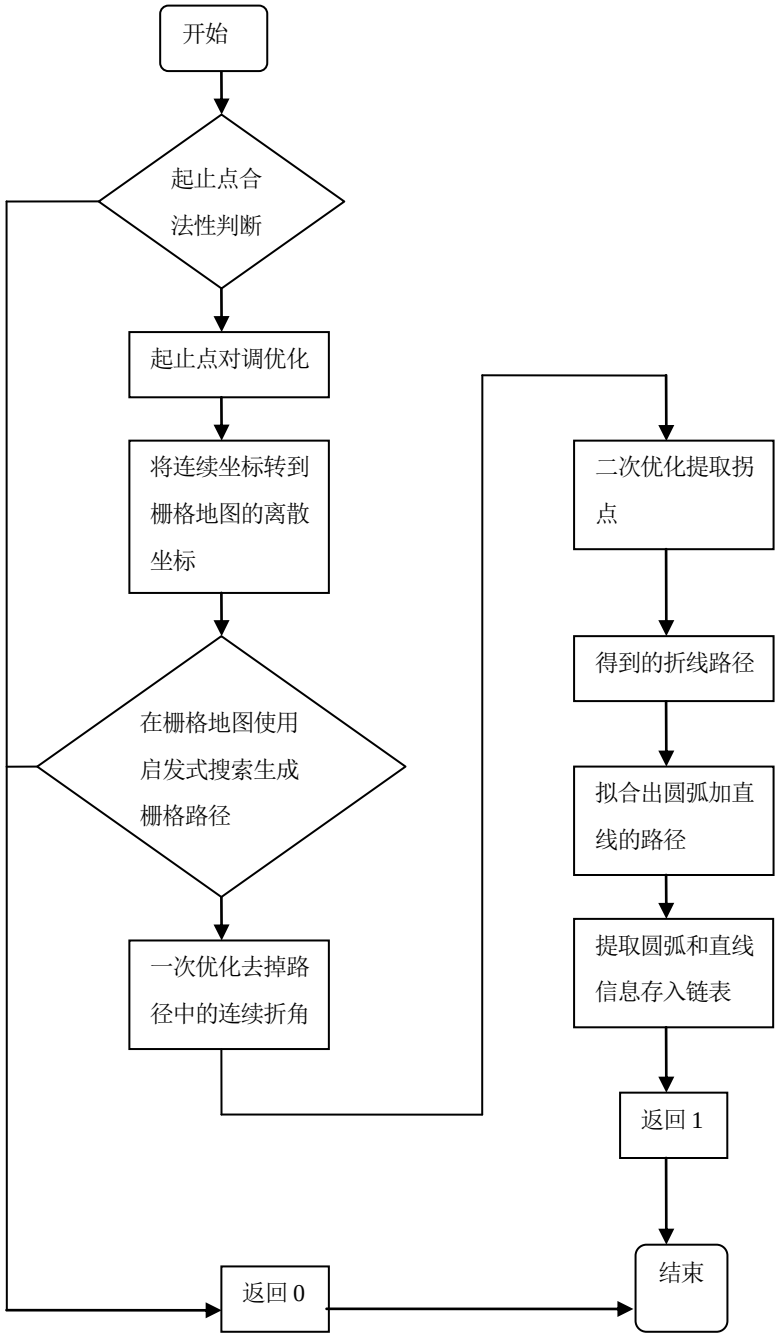


图 6.14　启发式深度搜索算法流程图

2） 和行走有关的函数

（1） int32 32GoStraightLine（struct CPoint startPoint,struct CPoint endPoint）

走直线函数,输入为起始点，终止点输出电机差速。

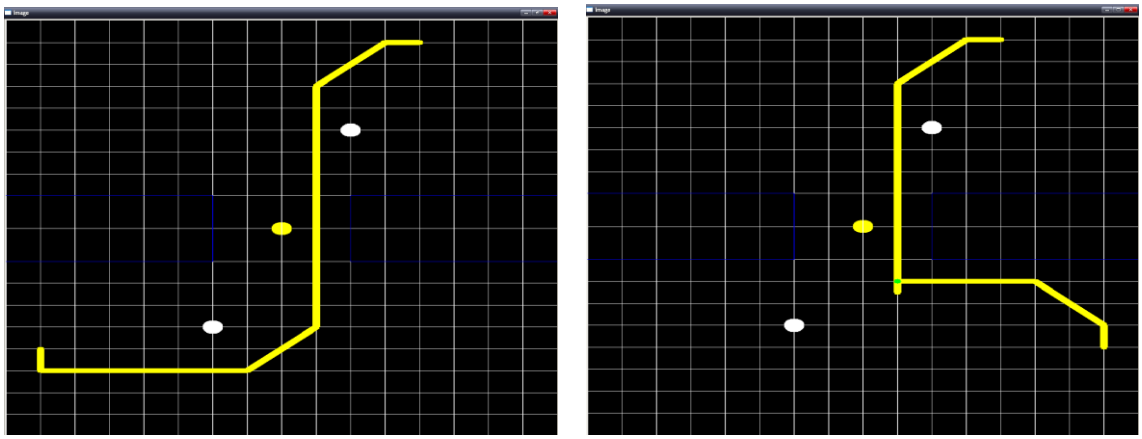
（2） int32 GoArc（struct CPoint startPoint,struct CPoint endPoint,struct CPoint arcCentre,int32 deltaDegAngle,int32 arcRadius）

走圆弧函数,输入为起始点，终止点，圆心，增加角度，半径，输出电机差速。

6.3.4 算法在单片机的移植和修改

自动车的路径规划算法在 PC 机上用 MFC 结合 OPENCV 仿真证明了它的有效性，作者下一步工作就是把算法从 PC 机上移植到本届大赛的主控 ARM 芯片 LPC2138 里并加入对自动车底盘驱动电机的控制已完成完整的自动车路径规划系统程序。

在移植过程中，作者遇到了很多的困难和问题，理论和实际之间存在很大的差距。其中最大的问题是原先搜索出来的路径虽然在长度上达到了几乎最短，所以在自主移动机器人上，作者使用了上述的路径规划方法，并取得了成功。但是，当作者想将此套路径规划方法移植到冲中干扰机器人上时，发现因为干扰机器人是同侧轮驱动，车身狭窄，不容易转向，这样规划出来的路径并不适合干扰车的路径行走控制，为了追求冲中干扰机器人实际控制中的稳定性，我们最后决定放弃路径长度上的最优化，而去追求轨迹的简单性和可控性[24]。所以，作者把电子地图的分辨率降低了一半，并略去了其中一些对路径长度上的优化，以此来换取路径的简单性。在单片机上移植修改的针对干扰机器人路径规划结果如下图 6.15 所示。



（a） 全场路径规划示例 1-简化后路径　　（b）全场路径规划示例 2—简化后路径

图 6.15　使用于干扰机器人的启发式深度搜索算法生成的路径

经过反复的调试和修改，终于完成了自动车完整的路径规划系统，并在全队自动车中普及了这一技术，实现了本届比赛中自动车自主路径规划的目标。在 ARM 上的移植

最重要的是加更多的错误处理，如果目标点是障碍点或者当前点合障碍点间已经无法连通，则路径规划会失败，程序无法正确执行下去，所以，在程序中要对搜索次数加以限制，当搜索次数大于 10000 次时，路径规划函数将返回 0，表示执行失败，避免了无法程序返回的情况发生。

表 6.1 机器人行走遇障之后成功到达目标点的情况统计

速度 1.5m/s	测试次数	成功次数	成功率
对方小白	20	16	80%
对方半场己方篮区	20	17	85%
对方半场对方篮区	20	19	95%

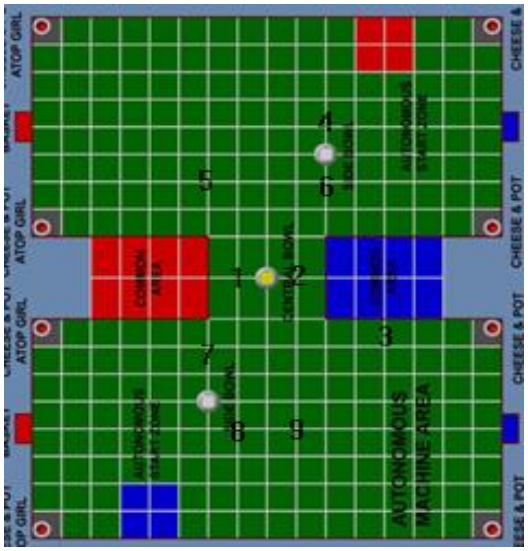
从表 6.1 以及我们的机器人在平时调试和内部比赛的发挥来看，算法的移植和运用是完全成功和可靠的。

6.4 自主移动机器人路径规划的实际应用

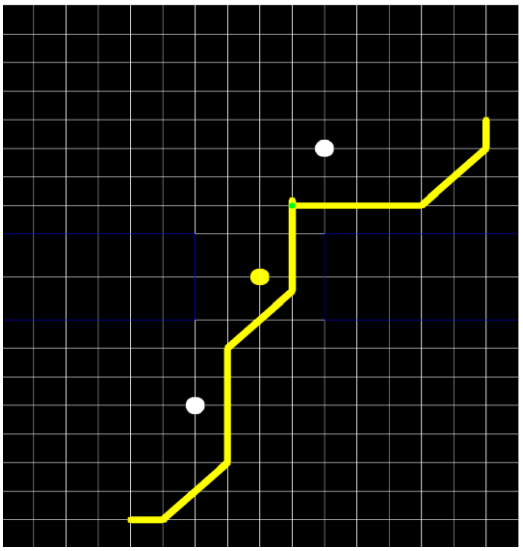
机器人的路径规划能力是实现机器人自主移动能力非常重要的一个方面，作者在利用这一技术时，主要用在了两个方面。一个是机器人在比赛中重试时的点到点自主到位，这时是操作手通过观察场上局势手输障碍点的位置。另一个作用是机器人在通过堵转检测或者视觉获得障碍的位置信息后，更新电子地图，动态重新规划出一条路径来实现机器人的自主蔽障，对在移动过程中出现的问题进行错误处理。

6.4.1 机器人重试时的路径规划

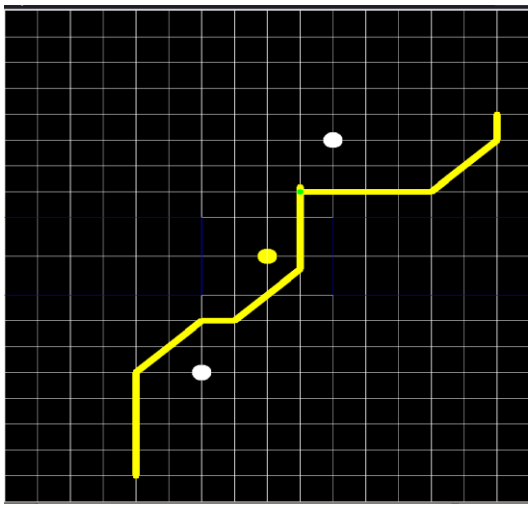
由于路径在内存中是以数组表示的，每段路径是一段直线或者圆弧，包括起点、终点、圆心以及半径和方向等多种信息，所以十分消耗内存。作者曾经尝试过手写 50 条点到点的单位路径存在内存中，结果程序占据了单片机 32M 内存中的 29M,十分惊人。尽管我们可以选择存在 flash 中，节省宝贵的内存资源，但是让操作手记忆如此多的各关键点间不同单元路径不够人性化，更重要的是比赛中操作手由于各种原因可能输错路径编号，导致比赛前功尽弃。



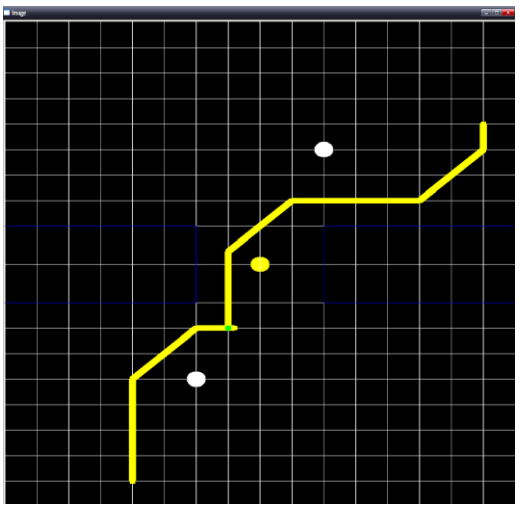
（a）障碍点设定区域对应的标号图



（b） 1379 区域设为障碍点后规划出的路径



（c）1389 区域设为障碍点后规划出的路径



（d）2389 区域设为障碍点后规划出的路径

6.16　重试时干扰机器人点对点干扰路径规划举例

所以，作者提出了一种新的操作方法。操作手在重试时需要输入目标点坐标和当前场上障碍点的坐标，即可利用路径规划使机器人自主路径达到目标点并执行相应动作。考虑到比赛时出现的障碍不外乎对方和己方机器人，所以作者对九个可能出现障碍的地点进行编号，免去了输入障碍点坐标的麻烦，而目标点也仅需要输入两个两位数，加上动作编号，一共需要操作手输入两次四位数和一次个位数，不到三秒钟的输入时间。提高了效率，极大地提高了针对场上混乱情况的灵活性，并且不容易出错。最重要的是，这种根据场上形势实现点到点精确到位的功能为我们在 12 分的黄块已经被对方得到的

不利局面下反败为胜提供了可能。我们可以在重试时启动干扰机器人到达指定位置干扰对方得分车，将其撞倒，使其得分无效，实现点对点的精确打击，如此出奇制胜，是我们反败为胜的关键！

