第十四届“恩智浦”杯全国大学生
智能汽车竞赛
室外光电创意组

技 术 报 告

参赛学校：湖北工业大学

队伍名称：湖北工业大学创意组

参赛队员：周恒

徐磊鑫

刘卓勋

黄金基

带队教师：詹云峰

韦琳

关于技术报告和研究论文使用授权的说明

本人完全了解第十四届“恩智浦”杯全国大学生智能汽车竞赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和恩智浦公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：_____

带队教师签名：_____

日　　期：_____

目录

摘 要

本文主要介绍了湖北工业大学第十四届智能车大赛创意组的主要思路与技术方案。根据规则要求，本届室外创意组需要通过组委会提供的车模根据构建好的地图自主导航，避开锥桶障碍物，从赛道起点跑到终点。本赛事所提供的车模为 ROS 竞速车，它由软件和硬件两个大部分，硬件以车体结构为整体框架，以主控电脑作为控制载体，车上搭载所有需要的传感器硬件以及驱动装置；软件以 ROS 机器人操作系统为基础，配合 SLAM 构建地图和路径规划和自主导航。因此在当前平台下要实现任务要求，需要使用 SLAM_Gmapping 功能包获取激光传感器数据构建二维栅格地图，然后通过 ROS 的 move_base 功能包将全局导航和局部导航链接在一起完成其导航任务，全局导航用于建立到地图上最终目标或一个远距离目标的路径，局部导航用于建立到近距离目标和为了临时躲避障碍物的路径，再通过 amcl 算法进行自身定位，完成机器人在未知环境下的自主导航任务，再通过串口，将导航所计算的 ROS 小车的线速度和角速度传输到单片机中，进行机器人导航。

关键字：SLAM；自主导航；定位

引 言

全国大学生“恩智浦”杯智能汽车竞赛是一项旨在加强学生实践、创新能力和培养团队精神的创意性科技竞赛。为了实现竞赛的“立足培养、重在参与、鼓励探索、追求卓越”的指导思想，第十四届竞赛创意组在竞速组基础上，适当增加挑战性，形成创意比赛的内容。

无人驾驶技术作为人工智能领域的“前沿”阵地，涉及智能控制、信息通讯、传感技术信息技术、电子工程、控制理论、传感技术等多领域技术融合，对“跨学院、跨专业、跨学科”新时代下的新型复合人才培养提出了更好的要求。

本赛项的设立能够场景化的复现基于无人驾驶的智能车在实际领域中的应用，尤其是在无人的环境中，实现定位导航、计算机视觉、雷达、人工智能、自动控制和电机控制等多种技术融合的场景。通过室外无人驾驶创意赛，期望达到以赛促教，进一步深化产学融合，拓宽高校人工智能及机器人相关专业的教学内容，提升高校人工智能及机器人科技创新能力和人才培养能力。 全国大学生智能汽车竞赛，是高校中影响力最大、参与度最广的大学生竞赛平台之一，设立室外无人驾驶创意赛，可以更好地培养大学生掌握机械电子、运动控制、传感器应用、机器学习、图像识别、SLAM 地图构建、自主导航等人工智能领域先进技术，从而让大学生提前了解并掌握产业界最常用、最实用的先进技术。

本设计报告基于第十四届“恩智浦”杯全国大学生智能汽车竞赛创意组比赛规则，技术报告主要包括无人车设计制作的主要思路以及实现的技术方案概要说明，以及电路设计说明。

第一章 无人车总体设计思路

1.1 系统概述

第十四届创意组比赛规则要求设计一辆车模，在已知地图环境下进行SLAM，并能躲避未知障碍物。

根据比赛要求，将无人车件主系统分为硬件系统和软件系统。硬件以主控电脑为核心，外部搭载激光雷达等传感器，用来采集整车周围所有的物体相关的距离信息；同时也搭载了姿态传感器，用来获得车辆整体的姿态信息和加速度信息。使用直流无刷电机作为车的驱动装置，舵机用来控制车辆整体的转向装置。通过KV58单片机作为下位机的控制器，KV58通过CAN总线接收电脑传输的控制信号，从而进一步再对直流无刷电机和舵机进行控制。由于车辆并不配有屏幕，所以车上搭载了路由器，方便操作人员进行远程控制的工作。软件主要以ROS机器人操作系统为框架主体，在ROS机器人操作系统上，添加激光雷达，IMU和摄像头的驱动程序，这样我们便可以感知周围环境信息和车的姿态信息；通过无刷直流电机和舵机控制无人车的移动，从而感知无人车周围的环境信息。然后对这些采集到的信息进行处理，通过SLAM_Gmapping算法对整个环境的地图进行构建，这样我们就可以获得整体环境的地图，再通过路径规划和自主导航，就可以控制车到达想要到达的相应位置上。在软件上，我们可以通过SSH或者VNC远程访问，用来控制整个系统。