6.4.2 机器人遇障时的路径规划

当机器人遇到障碍并检测到时，机器人将先后退一定距离，使用路径规划动态规划出一条到目标点的路径，并适当放慢速度继续沿新的路径前进，这时，我们可以使用原来规划出来的比较优化的连续的路径，而不是干扰机器人那种折线构成的路径。

6.5 本章小结

本章论述了自动机器人主脑层路径规划及和避障的方法与实现，比较了各种探测障碍的方法，并提出了一种基于启发式深度优先的搜索发法进行路径规划的方法和其具体的应用。在机器人全场定位和行走控制得到保证的基础上，路径规划在我们队内联调和模拟赛中已经发挥出了巨大的作用，他使得机器人的智能性得到了本质上的提高，更极大地丰富了我们在比赛中的重试策略。

7 基于计算机视觉的自主移动机器人目标跟踪和二次定位

7.1 计算机视觉简介

7.1.1 Marr 理论简介

根据 Marr 教授提出的理论，机器视觉有三大研究方向，即第一，低层视觉（low level），它主要是对输入的原始图像进行处理。这一过程借用了大量的图像处理技术和算法，如图像滤波、图像增强、边缘检测等，以便从图像中抽取诸如角点、边缘、线条、边界以及色彩等关于场景的基本特征；这一过程还包含了各种图像变换（如校正）、图像纹理检测、图像运动检测等。第二，中层视觉（middle level），它的主要任务是恢复场景的深度、表面法线方向、轮廓等有关场景的 2．5 维信息，实现的途径有立体视觉（stereo vision）、测距成像（rangefinder）运动估计（motion estimation）、明暗特征、纹理特征等所谓的从 X 恢复形状的估计方法。系统标定、系统成像模型等研究内容一般也是在这个层次上进行的．第三，高层视觉（high level），它的主要任务是在以物体为中心的坐标系中，在原始输入图像、图像基本特征、2.5 维图的基础上，恢复物体的完整三维图，建立物体三维描述，识别三维物体并确定物体的位置和方向。

Marr 理论是计算机视觉研究领域的划时代成就，但该理论不是十分完善的，许多方面还有争议。比如：视觉处理框架基本上是自下而上的，没有反馈，假定了视觉计算是由局部信息到整体信息的单向过程；没有足够地重视知识的应用，假定了视觉系统是对视觉环境的被动响应。虽然 Marr 理论存在一些缺陷，但在过去的 20 多年中该理论一直处于机器视觉的主导地位，所以本文就该理论对计算机视觉技术作了简单的论述。

7.1.2 计算机视觉应用系统的组成

机器视觉的应用系统一般可以分为感觉、预处理、分割、描述、识别和解释六个主要部分。感觉是获取视觉图像的过程，利用电视摄像机、CCD 或其他传感器，将光信号转换为电信号，数字化处理后供分析使用。预处理则是对获取的二维图像进行降噪、增强、二值化等技术处理，便于下一步图像分割。分割是将图像划分成若干有确定含义的物体的过程。描述则是讨论如何进行便于区分不同类型物体的特征（尺寸、形状等）

的计算。识别则是通过模板匹配理解物体的过程。解释是将某种含义赋给由已识别出的物体组成的组合体的过程。机器视觉应用系统的组成框图如图 7.1 所示。

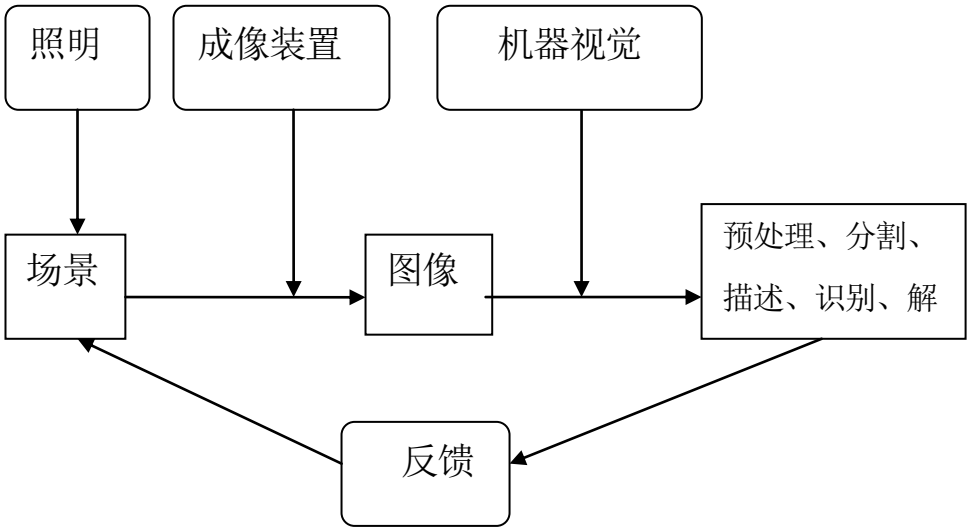


图 7.1 机器视觉应用系统组成框图

7.1.3 计算机视觉应用系统的应用领域

机器视觉的领域非常广阔，包括工业生产线上零件的识别与定位，工业产品质量检测（如钢板在线检测系统），自主移动机器人的导航（如星球机器人），遥感图像的分析，医学图像的分析，安全鉴别、监视与跟踪系统，国防系统，动画制作，考古等方面。

7.1.4 开放源代码的视觉函数库 OpenCV 简介

1） 什么是 OpenCV

OpenCV 是 Intel®开源计算机视觉库。它由一系列 C 函数和少量 C++ 类构成，实现了图像处理和计算机视觉方面的很多通用算法[25]。

2） 重要特性

OpenCV 拥有包括 300 多个 C 函数的跨平台的中、高层 API。它不依赖于其它的外部库——尽管也可以使用某些外部库。OpenCV 对非商业应用和商业应用都是免费（FREE）的。 OpenCV 为 Intel® Integrated Performance Primitives （IPP） 提供了透明接口。 这意味着如果有为特定处理器优化的的 IPP 库， OpenCV 将在运行时自动加载这些库。

7.2 计算机视觉在参赛机器人上的应用

尽管本章的内容有很多和第五第六章密切相关，但由于计算机视觉的软件实现相对独立， 本文将在本章专门论述计算机视觉在机器人上的应用。

7.2.1 利用计算机视觉搜索并定位目标

在本届比赛中，得分相对比较集中，中间柱子上的白块分值达到 6 分，是比赛双方必争之物，对白块的争夺必定非常激烈，甚至比赛双方都可能抢在对方之前先将白块从初始位置—柱子上破坏，白块很可能掉在地上。于是，我们有必要采取措施使机器人有能力捡地上的白块。这里的关键技术，就是计算机视觉[26]。

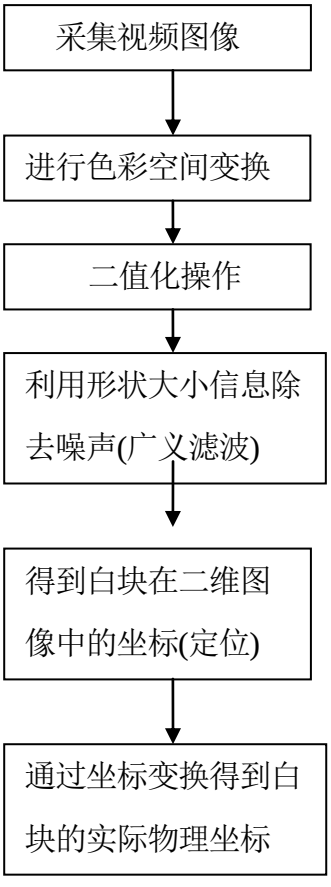


图 7.2 利用计算机视觉实现白块目标定位的过程

下面介绍目标定位中的关键函数的功能。

1） void InitializeCapture();

初始化图像采集设备，分配内存。

2） IplImage*cvQueryFrame(capture);

从摄像头句柄获取一帧视频图像 captur 是摄像头句柄，返回图像在内存中的指针。

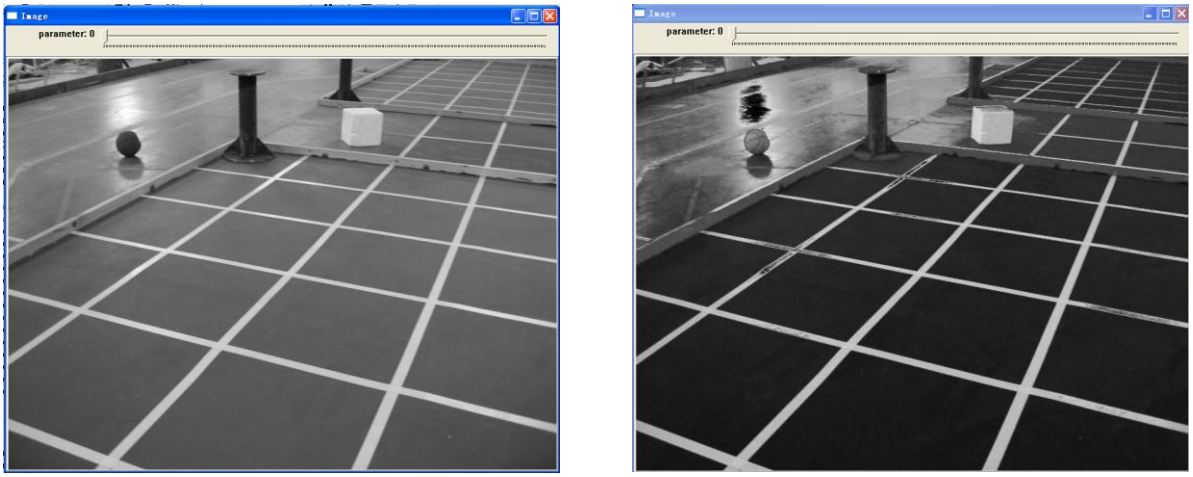
3） void cvResize(frame, src, CV_INTER_LINEAR）；

对原始图像进行缩放函数，提高帧率， frame 是源图像指针 src 是变换后的图像指针 宏 CV_INTER_LINEAR 代表线性变换。

4） void cvCvtColor（src, hsv, CV_HSV2BGR）；

将图像进行 HSV 转到 BGR 空间的变换函数，hsv 是变换后的图像指针 src 是变换前的图像指针，宏 CV_HSV2 标志 BGRHSV 转 BGR。

需要指出的是，实际上原始图像就是在 BGR 空间，而作者在程序中是将其当作 HSV 空间的三个通道转到了 BGR 空间。这个颜色空间转换的方法，是在作者使用一个测试程序测试各种不同的非线性变换后找到的一种针对白块搜索的比较稳定的方法。效果见图 7.3。



（a）灰度变换的效果　　（b）本文中使用的变换效果

图 7.3 两种色彩空间变换的效果对比

5） void cvCvtPixToPlane（hsv,h,s,v,0）；

提取 hsv 中的各个单通道图像。Hsv 是三通道的图像指针，h,s,v 分别是个自单通道的图像指针。

6） void thresh（s, THRESH）；

将图像进行二值化，s 是被二值化的图像，THRESH 是阈值。

7） void XErode（ WIDTH, HEIGHT, s, NEAR_GEZI, FAR_GEZI）;

横向去掉较短的黑线片断，WIDTH 是图像的宽, HEIGHT 是图像的高, s 是图像指针, 采用了透视的思想，NEAR_GEZI 是最近处的片断宽度阈值, FAR_GEZI 是最远处的片断宽度阈值。

8） void YErode（ WIDTH, HEIGHT, s, NEAR_GEZI, FAR_GEZI）;

纵向去掉较短的黑线片断，WIDTH 是图像的宽, HEIGHT 是图像的高, s 是图像指针, 采用了透视的思想，NEAR_GEZI 是最近处的片断高度阈值, FAR_GEZI 是最远处的片断高度阈值。

9） void XCombine（ WIDTH, HEIGHT, s, NEAR_GEZI, FAR_GEZI）;

将被干扰的间断的目标区域横向连接起来。WIDTH 是图像的宽, HEIGHT 是图像的高，s 是图像指针，采用了透视的思想，NEAR_GEZI 是最近处的片断宽度阈值, FAR_GEZI 是最远处的片断宽度阈值。

10） void YCombine（ WIDTH, HEIGHT, s, NEAR_GEZI, FAR_GEZI）;

将被干扰的间断的目标区域纵向连接起来。WIDTH 是图像的宽, HEIGHT 是图像的高，s 是图像指针，采用了透视的思想，NEAR_GEZI 是最近处的片断高度阈值, FAR_GEZI 是最远处的片断宽度阈值。

11） void XLongErode（WIDTH,HEIGHT,s,NEAR_FANGKUAI,FAR_FANGKUAI）;

横向去掉较长的黑线片断，WIDTH 是图像的宽, HEIGHT 是图像的高, s 是图像指针, 采用了透视的思想，NEAR_FANGKUAI 是最近处的片断高度阈值, FAR_FANGKUAI 是最远处的片断宽度阈值。

12） void YLongErode（WIDTH,HEIGHT,s,NEAR_FANGKUAI,FAR_FANGKUAI）;

纵向去掉较长的黑线片断，WIDTH 是图像的宽, HEIGHT 是图像的高, s 是图像指针, 采用了透视的思想，NEAR_FANGKUAI 是最近处的片断高度阈值, FAR_FANGKUAI 是最远处的片断高度阈值。

13） void x_serch（s,HEIGHT,WIDTH）;

横向搜索连续最宽的目标片断，WIDTH 是图像的宽, HEIGHT 是图像的高，s 是图像指针。

14） void y_serch（s,HEIGHT,WIDTH）;