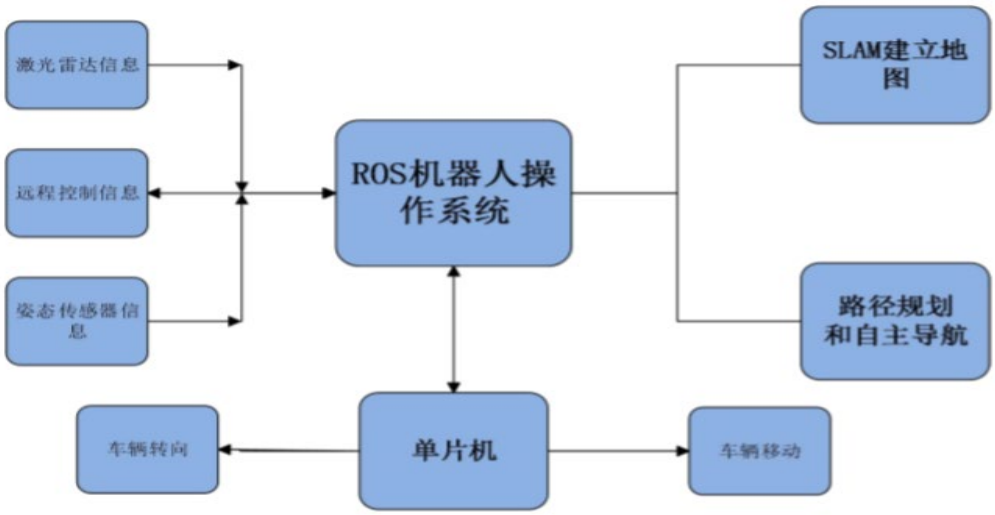

图 1.1 ROS机器人操作系统

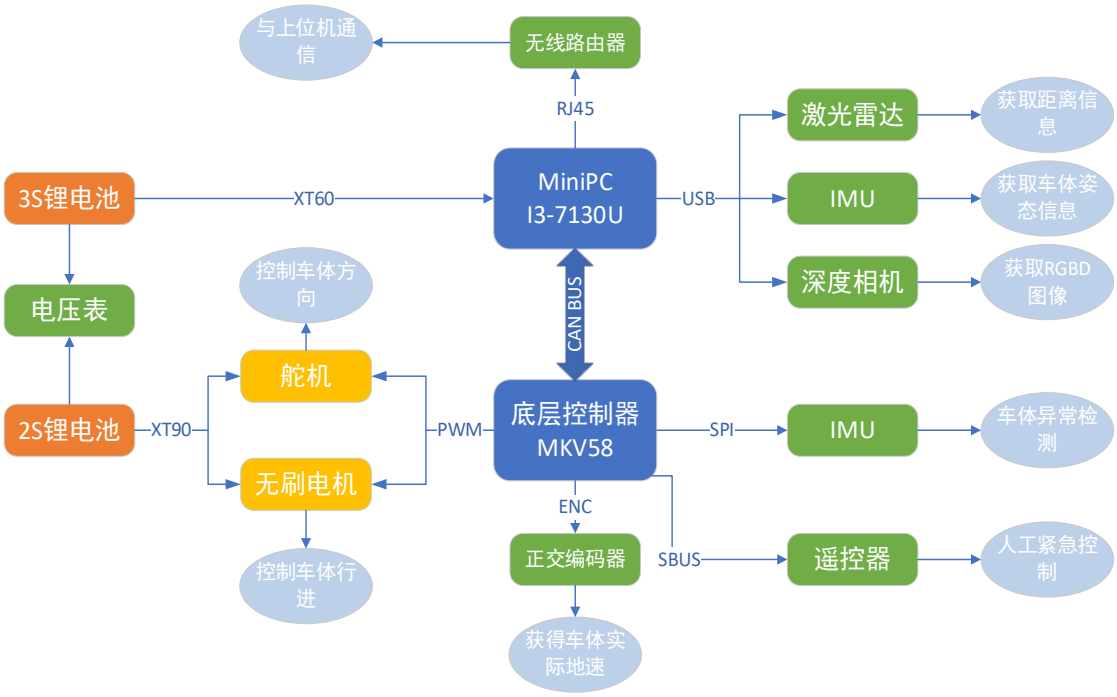

图 1.2：无人车系统框架

1.2 无人车技术方案概要

由规则可知，我们的比赛是在已知地图下进行自主导航，并尽可能快的到达目标位置。我们的计算平台基于 Intel 的 i3-7130U 处理器，内存只有 4GB，因此我们选用基于激光 SLAM 算法作为我们无人车的主要算法，它相较于视觉 SLAM 具有计算量低，可靠性高等优点。

对于激光 SLAM 来说，可获取的传感器信息一般有里程计，激光雷达以及 IMU 数据，然而激光雷达数据会由于传感器的移动而导致采集回来的数据产生畸变，需要通过估计或者里程计辅助等方法对畸变进行矫正，在我们的无人小车上采用的是 VICP 进行运动畸变去除。

无人车在 SLAM 过程中，之前产生的误差将不可避免的累积到下一时刻，使得整个 SLAM 出现累积误差，无法长期估计，因此采用回环检测对无人车的 SLAM 系统意义重大，他关系到我们估计的轨迹和地图在长时间下的正确性，当然也提供了当前数据与所有历史数据的关联，我们还可以利用回环检测进行重定位。因此，回环检测对整个 SLAM 系统精度与稳健性的提升是非常明显的。

回环的一种简单的方法就是对任意两帧数据都做一遍特征匹配，但是缺点是我们不能盲目的假设任意两个图像都可能存在回环，使得到检测的数量特别大，这种情况在大多数系统上是不实用的。而另一种是随机抽取历史数据进行回环检

测，这种方法能够减少运算量，但是这种盲目试探的方法在帧数 N 增长时，抽到回环的几率又会大幅下降，使得检测效率不高。

为了解决盲目检测，大体分为两种思路：一种是基于里程计的几何关系，另一种是基于外观。几何关系是指当我们发现机器人运动到了之前的某个位置附近时，检测他们有没有回环关系，但是由于当前由于累积误差的存在，我们往往没法正确的发现运动到了某个位置，所以回环也就无从谈起。而基于外观的方法，它和前端、后端的估计都没有关系，仅根据两幅图像的相似度，确定回环检测的关系。这种方法摆脱了累积误差，使得回环检测模块成为了一个相对独立的模块。这种基于外观的回环检测方式能够有效地在不同场景下工作，逐渐成为了视觉 SLAM 的主流做法，并被应用于实际的系统中去。

我们采用 ROS 中的 move_base 功能包和 amcl 功能包进行无人车的导航和定位。

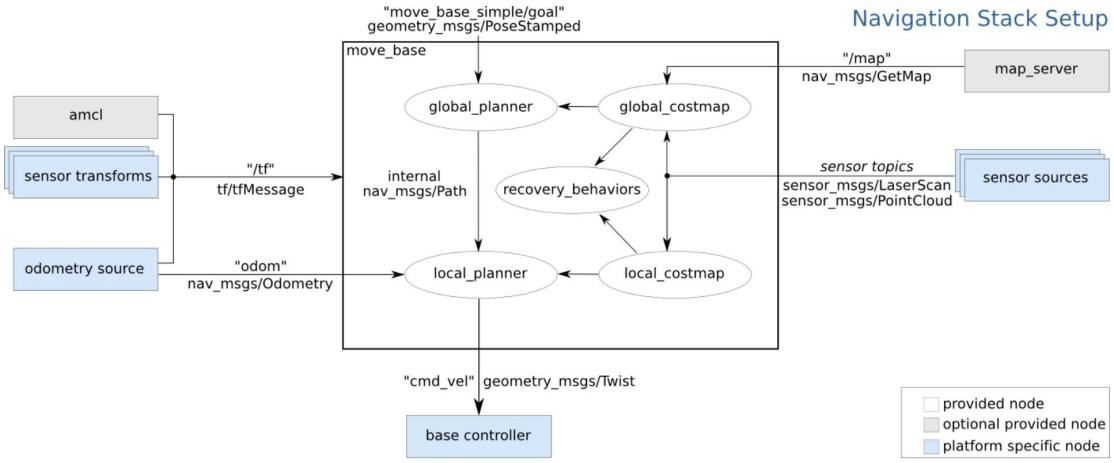

图 1.3：move_base 框架

其中椭圆框内是导航必须用到的基本组件，蓝色和灰色矩形框是供应用开发者需要的组件。

在进行导航过程中，我们需要对传感器进行坐标变换，因此我们采用 tf 变换树定义不同坐标之间的偏移，通常的，我们将无人车的基座的坐标系定义为 base_link，位于激光雷达的坐标系定义为 base_laser。通过 ROS 的 tf 软件包，我们便可以将激光雷达等数据转换到 base_link 坐标系下，机器人就可以利用此信息实现避障了。

在全局路径规划中我们采用 A*方法，A*算法是一种高效的路径搜索算法。采用启发函数来估计地图上机器人当前的位置到目标位置之间的距离，并以此选择最优的方向进行搜索，如果失败会选择其他路径继续搜索直到得到最优路径。

在局部实时路径规划中使用 DWA（Dynamic Window Approaches，规划推理和动态窗口）算法，计算机器人每个周期内应该行驶的速度和角度。DWA 算法中先离散采样机器人控制空间，再对于每个采样速度，从机器人当前的状态，进行模拟预测。如果采样速度应用于一段时间内，将会出现什么情况，然后用合并一些特征的度量标准来，从模拟预测中，评价每个轨迹结果，如障碍物接近目标，接近全局路径和速度。舍弃不合适的路径（有障碍物碰撞的） 最后挑选得分最高的轨迹，并且发布相关的速度给移动平台。

为了更好的学习 SLAM 技术，我们也对视觉 SLAM 进行了了解与尝试。我们在车上放置了 Intel RealSense D435i 深度相机，利用 OpenCV 对相机采集的 RGB-D 图像进行处理，先利用 RGB 图像对障碍物进行探测，后映射至深度图像探测获得障碍距离。

第二章 车体机械部分

2.1 车体机械结构调整

本次比赛所使用的车模为 1:8 越野大脚车模。车模到手，第一件事就是调节车辆机械结构。对于本比赛使用的大脚车模，网上很多说法说此类型车本身虚位大，精度差，用不着调节，此种言论有失偏颇。在车辆行走过程中，悬架的参数起着至关重要的作用，一些不当的调节，直接结果就是建图精度低，导航效果差；而合适、得当的调节，能让车辆行走路线更加完美，同时也大幅度降低车辆的机械损耗。对于此车模，我们的主要调节点有：

- 前后车高
- 主销后倾角
- 主销外倾角
- 前后束角
- 避震油标号
- 差速油标号
- 避震弹簧行程
- 舵机连杆

其中主销后倾角、主销外倾角、前后束角、舵机连杆与比赛常规竞速组车模的调节方法类似，本文不再赘述，主要讨论另外 4 点。

2.1.1 前后车高

大脚车由于被设计为经常在崎岖地面行走，其悬挂系统有比较大的行程。但本次比赛为在平坦地面运行，过高的悬挂系统行程有以下问题：

- 会抬高车辆重心使得车辆稳定性下降，增加车体控制难度
- 导致车体在加减速时晃动加大，给 IMU 模块及激光雷达带来不必要的噪声
- 使得轮胎传动轴与轮胎及传动轴心行程比较大的角度，降低了传动效率

在调节此项时，我们将车体放置于水平表面同时将车上所有装备装好。经过观察，我们选择前车高与后车高保持一致，同时尽量使得传动轴保持水平。经过调整，车模工作良好。

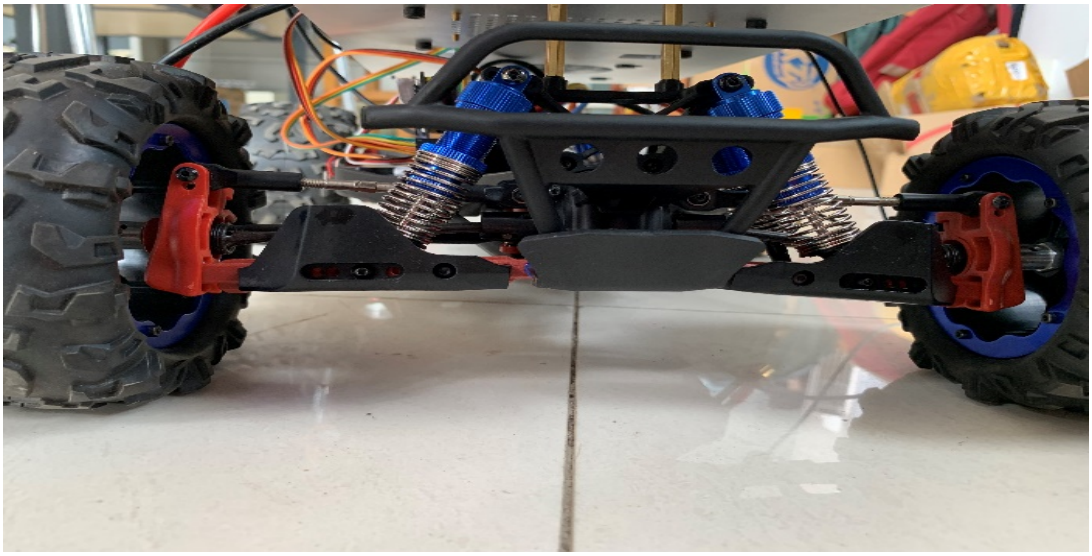

图 2.1 前悬挂

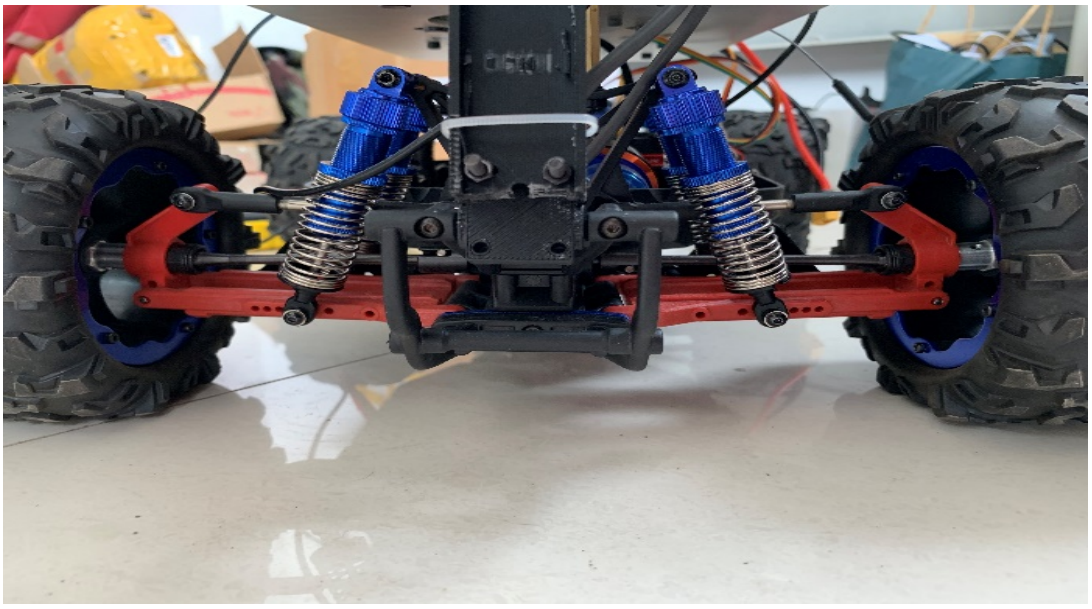

图 2.2 后悬挂

2.1.2 避震弹簧行程与避震油标号

由于这两项皆为避震系统相关，因此放在一起讨论。

大脚车避震系统的两大核心部件避震器和弹簧。其中，弹簧是主要的缓冲器，用以吸收冲击和震动；阻尼器则是用来约束弹簧，使其工作的更稳定。但这种结

构拥有急回特性，当弹簧被压缩后再放开，它在恢复原来的长度后，还会随惯性继续伸长；到达临界点后，又被向回拉，缩到比原长度更短。而且，这个过程并不是线性的。如此往复若干次后恢复平静，这在物理学中被称为阻尼效应。阻尼器其实就是提供阻尼，使弹簧更快的恢复原来的长度，不再来回来震荡。其中本次使用的车模避震为油压避震，油压避震的是通过避震油流过避震活塞上的小孔从而带来阻尼效果。

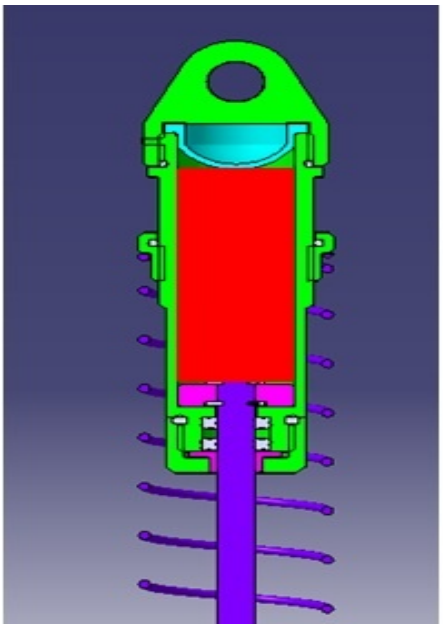

图 2.3 油压避震器原理图

在避震系统上，弹簧的长度是可以调整的。通过改变弹簧长度可以改变避震系统的动态力学性能，调长可使避震力度减弱，调短可使避震力度加强。通过实际外场测试我们发现，本车模在装配上导航系统后，车辆整体重心偏后，也就是整个车辆各个模块的安装位置使得车体前部重量较小，重量较为集中在后半部。为了尽量避免加速度过大时车体出现晃动，我们调节后部弹簧阻尼器使其力度加强，前部弹簧阻尼器力度减弱，即前部弹簧长度增加，后部弹簧长度减小。在调整弹簧长度时，为了避免左右轮避震力度不一致，我们不仅利用游标卡尺精确测量弹簧长度，也通过标准砝码块测试其弹性力以确保左右一致。

为了满足车辆在不同地面上运行，避震器内部的避震油有多种标号，其标号越高，浓稠度越大。避震油稀在颠簸路面有更强的抓地力，但是在转向时因为外侧车轮缺乏瞬时的支撑力有更弱的转向能力和反应速度。否则反之。一般来讲，越颠簸的场地用越稀的避震油，越平整的场地用越重的阻尼。我们比赛的场合为平坦地面且弯道较多，同时需要尽量避免车体晃动。经过尝试，我们选用了美国 AE 的 45 号避震油。