纵向搜索连续最宽的目标片断。WIDTH 是图像的宽, HEIGHT 是图像的高，s 是图像指针。

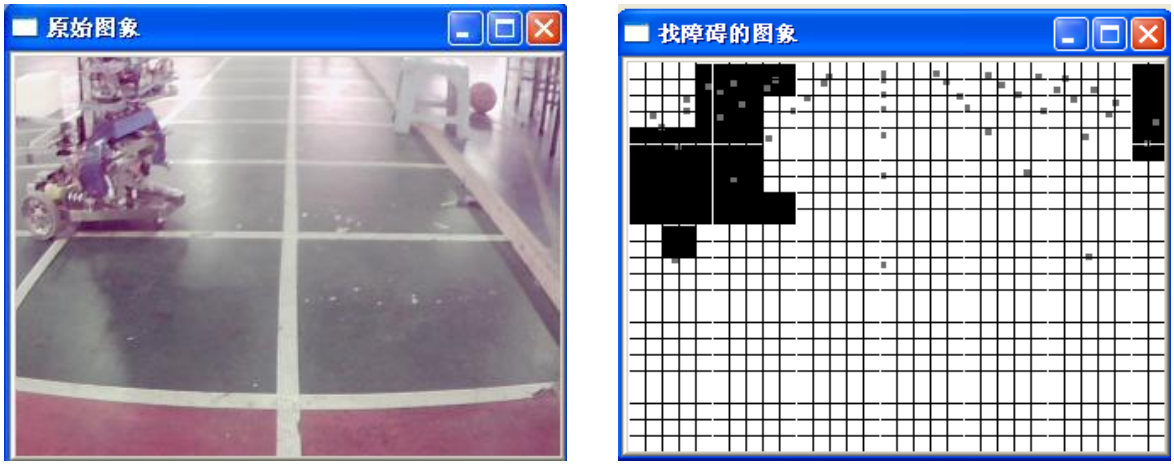
15） void Judge();

决策函数，判断是否存在白块以及白块在图像中的二维坐标。

16） void trans2real2（HEIGHT,WIDTH,1）;

将白块在图像中的坐标转到真实世界中相对于机器人的坐标。

下面介绍作者使用的标定方法。



（a）原始图象　　（b）参照点的描绘（灰色的点）

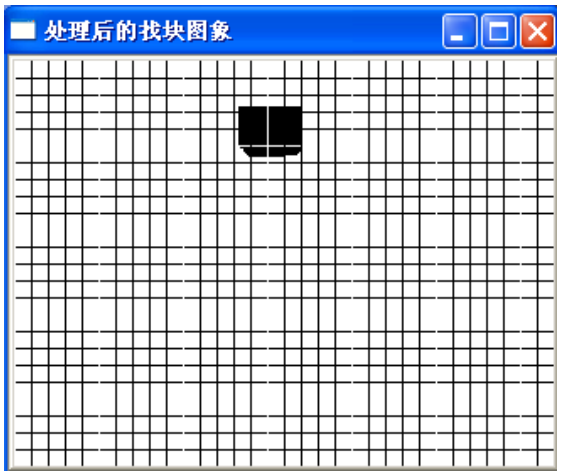
图 7.4 坐标变换示意图

如图 7.4 所示，这是一幅障碍定位的坐标，图 b 中的灰色的点是图 a 中白线交叉点所对应的点，而这些白线组成单位正是场地中的 50cm*50cm 方格，通过方格点的参照，我们可以将图象中坐标转到真实坐标。

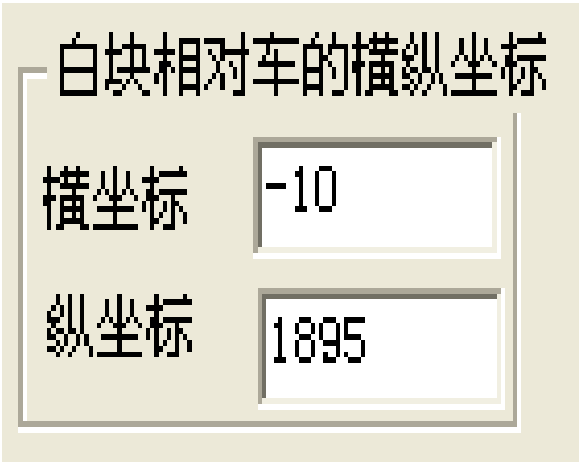
坐标转换的基本方法是：先根据目标在二维图像中的坐标找到离目标最近的几个参考点，再由这几个参考点的真实坐标和二维图像坐标插值计算出目标的真实坐标。利用这种可靠的标定方法，根据平时的反复测试，对目标的定位精度可以达到误差 5 厘米以内。图 7.5 展现了搜索白块并定位的过程和最终效果。

（a）原始视频图像

（b）色彩变幻后的图像



（c）定位白块的图像



（d） 白块相对车的横纵距离（单位：mm）

图 7.5 搜索白块并定位的过程和最终效果图

7.2.2 利用计算机视觉探障

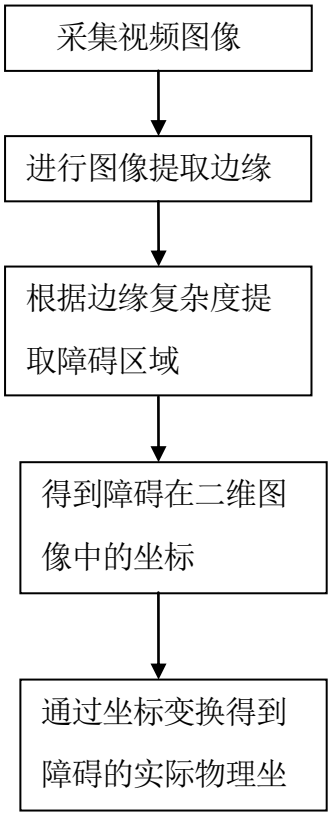


图 7.6 利用计算机视觉实现障碍目标定位的流程

如图 7.6 所示，图象的边缘提取这一步是关键步骤，图像的边缘提取有多种算法。由于传统的求取梯度的算法受噪声干扰较大，所以我们采用 Canny 算法进行边缘提取，并计算图像的梯度值。J Canny 证明了一维空间的指数滤波器的最佳性，并提出了边缘检测的准则。他还指出了最佳滤波实际上是用高斯函数的一阶导数来滤波的，并导出了二阶边缘检测最佳算子。由于 Canny 算子的良好特性，所以它已成为很多边缘检测器的比较准则。作者调用了 OpenCV 中的函数实现了这一功能。

1） void cvCanny（gray,edge, edge_thresh[0], edge_thresh[0]*3,3）;

gray 代表灰度图像，edge 代表提取边缘后存放结果的指针，edge_thresh[0]是提取边缘的阈值。

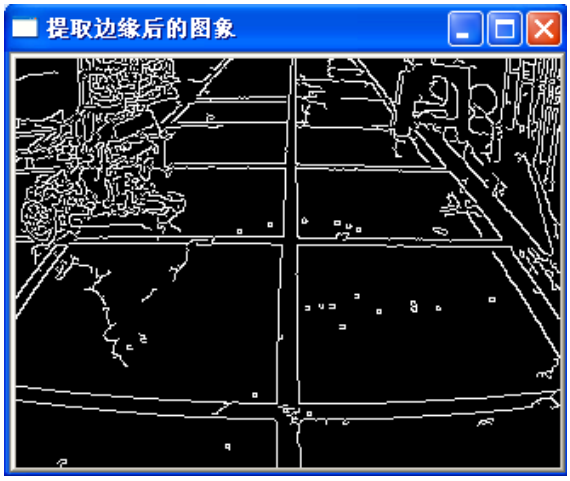
2） void detectComplex();

以 20*20 像素的方格为单位，对边缘信息较多的方格区域二值化。

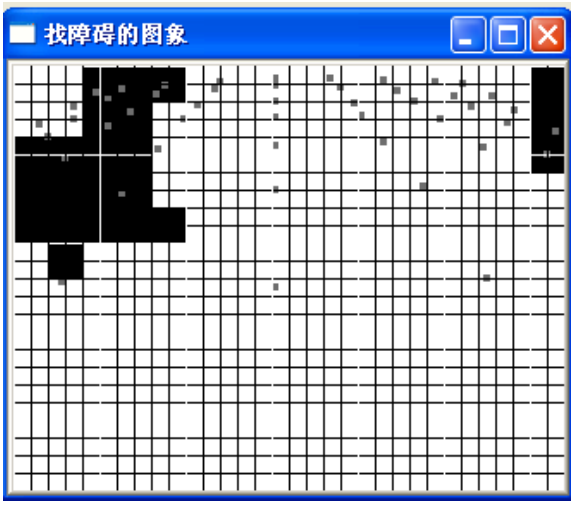
3） void localComplex();

用和定位白块类似的方法定位障碍向对于机器人的位置，并在电子地图中动态更新相应位置的障碍标志，在更新的电子地图的基础上，机器人重新规划一次路径。

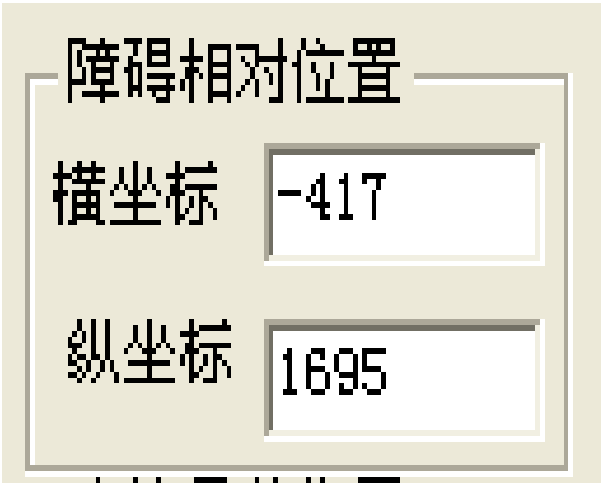
（a） 原始图像



（b） 边缘提取后的图像



（c）定位障碍的图像



（d）白块相对车的距离（单位：mm）

图 7.7 利用计算机视觉实现障碍定位的示意图

7.2.3 利用计算机视觉辅助二次定位

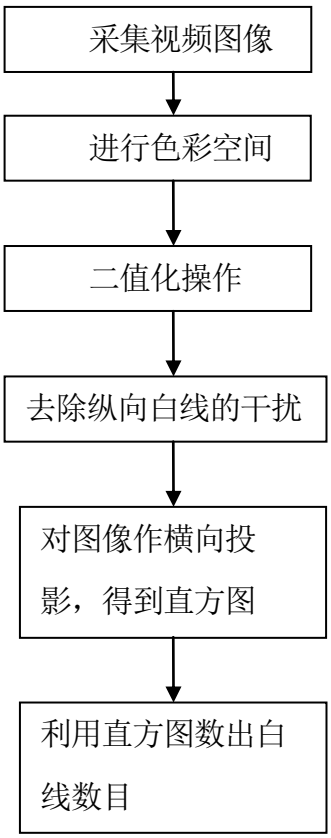


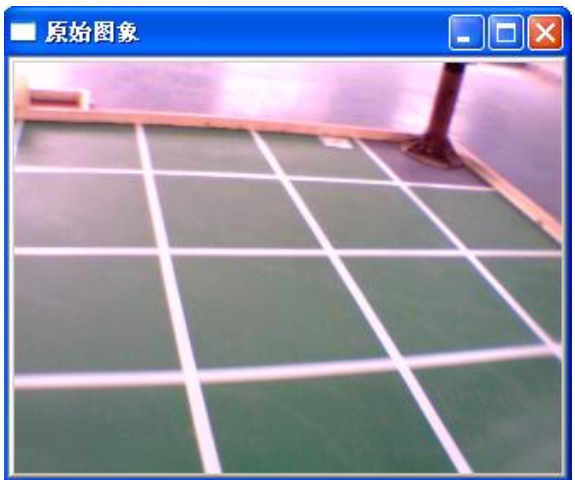
图 7.8 利用计算机视觉实现数线辅助定位的流程图

由于在数线的函数中，同样调用了很多先前介绍的白块定位中的很多基本函数，在此不再重复说明，下面仅就最主要的数线函数加以说明。

1） int CountLineNum();

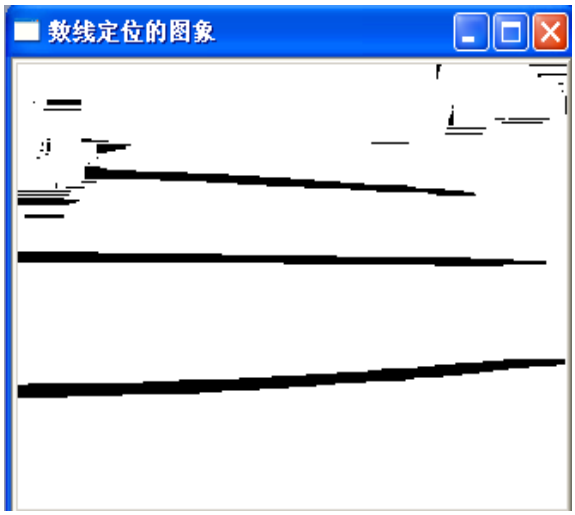
该函数通过直方图信息，对直方图进行平滑滤波，求出局部极值的个数，即返回了线的数目。