图 2.4 各种编号的 AE 避震油与差速油

2.1.3 差速油标号

大脚车的差速有三组，分别为：

- 前差速，负责两个前轮差速
- 中差速，负责前轮与后轮差速
- 后差速，负责两个后轮差速

其中前差速与后差速作用相似，差速油浓度高则车体稳定，但过弯时易推头；差速油浓度低则车体灵活，但易发生甩头甩尾车体不稳定。我们比赛时的场地抓地力强，同时需要稳定性大于灵活性，因此应选择浓度适中或者偏高的差速油。

对于中差速，差速油的浓度决定电机的动力传至前轮与后轮的差别，浓度越高甚至锁死中差速，则车辆越易在加速过程中抬头，因此我们应该选用浓度较低的差速油。经过多次挑选与尝试，我们前后差速使用美国 AE 的 7000 号差速油，中差速使用美国 AE 的 3000 号差速油。

图 2.5 大脚车模抬轮示意图

2.1.4 其他

除了以上几点外，车模在机械结构上仍有许多细节需要注意，如为了避免螺丝因震动而松开应采用防松螺母或者利用螺丝胶固定、为了减少 IMU 震动应加入避震措施等。

2.2 传感器的模块安装支架设计

由于本车模设计之初并未考虑深度相机以及编码器的安装，车模主体框架也没有多余安装位置及孔位。所以在充分研究了车模的机械结构、相关部件的材料选用以及受力情况分析，同时保证不修改车模主体框架又考虑其余传感器的尺寸及安装规范后，设计了深度相机支架以及编码器的安装部件。

2.2.1 深度相机支架

对于深度相机的安装来说，首先要考虑其摄像头视野与其他模块的干涉问题。因为毕竟深度相机最大实体尺寸相对于车模并不小，而且车模本身又安装了激光雷达、IMU 等模块，还要考虑深度相机对这些模块的干涉。所以，这个问题就需要充分考虑到所有传感器的工作状态。最终，我们决定将深度相机安装在激光雷达与 IMU 模块之间的位置，并且用车模本身孔位固定。在确定了设计思路以及要注意的问题之后，我们用 UG 进行了结构设计并且用 3D 打印完成了相关零件的制作。在安装了所有模块之后，测试所有模块的工作状态，从返回数据来看，所有模块工作正常，各模块间没有干涉。

2.2.2 编码器支架

而速度编码器的安装就显得更为棘手，因为其必须要与地面进行接触，所以对结构受力要求更高，但更强的结构意味着更大的重量，不过，在经过分析编码器安装支架具体的载荷分部情况后，我们得出了结论：其主要受弯曲应力以及扭转力。在进行了具体的受力分析以及综合计算后，再结合 3D 打印材料 ABS 的许用应力，并且考虑到实际工作中的一些不可控的外界因素，在保证最低受力要求的情况下，加入一个 3.5-4 左右的安全系数，最终确定了编码器安装部件的截面形状和尺寸，以及 3D 打印时选用的精度和填充度。

在完成上述模块的制作安装之后，同样在外场进行了模拟测试并加入外界环境干扰量，整个编码器安装结构稳定可靠，没出现形变和断裂，并且，独特的结构设计也保证了与编码器连接的摩擦轮时时刻刻可以与地面稳定接触，发回的数据连续可靠，即使在大角度转向或加速度过大时，依旧可以保证很好的工作状态。

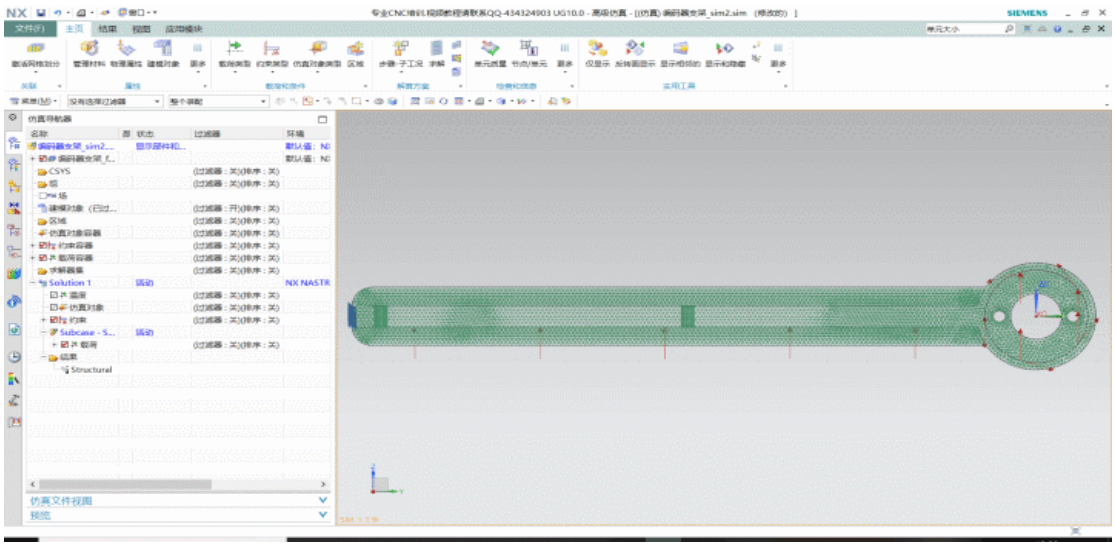

图 2.6 编码器支架

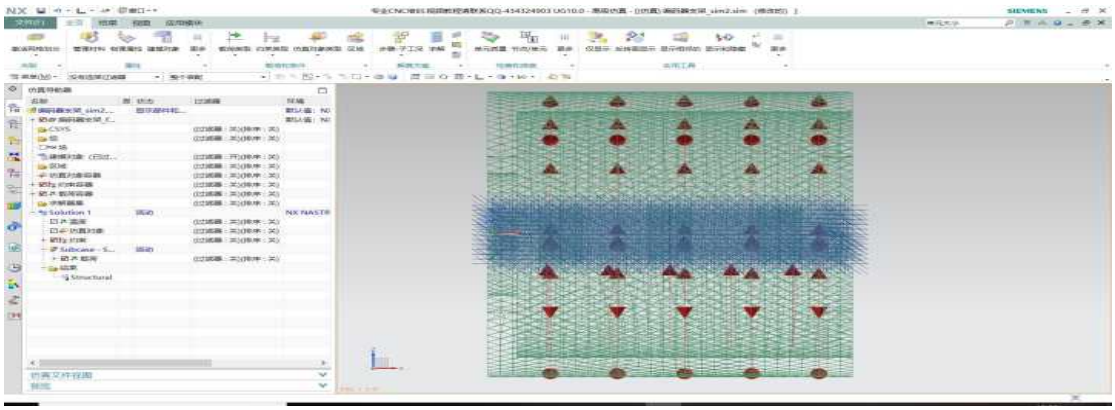

图 2.7 编码器支架受力分析

第三章 硬件系统设计及实现

3.1 硬件设计方案

系统的硬件电路是整个系统的基础，也是软件平台得以稳定运行的基础，所以硬件电路的设计是非常重要的。在硬件系统的设计过程中，本着系统安全、稳定的原则，电路设计尽量模块化，主要包括单片机最小系统、单片机供电模块、雷达供电模块、速度检测反馈模块、USB 转串口模块、舵机接口、电调接口、IIC 接口、SPI 接口、辅助调试模块等。

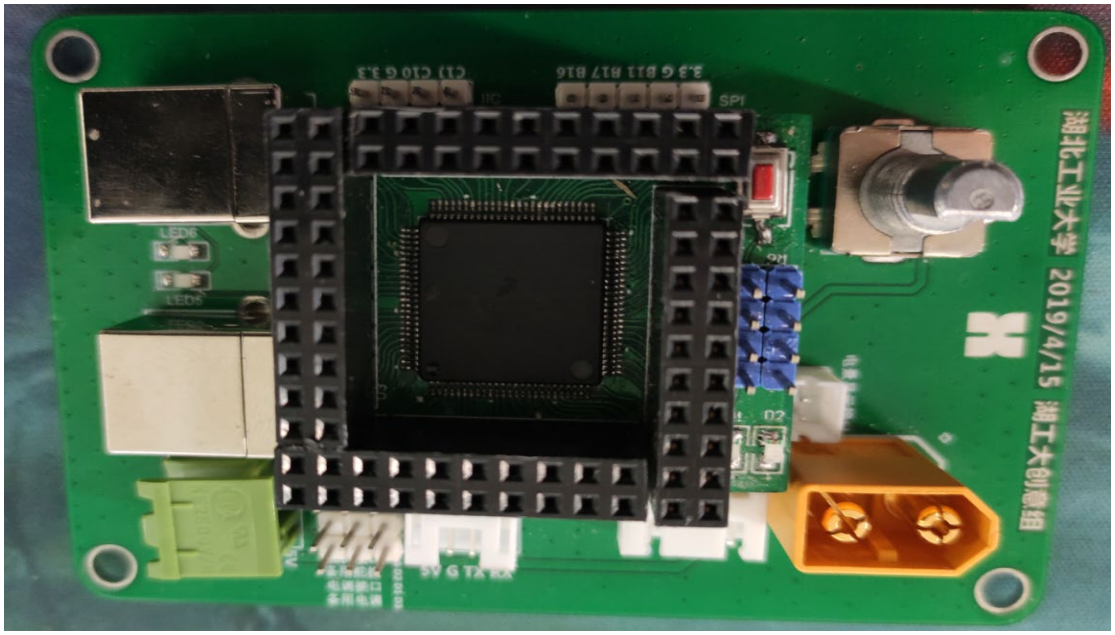

图 3.1 硬件设计

3.2 控制板的硬件设计

3.2.1 单片机最小系统

单片机最小系统采用 MKV58F1M0VLL24 芯片，该最小系统采用 3.3V 供电，负责处理部分雷达信息以及对舵机和电机组成的运动平台的控制，使其能完成车体运动。单片机最小系统如下图所示。