在得到机器人前方距场地边缘的线数后，机器人就可以在这些地点进行二次定位，纠正因为长距离里程计带来的误差。

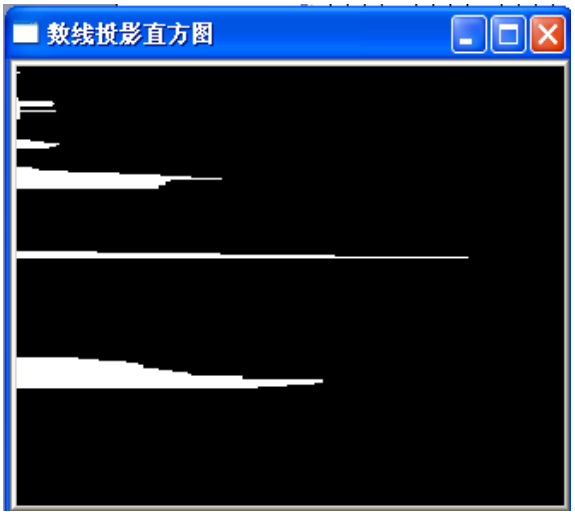


（a）原始图像

（b）色彩变换后的图像



（c）提取横线后的图像



（d） 直方图

图 7.9 利用计算机视觉实现数线辅助定位的示意图

7.3 计算机视觉程序架构和平台说明

与视觉相关的程序是在 VC6.0 上以 openCV 的函数库为基础实现的[27]。作者设计了两种程序的架构分别作了比较[28] [29]。

7.3.1 多线程的程序架构

主要代码如下所示（以 windows 平台下的代码为例）：

```
CWinThread *pThread, *pThread2,*pThread3, *pThread4;
pThread1 = AfxBeginThread(ProcessSearchBox,this); //开启找块线程
pThread2 = AfxBeginThread(ProcessAvoidCollision,this); //开启避障线程
```

```
pThread3 = AfxBeginThread(ProcessLocalize(),this); //开启定位 线程
pThread4 = AfxBeginThread(ProcessCom,this);
pThread1->SetThreadPriority(2);//THREAD_PRIORITY_TIME_CRITICAL
pThread2->SetThreadPriority(0);//THREAD_PRIORITY_HIGHEST
pThread3->SetThreadPriority(0);//THREAD_PRIORITY_HIGHEST
pThread4->SetThreadPriority(-1);//THREAD_PRIORITY_LOWEST
```

宏 THREAD_PRIORITY_HIGHEST 代表非实时最高优先级，THREAD_PRIORITY_TIME_CRITICAL 代表实时优先级，作者尝试了各种线程优先级的组合方式，相比而言，上述优先级组合方式效率最高。在测试过程中，串口监听线程的优先级设定最为关键，该线程若设为正常优先级 THREAD_PRIORITY_NORMAL，将拖慢整个程序的运行效率。

表 7.1 串口优先级对程序性能影响的比较

串口线程优先级	THREAD_PRIORITY_NORMAL	THREAD_PRIORITY_LOWEST
150 帧图像处理时间	45．1 秒	27.4 秒

7.3.2 单线程的程序

主要代码如下所示（以 windows 平台下的代码为例）：

```
CWinThread *pThread;
pThread = AfxBeginThread(ProcessCom,this);
pThread->SetThreadPriority(-1);//THREAD_PRIORITY_LOWEST
for(;;)
{Localize();//定位
SearchBox();//找块
AvoidCollison();//避障}
```

采用上面的程序架构，程序运行时间反而比多线程更快，出乎作者的意料。

表 7.2 单线程和多线程对程序性能影响的比较

程序架构	单线程	多线程
150 帧图像处理时间	24.7 秒	27.4 秒

7.3.3 linux 平台下的程序实现

考虑到程序调试和串口编程的方便，作者在 windows 下的程序是基于 MFC 的对话框的，在 linux 下对应由 Qt3.1 也实现了同样的功能，但是，考虑到 linux 下的串口编程在控制台程序下就非常方便，所以，就使用了控制台程序。关于算法的代码和程序的架构，几乎和 windows 下一致。

1）Linux 下开启一个线程

pthread_t　id;int ret;

ret = pthread_create（&id,NULL,ComProcess ,NULL）;

2）Linux 下串口通信的函数

（1） int OpenAdrPort（char* sPortNumber）

打开串口，输入为串口号

（2） int ReadAdrPort（char* psResponse, int iMax）

非阻赛读取串口。输入 psResponse 为存储的地址头指针，iMax 为最大输入数。

（3） int WriteAdrPort（char* psOutput）

写入串口，输入为要写入的内存地址指针

7.4 本章小结

本章论述了自动机器人基于视觉的避障、定位白块目标以及辅助定位的方法和实现[30] [31]，并介绍了 linux 和 windows 平台的计算机视觉程序开发。机器人视觉的开发，一方面是为了今年的比赛，另一方面也是为今后的比赛做技术积累。

总结和展望

从2007年11月加入机器人工作室到今天，已经有整整七个月的时间了。切身的体会告诉我，竞赛机器人的设计制作是一项体现控制技术与实战技术的工程性项目，它除了需要采用比较先进的技术外，还具有对抗性强，不可预测性高，策略变化大等特点。

作为一名自动化专业的学生，在大自动得分机器人的设计制作过程中，发现机器人控制技术涉及到了专业知识、硬件水平、软件研究等多方面知识。每一个机器人控制部分的开发调试，传感器的选择安装，电路的设计乃至焊接都是对自身专业水平的一次实战考验。对于机械知识相对薄弱的自动化专业学生来说，机器人制作的过程也是一个取长补短的过程，可以学到很多机械结构的知识和技术。结合本届大赛中主力得分机器人——双升降翻转型机器人的电气控制和智能系统部分，对所作内容总结如下：

1） 利用 ADS1.2 、CodeWarrior、MFC、openCV 等开发软件和工具包的使用，用 VC 进行高层算法仿真，底层采用 C 语言进行相关控制程序的编写，并最终搭建了 耦合度较低，扩展性较强的软件程序框架；

2）利用陀螺仪、码盘和光纤传感器，设计和实现了基于多传感器的机器人定位方法；

3） 利用经典的 PID 控制算法，作者提出了直线结合圆弧的机器人运行路径的跟踪方法，运用于自动机器人上效果明显，避免了两段直线路径间平滑稳定过渡的问题。顺利完成了机器人夹块得分的功能；

4） 本文作者最终完成了自动机器人连续空间自主路径规划系统，展开并完成了基于视觉的探障传感系统的避障路径规划，在机器人的实际使用中收到了良好的效果；

5） 采用计算机视觉技术，使用有效的标定方法实现了对场地上白块和障碍的较精确定位（误差 100mm 以内），以及机器人的二次定位。

在机器人控制系统的开发过程中，我们努力解决问题的同时，还是留下了很多有待进一步解决的问题。在此，同时进行简单总结。

有待进一步研究的工作：

1） 移动机器人中的多传感器信息融合问题

虽然本文在移动机器人的设计制作中用到了很多传感器，但对多传感器信息融合技术研究，如何将多个声纳传感器、激光测距传感器、摄像头等多种传感器在决策层上进行信息进行融合的方面有待开发。实际上利用视觉进行机器人的重定位比本文使用的坐标纠偏方法具有更高的通用性和前沿性。

2） 移动机器人移动机器人的导航定位问题

基于较为简单的数学模型和白线纠偏的信息融合算法，机器人虽然实现了一定的自主定位的能力，但是，应该看到目前有很多定位的更好的方法，比如使用卡尔曼滤波器对数据进行融合，再用马尔科夫模型进行定位等等方法，都可以在以后的比赛中继续研究运用，目的是进一步提高算法的通用性和可靠性。

3） 移动机器人移动机器人的路径规划问题

本文使用的路径规划办法，关键在于启发函数的选择，需要很多的先验知识，对场地的特征很有依赖性，最后的拟合过程，本文采用了直线加圆弧的方法来实现，实际上是一种二次曲线路径，在一些出场合，这种曲线容易产生振荡，不利于机器人的行走控制，我们最后不得不放弃了这种优化。为了使路径有更好的连续性，需要更好的拟合方法，比如四次曲线路径生成方法。另外，我们已经看到像香港科技大学等队，已经做了基于人工势场的算法来进行路径规划，并收到了很好的效果。

4） 移动机器人移动机器人的轨迹跟踪问题

两轮差速机器人的轨迹跟踪算法是一个非完整约束问题，使得对路径的精确跟踪变得很困难，我们在调试中用了很多试凑参数的方法。可以考虑三轮对称的底盘结构，因为理论上来说，这种结构的控制是一个完整约束问题，即使实现上的控制难度可能更大，但更适合实现轨迹跟踪。

5） 计算机视觉中标定方法的改进

本文的标定，并没有使用雅可比矩阵的坐标变换方法，而是使用了简单的参照点加线性插值的方法，实际的定位精度受一定限制，而且不方便标定。应该寻找更方便有效和方便的标定方法。

6） 移动机器人移动机器人的硬件系统精简问题

目前的系统底层对陀螺仪和码盘的数据采集都是用 ARM 来实现，实际上 CPLD 更适合这类应用。另外，机器人的视觉使用了精小的笔记本，但是，DSP 或者高端的 ARM 系列芯片更适合机器人系统的集成。

7） 移动机器人的软件平台的问题

一直以来，由于技术积累不够，另一方面是担心操作系统影响处理器处理的实时性，我们都的主脑程序是在处理器裸机上直接运行的，没有使用操作系统。这样的程序架构也是机器人实现智能性的软件体系上的瓶颈。今后，可以尝试使用操作系统实现多线程，增加更多的任务处理的余地，并精简硬件，提高系统的可靠性。

参 考 文 献

[1] 王民忠.缔造传奇--机器人大赛揭秘[M].北京:科学出版社,2004:110～130.

[2] Rolando, Introduction to Autonomous Mobile Robots [J].Massachusetts: Massachusetts Institute of Technology, 2004.

[3] 李佳宁,易建强.移动机器人体系结构研究进展[J].机器人,2003,25(7):756~760.

[4] 曹志强.未知环境下多机器人协调与控制的队形问题研究[D].中国科学院自动化研究所,2002:8~9.

[5] R•西格沃特 I•R•若巴什克.自主移动机器人导论[M].西安:西安交通大学出版社,2006.

[6] 周立功,张华,等.深入浅出 ARM7-LPC213x/214x.[M].北京:北京航空航天大学出版社,2006.

[7] Jean J. Labrosse.嵌入式实时操作系统 uC/OS-II[M].邵贝贝译.北京:北京航空航天大学出版社, 2002.

[8] 童诗白,华成英.模拟电子技术基础[M].北京:高等教育出版社, 2001.

[9] 罗蓉,徐红兵,田涛.复杂系统多传感器数据融合技术及应用研究[J].中国测试技术.2006,32(4):17～21.

[10] 吴伟,刘兴刚,王忠实.多传感器融合实现机器人精确定位[J].东北大学学报（自然科学版）.2007,28(2):161～164.

[11] 张嗣瀛,高立群.现代控制理论[M].北京:清华大学出版社,2006.

[12] 张培仁,张志坚,郑旭东,张华宾.基于 16/32 位 DSP 机器人控制系统设计与实现[M].北京:清华大学出版社,2006.

[13] 沈猛.轮式移动机器人导航控制与路径规划研究[D].西北工业大学博士论文,2006:26～65.

[14] 童艳.移动机器人路径跟踪控制方法研究[D].西北工业大学博士论文,2006:152～159.

[15] 沈猛,徐德民,崔荣鑫.基于动态逆的轮式移动机器人非时间路径跟踪控制[D].弹箭

与制导学报,2005,25(3):326～365.

[16] 祖莉,王华坤,范元勋.户外小型智能移动机器人运动路径跟踪控制[J].南京理工大学学报,2003,27(1):56～59.

[17] 南景富.轮式移动机器人的路径规划和跟踪控制[D].黑龙江科技学院,2003:12～20.

[18] 胡跃明,周其节,裴海龙.非完整控制系统的理论与应用[D].控制理论与应用,1996,13(1):1～10.

[19] Amin.Jayesh,Fast and Efficient Approach to Path Planning for Unmanned Vehicles[J]. Guidance,Navigation,and Control Conference Proceedings,2006:86～115.

[20] 章乃孝,裘宗燕.数据结构--C++与面向对象的途径[M].北京:高等教育出版社,2001.

[21] M.de Berg M.van Kreveld, M.Overmars O.Shwarzkopf. 计算几何—算法与应用（第 2 版）[M].北京:清华大学出版社, 2005.

[22] [美]Philip J.Schneider David H.Eberly [M].计算机图形学几何工具算法详解.北京:电子工业出版社,2005.

[23] Vendittelli M.Anewtool for mobile robot perception and planning.Joural of Robotic System[J],1997.4(3):179～197.

[24] 李智军,罗青.基于混合结构的机器人 Agent 控制结构研究[J].计算机工程与应用,2003.21:90~94.

[25] 李介谷.计算机视觉的理论和实践[M].上海:上海交通大学出版社, 1991.

[26] 章毓晋.图像工程[M].北京:清华大学出版社,2007.

[27] 刘瑞祯,于仕琪.OpenCV 教程--基础篇[M].北京:北京航空航天大学出社,2007.

[28] 求实科技.数字图像处理典型算法及实现[M].北京:人民邮电出版社, 2006.

[29] 倪继利. linux 内核分析及编程[M].北京:电子工业出版社, 2005.

[30] 李磊.移动机器人系统设计与视觉导航控制研究[D].中国科学院自动化研究所,2003:31～45.

[31] Sebastian:A Second-Generation Museum Tour-Guide Robot,Proceedings of IEEE International Conference On Robotics and Automation,1999.4(3):186～225.

[32] From topological knowledge to geometrical map. Franiois Tie` che, Joural of Robotic System[J],2000.4(1):39～47.