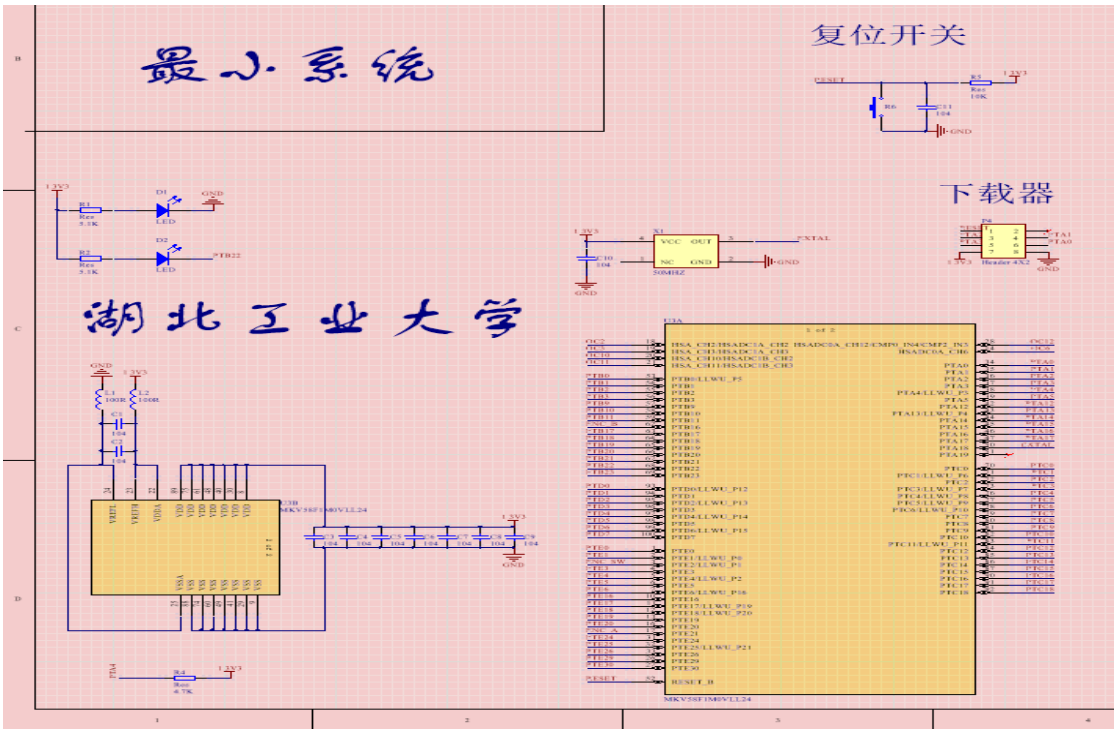

图 3.2 最小系统

3.2.2 电源管理模块

考虑到功耗影响和电压需求差别，本车模上使用 3 路供电电源：

1）智能车车体控制使用 2S 锂电池，锂电池可直接用于电机供电，锂电池分别给控制板和电机供电。

2）雷达使用直流 5V，5V 电源选用 DC-DC 芯片 TPS563201DDCR。其原理图如下。

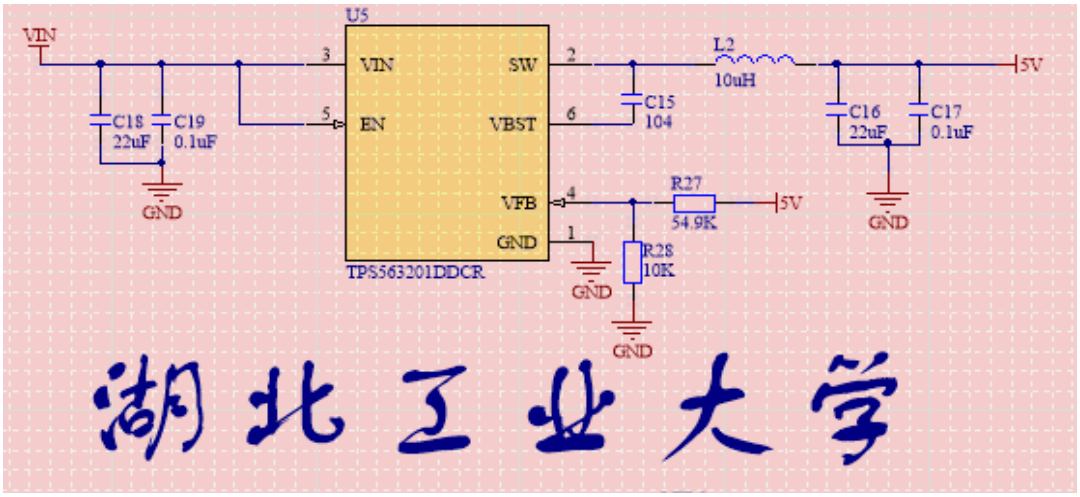

图 3.3 TPS563201DDCR 原理图

3）使用 3.3V 为单片机供电，采用线性稳压芯片 LP38691_NGG_6。其原理

图如下。

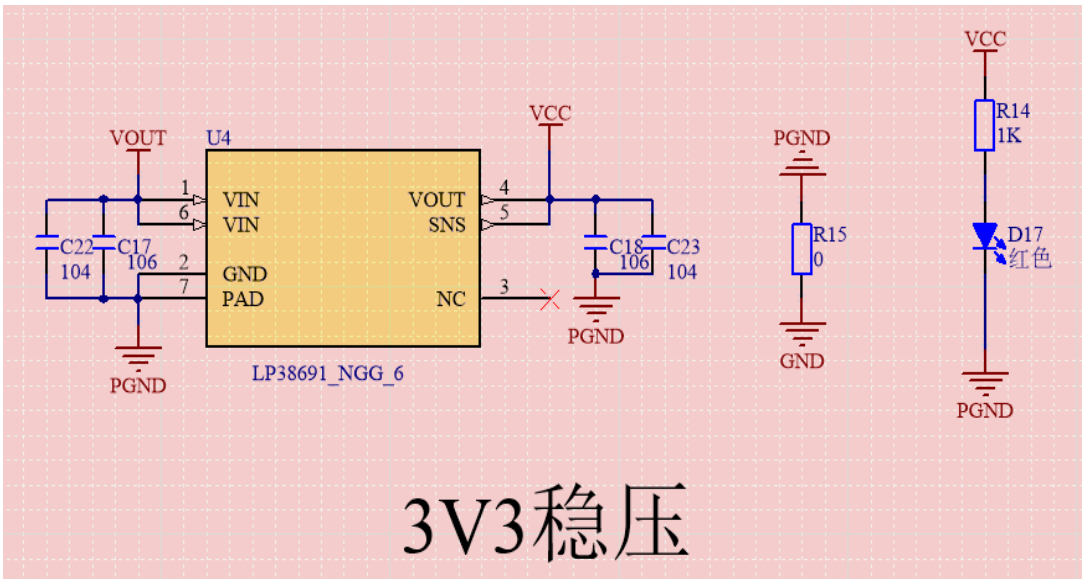

图 3.4 3v3 稳压模块

4）舵机模块使用直流 6V，使用线性稳压芯片 LM1084。其原理图如下。

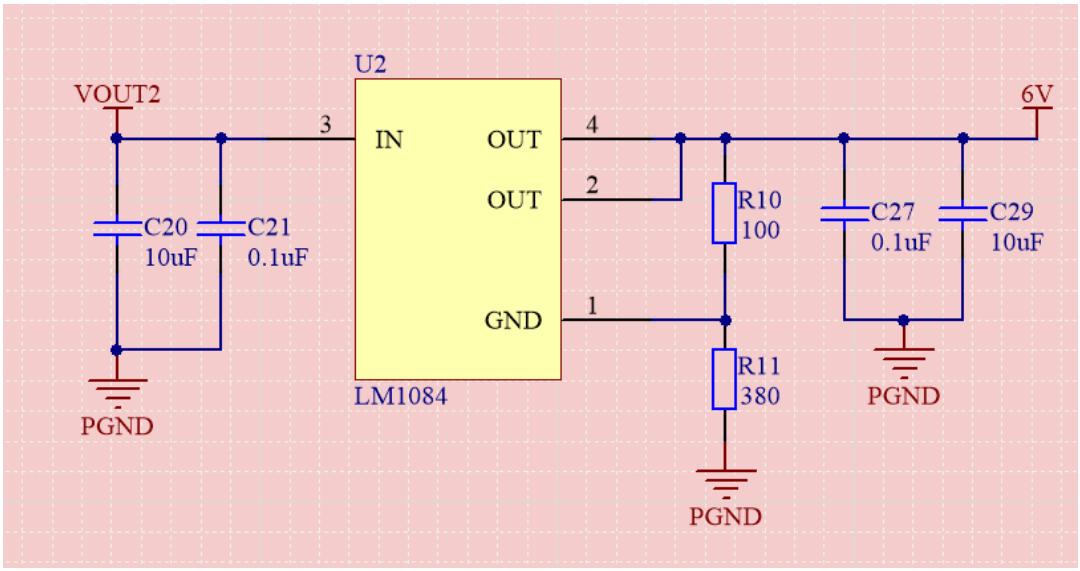

图 3.5 舵机稳压模块

3.2.3 USB 转串口模块

使用 CP2102 芯片作为 USB 转串口模块。CP2102 其芯片集成度高，内置 USB2.0 全速功能控制器、USB 收发器、晶体振荡器、EEPROM 及异步串行数据总线（UART）接口，支持调制解调器全功能信号，无需任何外部的 USB 器件。CP2102 与其他 USB-UART 转接电路的工作原理类似，通过驱动程序将 PC 的 USB 口虚拟成 COM 口以达到扩展的目的。