附 录 A 英文资料原文

From topological knowledge to geometrical map

Abstract

The behavioral approach to robot navigation,characterized by a representation of the environment that is topological and robot-environmental interactions that are reactive, is preferable to purely geometrical navigation because it is far more robust against unpredictable changes of the environment.Nevertheless,there is still a need to obtain geometrical maps.This paper considers a geometrical map reconstruction that relies on the topological knowledge and uses redundant odometric measurements taken while the robot moves along the paths of the topological map.Five methods are presented and compared,in experiments involving a Nomad200 mobile robot operating in a real environment. （ 1999 Published by Elsevier Science Ltd. All rights reserved.

Keywords: Autonomous mobile robots; Knowledge representation; Least-squares method; Robot navigation; Robot vision; Robot control; Robotics

A map of the environment is needed for a mobile robot to carry out navigation tasks. Various map representations and numerous map construction approaches have been considered. First there are geometrical maps, which integrate sensed data in a single frame of reference. In the Certainty Grid approach （Elfes, 1989）, the certainty about the existence of obstacles, detected by sonar, is reported in a grid map. In another approach, Crowley （1989） constructs geometric feature maps of line segments by means of an extended Kalman filter. Therefore these geometrical maps give an accurate description of the environment, and can be used to compute optimal robot paths. However they provide a poor interface to symbolic planning units, use large amounts of data, and require a complex process in order to maintain the map consistency in large environments.

Topological maps overcome some of these limitations. They represent the environment as neighborhood rela tionships of distinctive places. Places are differentiated by their sensing signatures, such as sonar signatures （Kurz, 1993） or sonar and vision signatures

（Kortenkamp and Weymouth, 1994）. In another approach, Thrun and BuK cken （1996） use Voronoi squeletonizing to extract identical topological regions from a grid map, and then create a topological map.

The construction of a map by combining landmark and topological information has been performed by the use of Kalman filtering. In the approach of Bulata et al. （1996）, Kalman filtering is applied incrementally to account for uncertainties, both at the landmarks and the topological level, whereas an alternative approach （HeH bert et al., 1996） reserves Kalman filtering at the level of a local map only, and proceeds by relocation-fusion and grouping at the global level.

Most of the approaches described so far fall into the class of "sense-map-plan-act" robot architectures, that are known to be ineffcient at reacting quickly to unpredictable changes in a dynamic world. In contrast the class of behavioral architectures allows the robot to move around safely, even in dynamic environments, by means of a set of individual behaviors that provide strong robot/environment interactions. Topological maps are well suited to represent these interactions （Mataric, 1990）. Such a topological map, known as a cognitive map, has been proposed by Kuipers and Byun （1991）. In this approach, distinctive places correspond to the activation of a particular class of behaviors, called self- positioning behaviors.These behaviors control the robot's movements and lock it into a specific pose relative to particular environmental characteristics: the self-positioning site. Also, neighborhood relations are expressed by behaviors that the robot between two self-positioning sites.

Although the behavioral navigation generally reacts well to changes in the environment, the associated topological map is completely useless in the case of a loss of behavioral stimulation. These limitations can be avoided by extending the topological map with additional geometrical information. As a benefit, such a new map allows one to determine paths that have not yet been explored. It also provides an interface which is more easily understood by a human operator.

This paper presents a way of extending the knowledge of a topological map of self-positioning sites by the construction of a consistent associated geometrical map. This construction proceeds by integrating recorded odometric paths. Five methods are proposed to integrate these paths into a single frame of reference according to the topological map.

The paper is organized as follows: After a description of the mobile robot architecture in Section 2, Section 3 describes the behaviors that are used in connection with the topological map. Then, Section 4 formally describes the topological map, and Section 5 explains how the geometrical information is added to the map. Section 6 presents the five methods used to construct the consistent geometrical map. The experimental results are shown in Section 7, and Section 8 concludes this paper.

2.Mobile robot architecture

The robot architecture （HuK gli et al.,1994） follows the principles of the behavioral approach. It is composed of four hierarchical layers: sensorimotor, behavioral, sequencing, and planning. The lowest one, called the sensorimotor layer, is based on control theory and on signal processing. It is responsible for the elementary movements of the robot, and processes data acquired by the sensors. The second is the behavioral layer, composed of a set of behaviors that on one hand control the robot with respect to environmental characteristics, and on the other hand extract measures of the world in order to feed the robot with an internal world representation: the topological map. The sequencing layer implements tasks, which are described as sequences of behaviors. Its kernel is formed of a state automaton that activates the elementary behaviors, based on the interpretation of both the status of the various behaviors and the parameters transmitted by the planning layer. This latter activates and parametrizes the sequencing tasks according to specications given by a human operator, to the information of the topological map and to the feedback from the sequencing tasks.

The architecture is implemented in the form of a development environment, which encompasses a Nomad200 mobile robot （Nomadics, 1992）moving in a room environment, a set of different sensors, dedicated vision hardware, a collection of sensory-based behaviors, and a versatile control unit. The successful implementation of several tasks in a real environment testifies to the validity of this architecture （Tie‘ che et al., 1995）

3.Behaviors

The behavioral layer comprises various behaviors. Some of these are directly related to the self-positioning sites, and others to the displacements between sites.

Two kinds of behavior are related to the sites: the self-positioning behaviors, which move the robot into sites, and the localization behavior which identifies the sites. Among the self-

positioning behaviors, the homing on corner behavior （Facchinetti and HuK gli, 1994） controls the robot to adopt a fixed pose, defined with respect to particular configurations of the environment: salient corners and reflex corners. In the specific pose of interest in this paper, the robot is oriented towards the corner and is located on the corner symmetry line, at a fixed distance from it. This behavior receives range profiles from the Sensus500 structured light vision system, and moves the robot such as to minimize the errors between a reference corner and the observed corner. Another vision-based self-positioning behavior is the homing on target behavior, which positions the robot with respect to a pair of visual landmarks.

The behavior that distinguishes the different homing sites is called localization behavior （Tie‘che et al.,1996）. It uses a gray-scale video camera, pointing to the ceiling, and identifies a site by comparing snapshots taken when the robot is standing at a self-positioning site with a set of reference images stored in a database. It returns the identification of the unknown place. The combination of both a homing behavior and the localization behavior allows the distinctive places to be defined very accurately, and in a non-ambiguous way.

The behaviors that are related to the robot displacements between sites are called the “move to” behaviors. One of these behaviors controls the robot to follow a wall detected by means of the Sensus500 structured light vision system.Another is activated when a reactive landmark is seen. This moves the robot towards the landmark, and stops it at a fixed distance from the site.

4.Topological map

Topological maps represent the environment in terms of neighborhood relationships between distinctive places. Formally, the topological map consists of a graph G=（V,E）,where V={v1 , …,vN}is the set of N nodes,and E={eij}={（vi,vj）} the set of M edges. It may be considered in two ways. From the topological point of view, it is centered on a symbolic representation of the environment. From the robot resources point of view, the map is based on the interactions of robot sensors and actuators performed by the behaviors. In the frame of this work, each node corresponds to a self-positioning site, and an edge to the displacement of the robot between two such sites.The behavior associated with the nodes are the homing on corner and the localization behaviors, while a move to behavior goes with edges.

The choice of corners as environmental characteristics for self-positioning behaviors is justified by the fact that the corners are easily detected, are represented in a large number in man-

made environments and appear in stable parts of the environment such as tables, walls, doors, etc. This gives the map high accuracy and good stability.

5.Addition of geometrical information

This section considers the extension of the topological map by adding geometrical information. The idea is to record the odometer path while the robot moves, between sites, along the edges of the topological graph. The result is a series of odometric paths, which must be integrated to form a consistent global map.

More precisely, the topological map is built by moving the robot, manually or with adequate behaviors, from corner to corner. This building process provides a sequence of visited nodes that is stored in a list: $\Sigma = \{v_i\}_{1 \leqslant i \leqslant M+1}$. The robot pose $p = (x, y, \varphi)^t$ is a three dimensional value that defines the position and the turret orientation of the robot, in a single frame of reference.

The odometric paths provide geometrical relations between the poses the robot takes at the self-positioning sites. A path w_{AB} between two sites $A\,(x_A, y_A, \varphi_A)^t$ and $B\,(x_B, y_B, \varphi_B)^t$ is represented by a three-dimensional vector $w_{AB} = (d_{AB}, \alpha_{AB}, \beta_{AB})^t$ （Fig. 1）.

Assuming the robot is in pose A, α_{AB} is the rotation angle that brings the turret to point towards the position B, d_{AB} is the distance between the two positions A and B, and β_{AB} is the rotation angle that aligns the turret to the pose B. A compounding operation is defined to express a pose p_B , in term of a pose p_A and a path w_{AB} linking p_A and p_B. This compounding operation is denoted as:

$$p_B = p_A \oplus r_{AB}.$$

$$\begin{pmatrix} x_B \\ y_B \\ \varphi_B \end{pmatrix} = \begin{pmatrix} x_A \\ y_A \\ \varphi_A \end{pmatrix} \oplus \begin{pmatrix} d_{AB} \\ \alpha_{AB} \\ \beta_{AB} \end{pmatrix}$$

$$= \begin{pmatrix} x_A + d_{AB}\cos(\varphi_A + \alpha_{AB}) \\ y_A + d_{AB}\sin(\varphi_A + \alpha_{AB}) \\ \varphi_A + \alpha_{AB} + \beta_{AB} \end{pmatrix}. \qquad (1)$$

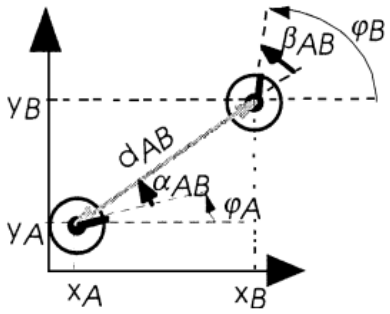


Fig. 1. The path between two robot poses A and B is defined by the three-dimensional vector $(d_{AB}, \alpha_{AB}, \beta_{AB})$.

The compound operation is associative on the right $p = ((p_0 \oplus w_1) \oplus w_2) \ldots) \oplus w_k$ and a sequence of compounding operations is denoted as

$$p = p_0 \bigoplus_{i=1}^{k} w_i. \tag{2}$$

In the same way, the inverse compounding operation $p_B \star P_A = w_{AB}$ expresses the path between two sites in terms of their poses.

$$\begin{pmatrix} d_{AB} \\ \alpha_{AB} \\ \beta_{AB} \end{pmatrix} = \begin{pmatrix} x_B \\ y_B \\ \varphi_B \end{pmatrix} \star \begin{pmatrix} x_A \\ y_A \\ \varphi_A \end{pmatrix}$$

$$= \begin{pmatrix} \sqrt{(x_B + x_A)^2 + (y_B - y_A)^2} \\ \arctan((y_B + d_A)/(x_B + x_A) - \varphi_A \\ \varphi_B - \arctan((y_B - y_A)/(x_B - x_A)) \end{pmatrix}. \tag{3}$$

The inverse path can also be defined: $\star w_{AB} = w_{BA}$. This implies that if a path is known, the inverse path can be computed.These compounding operators are close to those used by Lu and Milios （1997）, but differ because the paths are not defined in the same way.

6. Consistent geometrical map construction

Given the topological map and the associated information （the M measured geometrical paths w_{ij}^m, and the sequence of explored nodes Σ） the geometrical map-building problem is to determine N -1 robot poses $\bar{p}_i = (\bar{x}_i, \bar{y}_i, \bar{\varphi}_i)^t$ in a single coordinate system. One pose is given

a priori, and defines the origin of the system. Arbitrarily, the pose of the first explored node is chosen: $\rho_{\Sigma(1)} = (0, 0, 0)^t$.

Five methods of solving this problem are proposed below.

6.1. M1: Path integration along the exploration sequence

This method takes the nodes from the exploration list one by one, and "nds their poses by a simple integration of successive paths.

Formally, the pose $\hat{p}_{\Sigma(l)}$ of the node $\Sigma(l)$ can be expressed by compounding the origin of pose with the sequence of paths joining it to $\Sigma(l)$:

$$\hat{p}_{\Sigma(l)} = p_{\Sigma(1)} \bigoplus_{k=2}^{l} w^{m}_{\Sigma(k-1)\Sigma(k)}. \qquad (4)$$

As soon as circuits appear in the graph, nodes are visited more than once; hence their poses are computed several times. In order to assign a single pose to each node, this method keeps only the first computed pose, and discards the remaining ones.

The complexity of M1 is O（M）, where M is the number of edges.

6.2. M2: Path integration without circuits along the exploration sequence

This method also takes the nodes from the exploration list one by one. The pose is found by integration of successive paths, but when a circuit is closed on the explored sequence, the integration is interrupted and restarted from the first node belonging to the circuit. In this case, one pose is assigned to each node.

The complexity of M2 is O（M）.

6.3. M3: Path integration along the minimum distance tree

This method determines the pose of a node by compounding the original pose with a sequence of paths. In the graph, several sequences possibly link the origin to the current node. The chosen sequence is the one that has the minimum distance cost, defined as the sum of the distance d of each path along the sequence. This method finds the minimum spanning tree for a given root.

The complexity of M3 is O（MN）.

6.4. M4: Path integration along the minimum orientation tree

This method determines the pose of a node by compounding the original pose with a sequence of paths. In the graph, several sequences may possibly link the origin to the current node. The chosen sequence has the minimum angular cost, defined as the sum of the angular variation $|\alpha| + |\beta|$ of each path along the sequence. This method finds the minimum spanning tree for a given root.

The complexity of M4 is O（MN）.

6.5. M5: Least-squares minimization

The least-squares method minimizes the error between the measured paths w_{ij}^{m} and the estimated paths $\hat{w}_{ij}$.The function to be minimized is:

$$f(\hat{w}) = (w^{m} - \hat{w})^{t} P(w^{m} - \hat{w})$$

where P is a matrix of weights.

The estimated relations can be expressed as a nonlinear function of the estimated poses: $\hat{w}_{ij} = \hat{p}_j \ominus \hat{p}_i$.Hence, the function to be minimized depends on the estimated robot poses f（pL ）. It is the minimum or maximum if its gradient is equal to zero.

$$\nabla f(\hat{p}) = \frac{\partial f(\hat{p})}{\partial \hat{p}} = 0. \qquad (5)$$

This provides a system of nonlinear equations with 3N unknown variables. It is solved by means of the Newton-Raphson iterative method.

The complexity of M5 is O（N^3 ）.

7. Experimental results

This section presents the geometrical map reconstruction for a real environment explored by a Nomad200 mobile robot. The results of the five methods are compared.

7.1. Exact map

The real environment is composed of 28 homing sites（11 reflex corners, 17 salient corners）, distributed over a 10 * 12 m surface （Fig. 2a）. In orde to compare the reconstructed maps of robot poses $\hat{p}_i = (\hat{x}_i, \hat{y}_i, \hat{\varphi}_i)^t$ an exact map of the robot poses $p_i^e = (x_i^e, y_i^e, \varphi_i^e)^t$ is measured. It is constructed in two steps. First, the corners are mapped by means of a precise measurement. Then, the robot pose with respect to a corner is established, by

averaging several measurements. Finally, these values are added to the precise map of the corners,in order to obtain the exact map of the robot poses.

7.2. Comparison of exact and estimated maps

After a rigid alignment transformation, the exact and the estimated maps are compared. The di!erence between the two maps is expressed as the root mean square of the distance Δd, and the di!erence of orientation $\Delta\varphi$, between corresponding site poses.

$$\Delta d = \sqrt{\frac{1}{N}\sum_{N}(x_i^e - \hat{x}_i)^2 + (y_i^e - \hat{y}_i)^2} \qquad (6)$$

$$\Delta\varphi = \sqrt{\frac{1}{N}\sum_{N}(\varphi_i^e - \hat{\varphi}_i)^2}\,. \qquad (7)$$

7.3. List of explored paths

The paths were measured by odometers while the robot was exploring its environment. Seventy-two paths between the twenty-eight self-positioning sites were measured. Fig.2b shows the compounding of the starting pose with the 72 paths, along the sequence of exploration. Note that if a node is visited more than once, it is represented by several site poses.

7.4. Estimated maps

Fig.3 compares the reconstructed geometrical maps with the exact maps for the five methods M1-M5. The edges correspond to the paths needed to build the estimated maps. Note that with methods M2, M3 and M4 many edges are not taken into account.

Obviously, the map provided by M1 is bad; those provided by M2, M3 and M4 show acceptable results, while M5 is excellent. The visual results are con"rmed by comparing the reconstruction errors reported in Table 1, expressed by the root mean square value of the distance and angular di!erences between the site poses of exact and reconstructed maps.

For every method except M5, the errors are accumulated along the paths. Thus the poses become less accurate as soon as they are far from the origin. Furthermore, a variation in the measures can signi"cantly modify the map. Since M5 is stable and very accurate, it would be preferred even if the processing time is longer.

Concerning processing time, Table 2 provides a comparison. First, it summarizes the time complexity of the different methods,where N stands for the number of nodes and M for the number of edges. Then, the table shows the e!ective computing times that were required for

building the map of Fig.2（N =28, M =72）.It is meant as a relative quantitative comparison of the methods, and clearly shows large discrepancies.Notice that the reported alues were obtained with Mathematica on a personal computer. They are by no means optimal, and their absolute value could be reduced. Nevertheless, in the presence of large maps with a large number of nodes （N）, if M5 is preferred, its O（N ） complexity calls for the use of special measures like graph division, if large computing times are not acceptable.

8. Conclusions

This paper shows how to extend the knowledge of a topological map of self-positioning sites by the construction of a consistent associated geometrical map.Five methods have been proposed to determine the robot poses in a single frame of reference, using the topological map knowledge and odometric measurements along the paths linking the robot poses. Four methods integrate paths according to di!erent strategies, and one uses a global minimization.

These geometrical map-building methods were implemented in a development environment involving a Nomad200 mobile robot and were tested on a map reconstruction problem with 28 self-positioning sites. The five methods were evaluated numerically by a compari son of the reconstruction errors, and graphically by comparing the maps they deliver with an exact geometrical map. The least-squares method shows the best accuracy and gives excellent results, even for a large environment.

Table 1
Difference between exact and estimated map

	M1	M2	M3	M4	M5
Δd (cm)	153.0	47.9	40.4	28.3	10.7
$\Delta\varphi$ (°)	28.4	11.1	5.3	5.7	2.0

Table 2
Time complexity and computing time

	M1	M2	M3	M4	M5
Complexity	$O(M)$	$O(M)$	$O(M.N)$	$O(M.N)$	$O(N^3)$
Time (s)	0.1	0.1	2	2	65

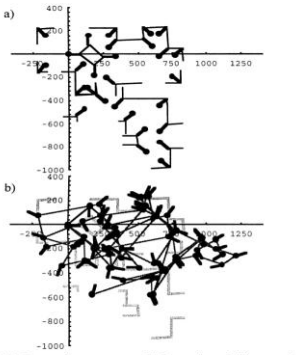


Fig. 2. (a) Exact robot poses map; (b) Integration of all measured paths.

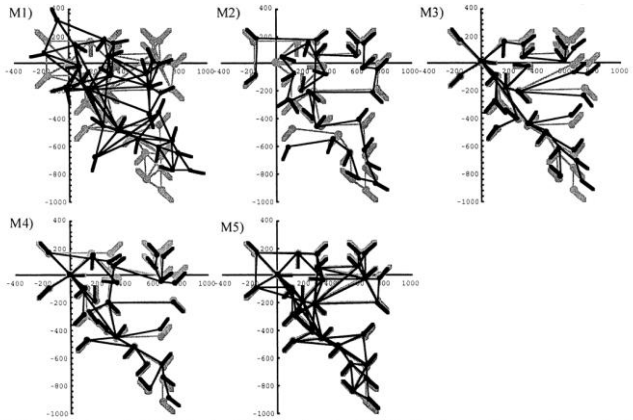


Fig. 3. Comparison of reconstructed (black) and exact (gray) maps for each method. The edges shown are the ones used for reconstruction.

附 录 B　英文资料翻译

从拓补地图到几何地图

摘要

基于行为的机器人导航方法，其特征是对环境的拓补表示和机器人与环境之间的交互。相比于纯粹几何学的导航方法，这种方法对不可预知变化的环境有更强的稳定性。然而，仍然有需要获得几何学的地图。本文论述了当机器人依赖拓补知识和采取冗余的里程测量沿着在拓补地图中的路径移动时几何地图的重建。以下介绍了五种方法，并在基于一个 Nomad200 机器人在真实环境中操作的实验中比较了这五种方法的效果。
（1999 Elsevier 科学公司出版 版权所有）

关键字：

自主移动机器人；知识表示；最小二乘法；机器人导航；机器人视觉；机器人控制；机器人学

机器人在执行导航任务时，需要一张环境的地图。现在已经出现了很多不同的地图表示方法和地图重建方法。第一种是把有意义的数据放入一张单一参考画面的几何学地图。在确定网格方法中（Elfes, 1989）,障碍存在的可能性被记录在一张网格地图当中。在另一种方法中，Crowley （1989）通过广义的卡曼滤波器构造了线片段的几何学特征地图。因此这些几何学的地图提供环境的正确描述, 而且能用来计算最佳的机器人路径。然而他们提供一个糟糕的接口给基于符号的单位路径规划,使用大量的数据, 而且为了要维持大的环境地图的连续性, 需要一个复杂的处理过程。

拓补地图则可以克服这些限制。他们可以表示环境中邻近的有明显特征的地点。这些地点由于它们明显的特征被区分出来了，比如声纳特征（Kurz,1993） 或声纳和视觉特征.（Kortenkamp 和 Weymouth,1994）。在另外的方式， Thrun 和 BuK cken （1996） 中使用 Voronoi 的骨架化吸取来自一张格子地图的同一拓补区域, 然后产生一张拓补地图。

通过卡曼滤波器，结合路标和拓补信息已经可以用来建立地图。在 Bulata et al.（1996）的方法中，递增地运用卡曼滤波器从路标和拓补两个层面来计算不确定度。而

另外的一种方法（HeH bert et al., 1996）仅仅从拓补地图的层面使用卡曼滤波器，并在全局上融合进重定位的方法。

大多数目前所描述方法都可以归为感知-地图-规划-行动的机器人体系,而这些体系在需要对动态环境不可预知的变化作出快速反应时被认为是不太有效的。相对的，基于行为的体系则有一套提供机器人和环境充分交互的个体行为，使机器人甚至在动态的环境中，也能安全地移动。拓补地图非常适合于用来表示这些交互（Mataric, 1990）。这种被称为感知地图的拓补地图，已经被 Kuipers and Byun （1991）提出。在这种方法中，显著特殊的地点对应于机器人一种特殊的行为，称之为“自主规划“的行为。这些行为控制机器人的运动并锁定机器人特定的姿态，这种姿态则与周围的环境特征有关，这些环境特征被称之为“自主规划点”。另外，当机器人处在两个 “自主规划点”之间时的相邻的关系也通过行为表示出来了。

尽管基于行为的机器人导航对于环境中的变化通常表现良好，然而当机器人失去行为刺激的情况下，相关的拓补地图完全失去作用。在拓补地图上添加额外的几何信息则可以避免这个限制。这张新地图的好处是可以允许机器人选择尚未被探测的路径。同时，它也向操作人员提供了更友好易懂的接口。

本文提供了一种通过建立连续相关的几何地图，来扩展拓补地图中的”自主规划点”的方法。建立地图的过程则是由历程计来实现。下文提出了五种方法根据拓补地图，把这些路径整合到一张单一的画面。

本文按以下各项展开：在第二节移动机器人体系的描述之后，第三节描述和拓补地图交互的行为。然后,第四节正式地描述拓补地图，而第五节解释几何学的数据如何被添加到地图。在第六节，展示了五种方法构建连续几何地图。第七节列出了试验的结果。第八节对本文作了总结。

２．移动机器人体系

机器人体系（HuK gli et al.,1994）遵照基于行为方法的原则，被分为四个层面：感知层，行为层，任务层，规划层。最低级的感知层，是基于控制理论和信号处理的。它实现了机器人最低层的动作，并处理传感器得到的信息。第二层是行为层，该层由一整

套动作构成，这些动作一方面根据外部环境的特征来控制机器人，另一方面吸取对外界的测量用来对机器人进行内部的表示：拓补地图。表示任务的序列层用来完成动作序列

，其核心是由机器人自身状态的外界感知和上层规划的参数设定决定的状态来自动激发单位动作，后者通过操作者的指示和拓补地图以及外界反馈信息给动作序列合理的参数。

该体系是由动态的实验环境组成的，包括一个在室内环境移的 Nomad200 移动机器人（Nomadics,1992），一组不同的传感器，精致的视觉传设备，以及一套基于传感器的动作序列和多功能控制单元。机器人在真实的环境中成功地完成一些任务证明了这种体系的有效性（Tie‘eche et al., 1995）。

3. 行为

行为层包括了不同的动作行为。其中的一些是直接和“自主规划点”相关的，其他的则是和各点间的移动相关。

有两种行为是和“自主规划点”相关的：一种是自主规划行为，这种行为帮助机器人直接到达目标点，而定位行为则帮助机器人确认自己的位置。在自主规划行为中，遇角点归巢行为（Facchinetti and HuK gli, 1994）控制机器人固定的姿态，这些姿态由环境的特殊构成定义：凸角和缓角，在本文感兴趣的特定姿态：机器人方向对着角，位于对角线与角固定距离处.这个行为可以接收 Sensus500 结构光视觉系统提供的（角度）范围描述，并驱动机器人减小参照角度和和观察角度的误差。另一种基于视觉的“自主规划”行为是遇目标归巢行为，该行为和视觉系统中的一对路标相关。

区分不同归巢地点的行为被称为定位行为（Tie‘che et al.,1996）。它使用灰色视频相机，朝向屋顶，当机器人达到自主规划地点时，机器人拍下当前周围的快照，与事先存在数据库中的一系列照片比较，它返回对未知地点的辨认。归巢行为和定位行为的结合使得机器人可以精确地无歧义地辨认明显的地点。

和机器人在两个规划点间的置换相关的行为被称为“出发”行为。其中的一种行为控制机器人循着由 Sensus500 结构光视觉系统探测到的墙。另一种则是由路标被激发，控制机器人向路标进发,并在指定距离使得机器人停止，定位。

４．拓补地图

拓补地图描述了环境中标志地点间的相邻关系,规范来说，拓补地图由 G=（V,E）构成，V={v1 ，…,vN}是点的集合， E={eij}={（vi,vj）} 是边的集合。它可以从两个方面来论述。从拓补的观点来看，它是关注于对环境以符号的方式表示。从机器人资源的观点来看，这张地图是基于机器人通过传感器和自己的行为和外界的交互。在此地图中，每个结点对应于一个自主规划点，而边则对应于不同点间的置换。遇角点归巢和定位的行为对应于点，而“出发”行为则对应于沿边移动。

选择角作为环境的特征地点实现自主规划是建立在角点容易探测的前提上的，这一点在很多人造环境（像桌子、墙、门等）中得到了证实。这给了地图很高的精度和稳定度。

5．额外的几何信息

本节论述了在拓补地图中添加额外的几何信息。这种思想在机器人在规划点间和沿着地图中的边移动时记录了里程路径。据路的结果是一系列里程路径，这些信息可以用来构建全局（含几何信息）地图。

更精确地说，拓补地图是通过在一个一个角之间手工地或者依靠一组充足的动作，移动机器人构建的，这个构建过程生成了一个序列的已达点，存储在一个列表中 $\Sigma = \{v_i\}_{1 \leqslant i \leqslant M+1}$。机器人的姿态是三维的，包括位置和角度的信息，可以定义为 $p = (x, y, \varphi)^t$。

里程路径提供了机器人在两个规划点时所摆姿态的几何关系。一个在两个点 $A\,(x_A, y_A, \varphi_A)^t$ 和 $B\,(x_B, y_B, \varphi_B)^t$ 间的路径 w_{AB} 可以用一个三维向量表示为 $w_{AB} = (d_{AB}, \alpha_{AB}, \beta_{AB})^t$（图. 1）.