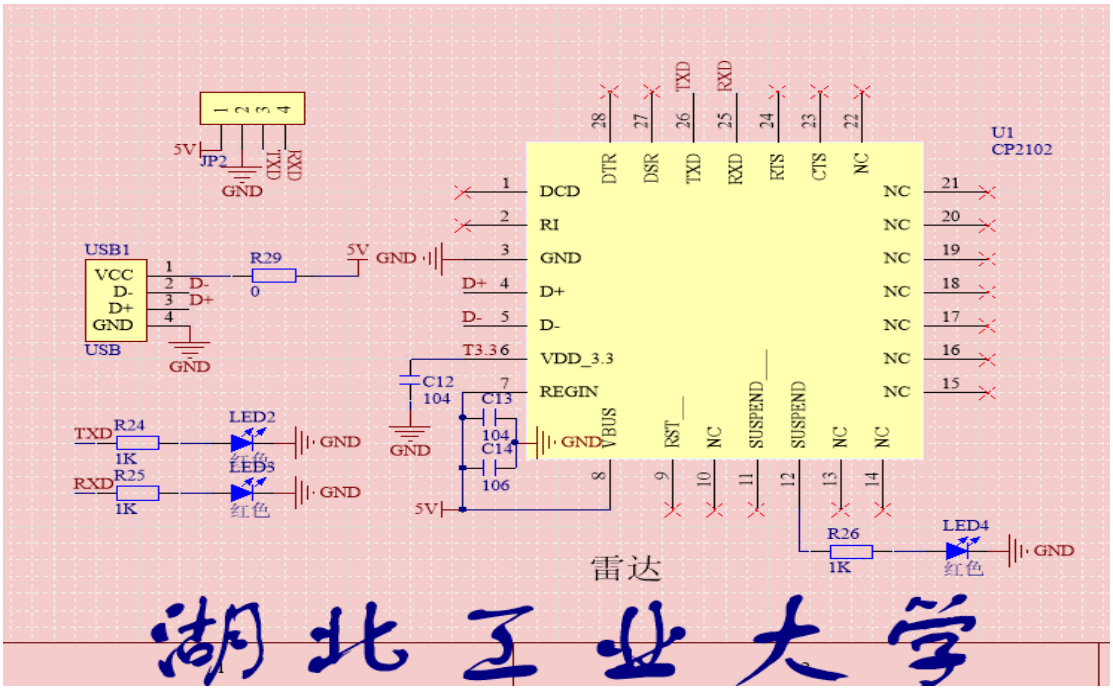

图 3.6 USB 转串口模块

3.2.4 人机交互

人机交互：增加旋转编码器用于输入参数，策略调整，加入液晶屏显示车辆状态便于调试。

1） 旋转编码器原理图如下

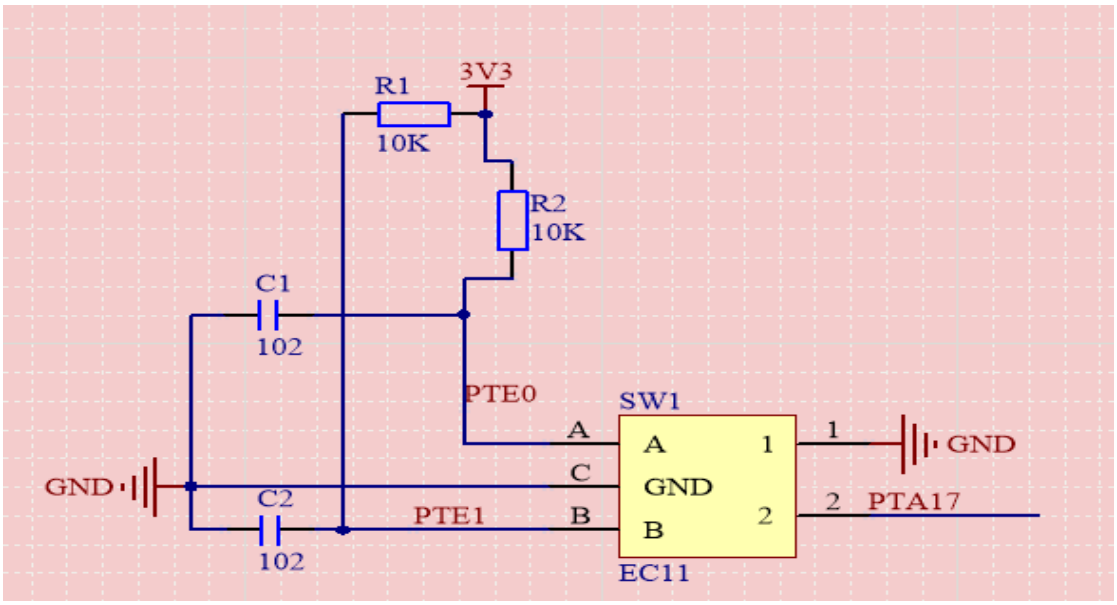

图 3.7 旋转编码器原理图

2）液晶屏显示原理图如下

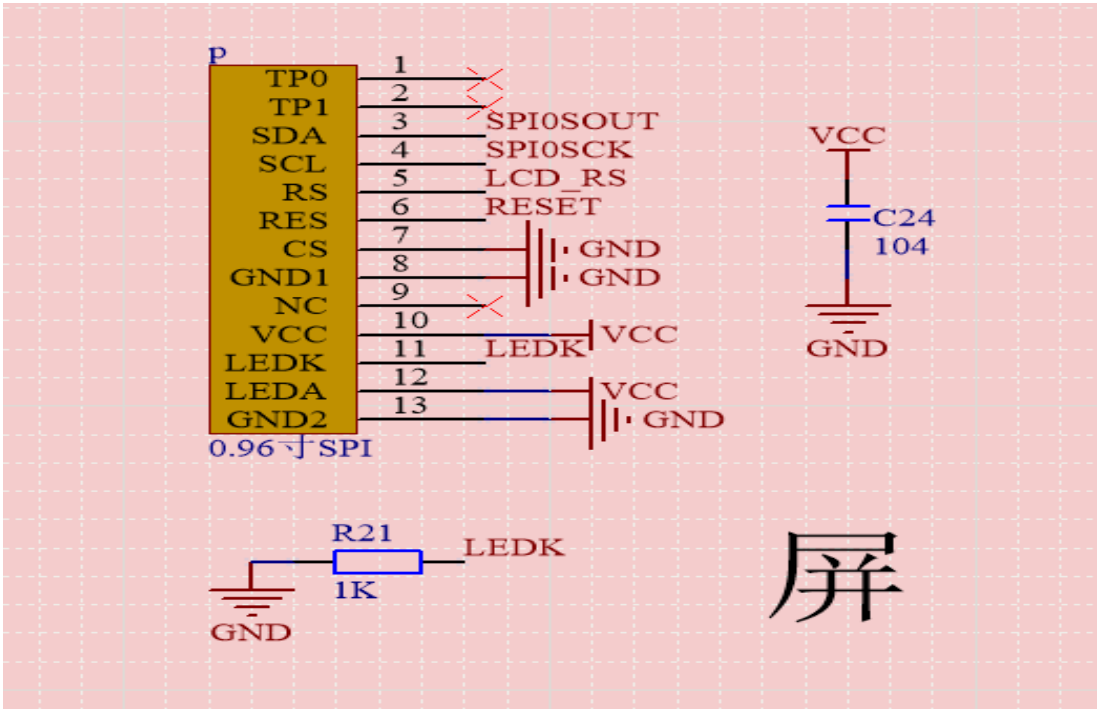

图 3.8 液晶显示原理图

3.2.5 陀螺仪

集成陀螺仪与磁力计，辅助检测车体状态。

1） 陀螺仪使用 ICM20602，原理图如下。

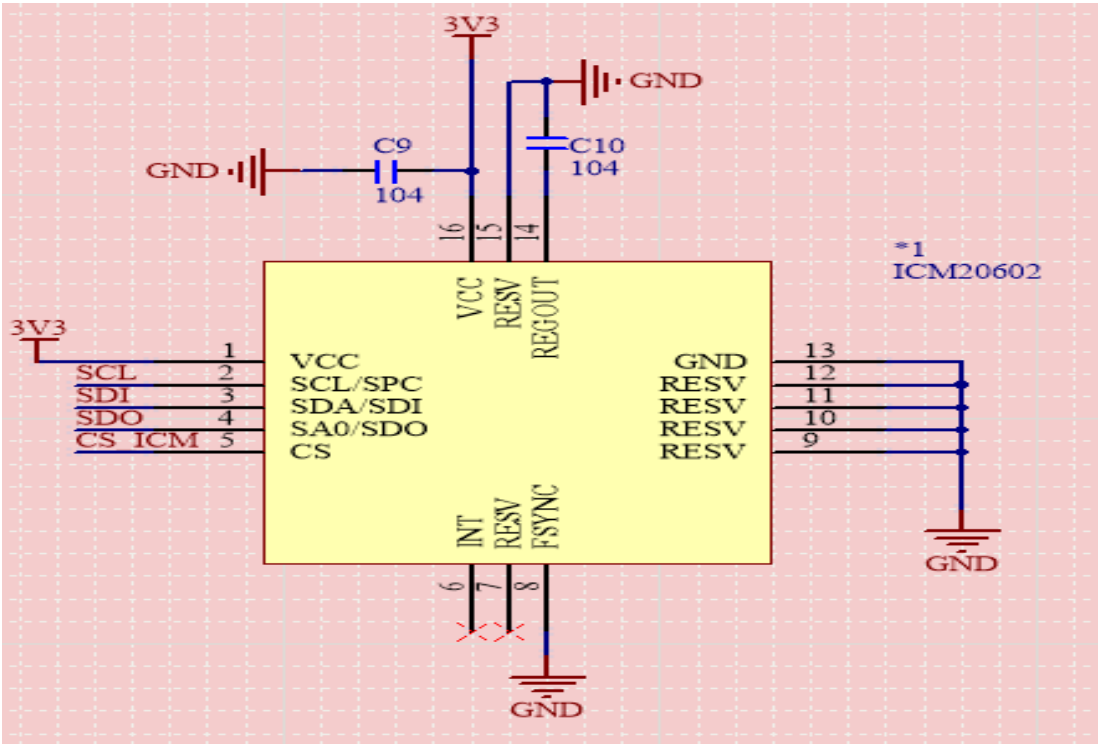

图 3.9 ICM20602 原理图

2） 磁力计使用 AK8975，原理图如下。

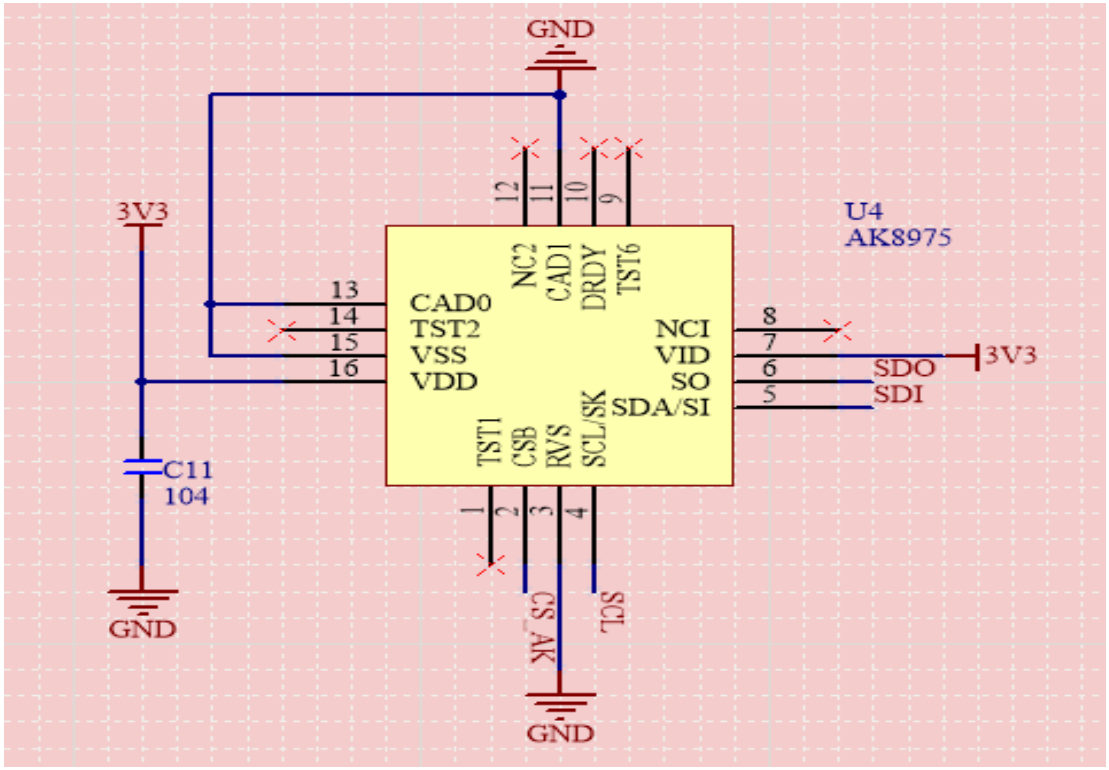

图 3.10 AK8975 原理图

3） 由于陀螺仪存在温飘，我们使用加热电阻进行恒温控制。

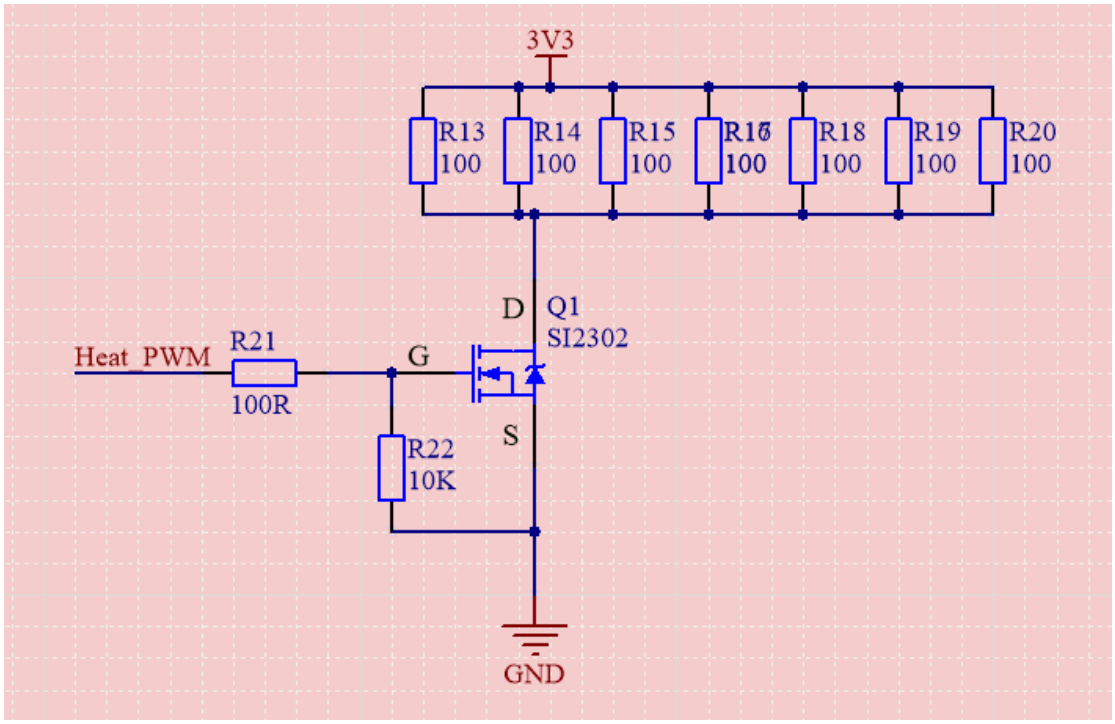

图 3.11 温控

第四章 车体控制器