假定机器人是处于姿态 A，α_{AB} 是其和目标点间连线的夹角，d_{AB} 是；两点间的距离，β_{AB} 是 B 位置机器人姿态和连线的夹角。定义复合表达式表示 B 点姿态 p_B，A 点姿态 p_A，以及两点间的路径 w_{AB}，这个表达式定义为：

$$p_B = p_A \oplus r_{AB}.$$

$$\begin{pmatrix} x_B \\ y_B \\ \varphi_B \end{pmatrix} = \begin{pmatrix} x_A \\ y_A \\ \varphi_A \end{pmatrix} \oplus \begin{pmatrix} d_{AB} \\ \alpha_{AB} \\ \beta_{AB} \end{pmatrix}$$
$$= \begin{pmatrix} x_A + d_{AB}\cos(\varphi_A + \alpha_{AB}) \\ y_A + d_{AB}\sin(\varphi_A + \alpha_{AB}) \\ \varphi_A + \alpha_{AB} + \beta_{AB} \end{pmatrix}. \quad (1)$$

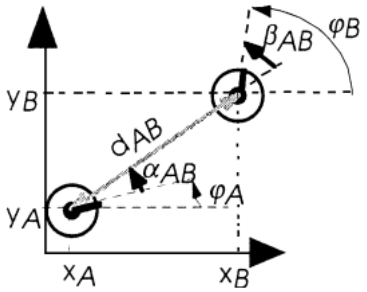


Fig. 1. The path between two robot poses A and B is defined by the three-dimensional vector $(d_{AB}, \alpha_{AB}, \beta_{AB})$.

这个表达式是右结合的$p = ((p_0 \oplus w_1) \oplus w_2) \ldots) \oplus w_k$，一系列复合表达式被定义为：

$$p = p_0 \bigoplus_{i=1}^{k} w_i. \quad (2)$$

同样的，逆运算符合表达式$p_B \star P_A = w_{AB}$表示了和机器人姿态有关的两点间的路径：

$$\begin{pmatrix} d_{AB} \\ \alpha_{AB} \\ \beta_{AB} \end{pmatrix} = \begin{pmatrix} x_B \\ y_B \\ \varphi_B \end{pmatrix} \star \begin{pmatrix} x_A \\ y_A \\ \varphi_A \end{pmatrix}$$
$$= \begin{pmatrix} \sqrt{(x_B + x_A)^2 + (y_B - y_A)^2} \\ \arctan((y_B + d_A)/(x_B + x_A) - \varphi_A \\ \varphi_B - \arctan((y_B - y_A)/(x_B - x_A)) \end{pmatrix}. \quad (3)$$

相反的路径同样可以定义为$\star w_{AB} = w_{BA}$，这意味着路径已知的情况下，它的泥路经也可以被计算出来。这些运算符和另两位学者 Lu 和 Milios （1997）很相近，但由于路径的定义不同，所以实际上是不同的。

6．连续路几何地图的构建

在给出拓补地图，及其相关信息用 M 测算的几何路径w_{ij}^{m}，以及已探测的结点序列Σ）。几何地图构建的问题是如何在一个单一坐标系统中确定 N-1 个机器人姿态$\bar{p}_i = (\bar{x}_i, \bar{y}_i, \bar{\varphi}_i)^t$。首先考察其中一个姿态，并以设定原点。原点时可以任意选择的，可以选择第一个探测的点作为原点$\rho_{\Sigma(1)} = (0, 0, 0)^t$。

下面提出了解决这个问题的五种方法。

6.1 M 1：沿探测序列的路径集成

这种方法一一考察探查表中的结点，然后通过集成连续的路径找到他们的姿态。正式来说，结点集合$\Sigma(l)$中的姿态$\hat{p}_{\Sigma(l)}$可以通过把一系列基本路径联合起来形成$\Sigma(l)$：

$$\hat{p}_{\Sigma(l)} = p_{\Sigma(1)} \bigoplus_{k=2}^{l} w_{\Sigma(k-1)\Sigma(k)}^{m}. \qquad (4)$$

一旦在结点集合中出现回环，就会有结点不止一次被访问，因此它们的姿态就会被计算数次。为了对每个结点仅分配一个姿态的计算结果，算法仅保留了第一次计算的结果，丢弃了另外剩余的计算结果。

算法 M1 的复杂度是 O（M），其中 M 代表边数。

6.2 M2:：沿探测序列的路径集成（无回环）

这种方法也一一考察探查表中的结点，然后通过集成连续的路径找到他们的姿态。但是当回环出现后，计算被终止并从回环的第一个结点开始重新计算。在这种情形下，一个姿态被赋予每个结点。

算法 M2 的复杂度是 O（M）。

6.3 M3：:沿最小距离生成树的路径集成

这种方法通过对原始结点进行一系列路经计算得到当前结点的姿态。在图中，在原始结点和当前结点有许多路径，算法选择两点间距离最短的路径,最短路经定义为一个序列中所有单元路径距离和。这种算法为根结点计算出了最小生成树。

算法 M3 的复杂度是 O（MN）。

6.4 M4：沿最小方向生成树的路径集成

这种方法通过对原始结点进行一系列路经计算得到当前结点的姿态。在图中，在原始结点和当前结点有许多路径，算法选择两点间角度花销最少的路径,最短路经定义为一个序列中所有单元角度和 $|\alpha| + |\beta|$ 。这种算法为根结点计算出了最小生成树。

算法 M4 的复杂度是 O（MN）。

6.5 M5：最小二乘法

最小二乘法使实际测量路径 w_{ij}^{m} 和算法估计路径 $\hat{w}_{ij}$ 的误差达到最小。

$$f(\hat{w}) = (w^{m} - \hat{w})^{t} P(w^{m} - \hat{w})$$

其中 P 是矩阵的权重。

可使用非线性函数来表达对姿态关系的估计： $\hat{w}_{ij} = \hat{p}_j @ \hat{p}_i$，因此，函数的最小值取决于对机器人姿态的估计。当梯度为零时，函数达到最大或者最小值。

$$\nabla f(\hat{p}) = \frac{\partial f(\hat{p})}{\partial \hat{p}} = 0. \qquad (5)$$

这种方法提供了带有 3N 个未知数的非线性方程组，可以用 Newton-Raphson 迭代方法来解此方程组。

算法 M2 的复杂度是 O（N^3）。

7．实验结果

本节提供了由一个 Nomad200 移动机器人探测环境来重建的几何地图。并比较了五种不同方法得到的实验结果。

7．1 精确地图

真实的环境由散布在一个 10*12 米的表面的 28 个归巢地点（11 个缓角，17 个凸角）组成，见图 2a。为了比较对机器人姿态 $\hat{p}_i = (\hat{x}_i, \hat{y}_i, \hat{\varphi}_i)^t$ 的重建地图，测量了一个对机器人姿态 $p_i^e = (x_i^e, y_i^e, \varphi_i^e)^t$ 的精确地图。地图的构建分两步。首先，精确地测量角并将其映射到地图上。然后，通过平衡几种测量方法，建立相对于角的机器人姿态。最后，这些结果被添加到精确地图的角上，以此得到对机器人姿态的精确地图。

7．2 精确地图和估计地图的比较

经过严格的排列变换，我们比较了精确和估计的地图。我们用地图中对应位置距离的均方差 Δd 和角度的差别 $\Delta\varphi$ 来衡量两者的差别。

$$\Delta d = \sqrt{\frac{1}{N}\sum_{N}(x_i^e - \hat{x}_i)^2 + (y_i^e - \hat{y}_i)^2} \quad (6)$$

$$\Delta \varphi = \sqrt{\frac{1}{N}\sum_{N}(\varphi_i^e - \hat{\varphi}_i)^2}\,. \quad (7)$$

7.3 已探测路径的集合

我们用机器人在环境中行进时纪录的里程来测量路径。我们测量了在 28 个自主规划点中的 72 条路经。图 2b 展示了初始姿态沿着探测序列经过 72 条路径的复合。注意如果一个结点被访问超过一次，它将被数个点的姿态表示。

7.4 估计地图

图 3 比较了通过五种方法重建的几何地图和精确地图。地图由对应于路径的几何边构建，注意在方法 M2、M3 和 M4 种，很多边被忽略了。

很明显，方法 M1 重建的地图很糟糕；M2、M3 和 M4 的结果尚可接受，而 M5 获得了很好的结果。可视的结果在表 1 的结果中被印证了。表一的结果是用对距离均方差和角度的最小二乘法得到的。

除了 M5 的各种方法，误差会沿着路径的延伸被积累。离原点越远，对机器人的姿态的估计会越不精确。而且，一些变化会对地图产生重大的变化。而 M5 非常精确并且稳定，所以即使该方法较费时，还是更可取的。

表 2 给出了各种方法的处理时间的比较结果。首先，它概括了各种方法的计算复杂度，N 代表结点个数，M 代表边数。表格记录了构建地图（图 2,N=28,M=72）的有效计算次数

表格表现了各种方法相对的数量关系,并清楚地表现了各种方法的差别。注意我们的结果是在个人电脑上的 Mathimatica 软件计算出来的，并没有用优化的方法，所以结果还可以得到优化。但是，在地图信息庞大，存在大量的结点。如果使用 M5 的方法，如果对该方法复杂度 O（N^3）的花费无法接受，我们需要特殊的测算方法，比如地图分解。

8．结论

本文论述了如何通过连续相关的几何地图去扩展包含规划点的拓补地图。并提出了利用地图拓补信息和在规划点间的路径里程信息在单幅参照地图中定位的五种方法。四种根据不同策略结合路径的方法，以及一种全局求最小极值的方法。

这些几何地图构建方法在一个 Nomad200 机器人在含有 28 个规划点的环境中测试实现了。五种方法数学地计算了地图重建的误差，并图形化地比较了构建的地图和原始精确地图间的差别。甚至在大的环境中，最小二乘法也显示了最好的精度并给了优良的结果。

Table 1
Difference between exact and estimated map

	M1	M2	M3	M4	M5
Δd (cm)	153.0	47.9	40.4	28.3	10.7
$\Delta\varphi$ (°)	28.4	11.1	5.3	5.7	2.0

Table 2
Time complexity and computing time

	M1	M2	M3	M4	M5
Complexity	$O(M)$	$O(M)$	$O(M.N)$	$O(M.N)$	$O(N^3)$
Time (s)	0.1	0.1	2	2	65

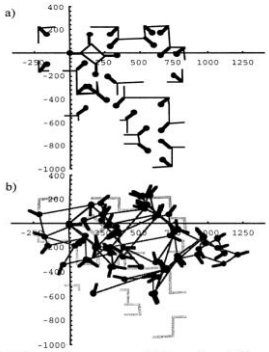


Fig. 2. (a) Exact robot poses map; (b) Integration of all measured paths.

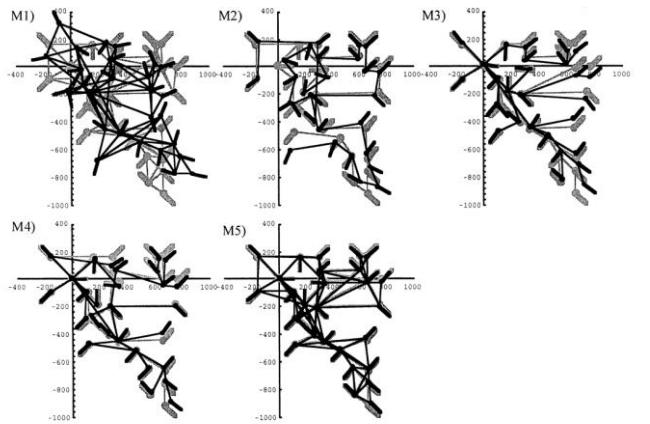
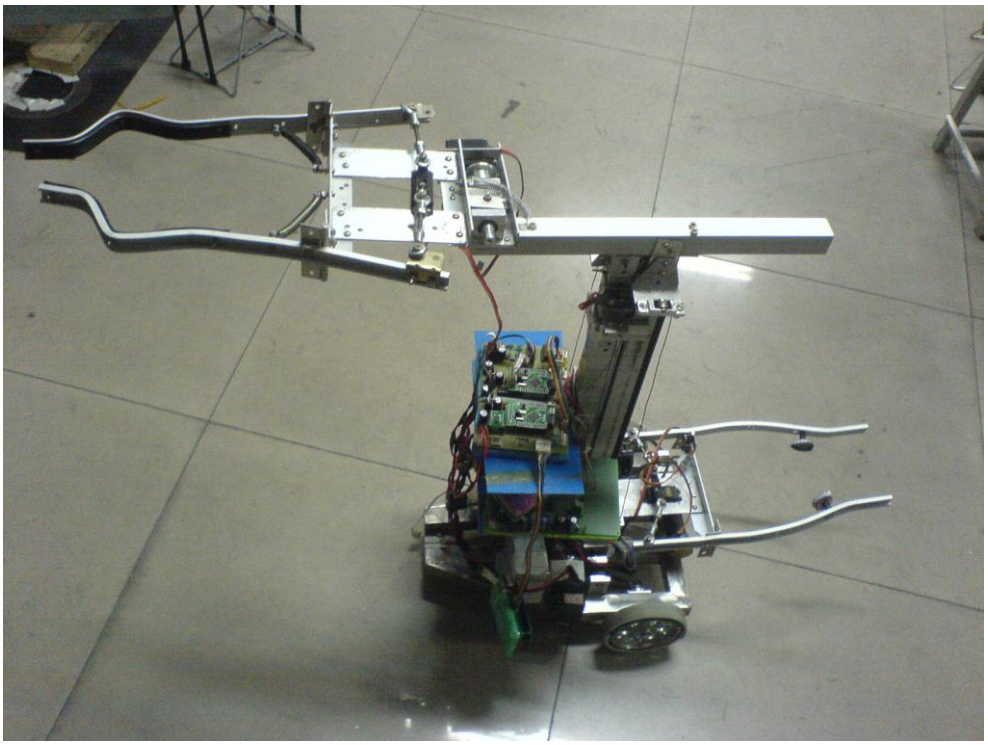


Fig. 3. Comparison of reconstructed (black) and exact (gray) maps for each method. The edges shown are the ones used for reconstruction.

附 录 C　自动机器人照片



双升降翻转型机器人总装图

附 录 D　自主移动机器人部分程序代码

（1）任务调度中的核心函数 ErrorProcess（AI_TASK_POINTER）源代码

```
uint8 ErrorProcess（AI_TASK_POINTER pCurrentTask）
{
    fp32 yDst;
     set_speed（255,0）;
//失败后的处理过程
        if（pCurrentTask->cError == MOVE_FAIL||pCurrentTask->cError == ACHIEVE_FAIL）{
              //使指针指向下一个任务
                UpdateMap（robotState.CurrentPoint,robotState.CurrentAngle）;
                set_speed（255,0）;
              Backward（1000,500,robotState.CurrentAngle,pCurrentTask）;
              if（pCurrentTask->MoveRetryTimes > 0）{                    pCurrentTask-
>MoveRetryTimes --;
                 if（（robotState.CurrentPoint.y* pCurrentTask->EndPoint.y<0）&&（ABS
（robotState.CurrentPoint.y）<2000）&&（robotState.CurrentPoint.x>500））{
                    if（robotState.CurrentPoint.y<0）{
                        TurnAngle（TURN_ANGLE（-900,robotState.CurrentAngle）,100）;
    P2PForward（800,robotState.CurrentPoint,PPoint（robotState.CurrentPoint.x,-3250））;
                        TurnAngle（TURN_ANGLE（900,robotState.CurrentAngle）,100）;
                    }
                    else{
                        TurnAngle（TURN_ANGLE（900,robotState.CurrentAngle）,100）;
    P2PForward（800,robotState.CurrentPoint,PPoint（robotState.CurrentPoint.x,3250））;
                        TurnAngle（TURN_ANGLE（-900,robotState.CurrentAngle）,100）;
                    }
                 }
                 ////////////////////////////////////
                if（0 < UseAutoRoute（pCurrentTask,pCurrentTask->EndPoint））{//如果可以规
划出到当前任务目标点的路径
                    Backward（500,500,robotState.CurrentAngle,pCurrentTask）;
                    RUN_IN_AI
                    return 2;
                }
                else{//如果规划不出到当前任务目标点的路径
                    pCurrentTask = Go2NextTask（pCurrentTask,AI_MODE）;
Unavailable:        if（NULL != pCurrentTask）{//如果还有后续任务
                        PathInit（pCurrentTask）;//有必要初始化一次，使EndPoint初始化
                        ////////////////////////////////////
                         if（（robotState.CurrentPoint.y* pCurrentTask->EndPoint.y<0）&&（ABS
（robotState.CurrentPoint.y）<2750）&&（robotState.CurrentPoint.x>500））{
                            if（robotState.CurrentPoint.y<0）{
                                TurnAngle（TURN_ANGLE（-
900,robotState.CurrentAngle）,100）;
    P2PForward（800,robotState.CurrentPoint,PPoint（robotState.CurrentPoint.x,-3250））;
                                TurnAngle（TURN_ANGLE
（900,robotState.CurrentAngle）,100）;
```

```
                                }
                                else{
                                    TurnAngle（TURN_ANGLE
（900,robotState.CurrentAngle）,100）;
    P2PForward（800,robotState.CurrentPoint,PPoint（robotState.CurrentPoint.x,3250））;
                                    TurnAngle（TURN_ANGLE（-
900,robotState.CurrentAngle）,100）;
                                }
                            }
                            ////////////////////////////////////
                            if（0 < UseAutoRoute（pCurrentTask,pCurrentTask->EndPoint））{
                        Backward（500,500,robotState.CurrentAngle,pCurrentTask）;
                                RUN_IN_AI
                                return 2;//跳出调用层的for循环
                            }
                            else{/                                    set_speed（255,0）;
                                goto Unavailable;//return 2;//路径规划失败，目标点不可达，暂时按
照原来固定路径再走一遍
                            }
                        }
                        else{//如果没有后续任务
                            pCurrentTask =  （AI_TASK_POINTER）malloc（sizeof（AI_TASK））;
                            AITaskInit（pCurrentTask,AI_MODE）;
                            PathInit（pCurrentTask）;
                            ////////////////////////////////////
                            if（（robotState.CurrentPoint.y* pCurrentTask->EndPoint.y<0）&&（ABS
（robotState.CurrentPoint.y）<2000）&&（robotState.CurrentPoint.x>500））{
                                if（robotState.CurrentPoint.y<0）{
                                    TurnAngle（TURN_ANGLE（-
900,robotState.CurrentAngle）,100）;
                                    P2PForward（800,robotState.CurrentPoint,PPoint
（robotState.CurrentPoint.x,-3250））;
                                    TurnAngle（TURN_ANGLE
（900,robotState.CurrentAngle）,100）;
                                }
                                else{
                                    TurnAngle（TURN_ANGLE
（900,robotState.CurrentAngle）,100）;
                                    P2PForward（800,robotState.CurrentPoint,PPoint
（robotState.CurrentPoint.x,3250））;
                                    TurnAngle（TURN_ANGLE（-
900,robotState.CurrentAngle）,100）;
                                }
                            }
                            ////////////////////////////////////
                            if（0 < UseAutoRoute（pCurrentTask,ReferancePoint[RIVAL_ZONE]））
{//如果可以规划出到达对方出发区的路径
                                Backward（500,500,robotState.CurrentAngle,pCurrentTask）;
                                RUN_IN_AI
                                return 2;
```

```
                    }
                    else{//如果规划不出到达对方出发区的路径
                        set_speed（255,0）;
                        while（1）;//return 2;//路径规划失败，目标点不可达，暂时按照原来固定路径再走一遍
                    }
                }
            }
        }
        else{//如果是第二次碰撞，放弃当前任务，执行下一个任务
//NextTask:
            AITaskInit（pCurrentTask,AI_MODE）;
            pCurrentTask = Go2NextTask（pCurrentTask,AI_MODE）;
            if（pCurrentTask != NULL）{//如果还有后续任务
                PathInit（pCurrentTask）;//有必要初始化一次，使EndPoint和RetryTimes初始化
                ////////////////////////////////////
                if（（robotState.CurrentPoint.y* pCurrentTask->EndPoint.y<0）&&（ABS（robotState.CurrentPoint.y）<2000）&&（robotState.CurrentPoint.x>500））{
                    if（robotState.CurrentPoint.y<0）{
                        TurnAngle（TURN_ANGLE（-900,robotState.CurrentAngle）,100）;
                        P2PForward（800,robotState.CurrentPoint,PPoint（robotState.CurrentPoint.x,-3250））;
                        TurnAngle（TURN_ANGLE（900,robotState.CurrentAngle）,100）;
                    }
                    else{
                        TurnAngle（TURN_ANGLE（900,robotState.CurrentAngle）,100）;
                        P2PForward（800,robotState.CurrentPoint,PPoint（robotState.CurrentPoint.x,3250））;
                        TurnAngle（TURN_ANGLE（-900,robotState.CurrentAngle）,100）;
                    }
                }
                ////////////////////////////////////
                if（0 < UseAutoRoute（pCurrentTask,pCurrentTask->EndPoint））{//如果可以规划出到达下个任务终点的路径
                    Backward（500,500,robotState.CurrentAngle,pCurrentTask）;
                    RUN_IN_AI
                    return 2;
                }
                else{//如果规划不出到达下个任务终点的路径
                    goto Unavailable;//路径规划失败，目标点不可达，暂时按照原来固定路径再走一遍
                }
            }
            else{//如果没有后续任务
                pCurrentTask =  （AI_TASK_POINTER）malloc（sizeof（AI_TASK））;
                AITaskInit（pCurrentTask,AI_MODE）;
                PathInit（pCurrentTask）;
                ////////////////////////////////////
                if（（robotState.CurrentPoint.y* pCurrentTask->EndPoint.y<0）&&（ABS
```

```
（robotState.CurrentPoint.y）<2000）&&（robotState.CurrentPoint.x>500））{
                                if（robotState.CurrentPoint.y<0）{
                                    TurnAngle（TURN_ANGLE（-900,robotState.CurrentAngle）,100）;
                                    P2PForward（800,robotState.CurrentPoint,PPoint
（robotState.CurrentPoint.x,-3250））;
                                    TurnAngle（TURN_ANGLE（900,robotState.CurrentAngle）,100）;
                                }
                                else{
                                    TurnAngle（TURN_ANGLE（900,robotState.CurrentAngle）,100）;
                                    P2PForward（800,robotState.CurrentPoint,PPoint
（robotState.CurrentPoint.x,3250））;
                                    TurnAngle（TURN_ANGLE（-900,robotState.CurrentAngle）,100）;
                                }
                             }
                            ////////////////////////////////////
                            if（0 < UseAutoRoute（pCurrentTask,ReferancePoint[RIVAL_ZONE]））{//如
果可以规划出到达对方出发区的路径
                                 Backward（500,500,robotState.CurrentAngle,pCurrentTask）;
                                RUN_IN_AI
                                return 2;
                            }
                            else{//如果规划不出到达对方出发区的路径
                                goto Unavailable;//return 2;//路径规划失败，目标点不可达，暂时按照原
来固定路径再走一遍
                            }
                        }
                    }
    }
    else if（pCurrentTask->cError == WORK_FAIL）{
    //再执行一遍机构动作DoWork()函数
    return 5;
    }
     return 1;
}
```

在 学 取 得 成 果

1、 荣获 2007 年飞思卡尔杯全国大学生智能汽车竞赛一等奖（2007）

2、 荣获 2007 年飞思卡尔杯全国大学生智能汽车竞赛华北区一等奖（2007）

3、 荣获全国大学生电子设计大赛北京市二等奖（2007）

4、 荣获全国大学生英语竞赛二等奖（2007）

5、 荣获北京科技大学非数学专业高等数学竞赛二等奖（2007）

6、 北京科技大学“摇篮杯”科技作品竞赛三等奖（2007）

7、 荣获北京科技大学人民二等奖学金（2007）

8、 荣获北京科技大学非物理专业物理竞赛二等奖（2006）

9、 荣获北京市青少年多米诺骨牌比赛优胜奖（2006）

10、荣获全国大学生数学建模竞赛成功参赛奖（2006）

11、荣获北京科技大学三好学生荣誉称号（2006）

12、荣获北京科技大学非数学专业高等数学竞赛一等奖（2006）

13、荣获北京科技大学信息工程学院第四届机器人运动会第五名（2006）

14、荣获北京科技大学人民二等奖学金（2006）

15、荣获北京科技大学非数学专业高等数学竞赛一等奖（2005）

16、荣获北京科技大学国家二等奖学金（2005）

17、荣获北京科技大学优秀三好学生荣誉称号（2005）

18、荣获北京市大学生数学竞赛非数学类 A 组三等奖（2005）

19、荣获北京科技大学 2004 年度学生社会实践优秀调研报告（2005）

20、荣获北京科技大学人民三等奖学金（2005）

21、荣获北京科技大学英语竞赛三等奖（2004）

致　　谢

本文的撰写是基于第七届 CCTV 全国大学生机器人大赛进行的。

在此，我要深深的感谢在此次比赛中带领全体参赛队员进行策略制定、具体制作的郁安民老师、刘颖老师；感谢王玲老师对毕业设计整个过程的关心、严格要求和指导指正；老师们严谨的治学作风、渊博的专业知识让我受益匪浅，使这份毕业设计从实践到理论得到了进一步的提高，在此对他们表示由衷的感谢和崇高的敬意。

同时，感谢领队刘万鹏、于洪金对机器人大赛的支持，感谢学校为我们提供了良好的试验环境与场地。

感谢奚为宁、周珂、王盛、冯涛、刘涛、陈曦、李博等顾问在技术指导、策略经验等问题上给予的大力支持。

在整个设计制作的过程中，我还要感谢蔡颖鹏、安斌、罗文灿、刘凯、李哲媛的关心和技术支持，让我在最困难的时候能够解开难题，完成最终的设计；感谢刘谦和安斌同学在电机选配上不厌其烦的挑选和测试，使机器人能够拥有合适而匹配的驱动器；正是基于整个团队的协调与合作，此次机器人制作才能得以实现和完成，在此向工作组的所有同学表示深深的谢意。

另外，感谢我的父母对我参加机器人大赛的理解与支持，感谢我的本系各位老师给予的鼓励和支持，正是这些让我能够全力准备此次机器人大赛，完成毕业设计的构思和论文的撰写。

最后向全体参加论文评审和答辩的各位老师表示感谢！