随着车速的提升，对车体基本控制的要求逐渐升高。为了提高车体的控制性能，我们使用恩智浦（NXP）公司的 MKV58F1M0VLL24 作为车体控制的微控制器，MKV58F1M0VLL24 是一款具有 240 MHz 的 Cortex-M7 内核的高性能微控制器。得益于其最小达 260ps 的高精度 FlexPWM，在电调与舵机控制方面具有巨大的精度优势。丰富的 USART 和 FlexCAN、IIC 等资源使得在芯片与 PC 端、陀螺仪等通讯发挥巨大作用。

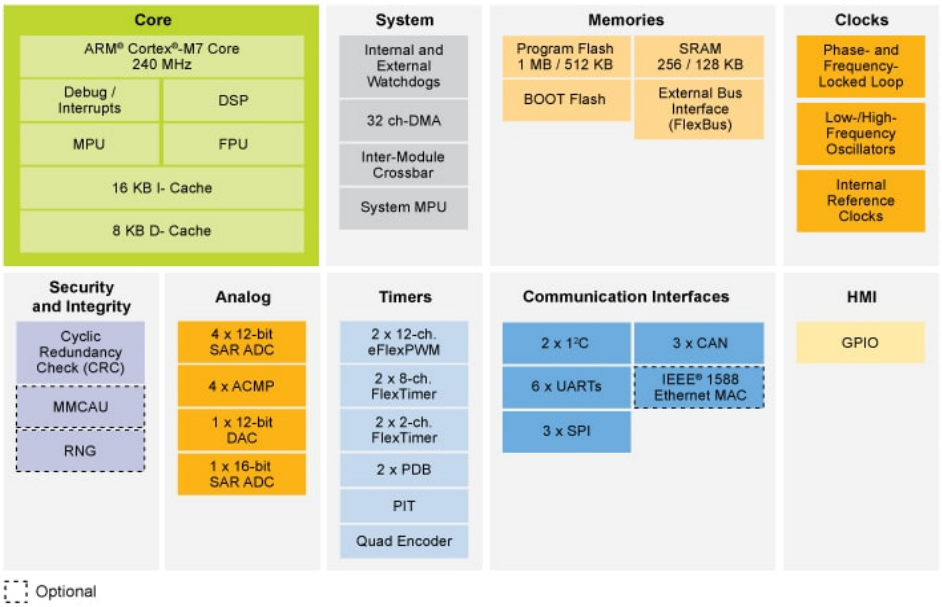

图 4.1 KV58 芯片系统框图

4.1MiniPC 与车体控制器通讯

原车体控制器使用串行通信与 MiniPC 通讯。考虑到串行通信具有数据传送效率低，数据易失真的特性，在行车时，特别是在室外恶劣环境中极其强烈的电磁干扰下，串行通信出错率尤为高。使用 CRC 和软件计算校验虽说可以降低出错率，但是难以在根本上确保信息的准确性。为了避免这种情况，本车采用控制器局域网络(Controller Area Network, CAN)进行交互通讯。它是一种多主总线，即每个节点机均可成为主机，且节点机之间也可进行通信。CAN 协议采用 CRC 检验并可提供相应的错误处理功能，保证了数据通信的可靠性。CAN 总线所具有的卓越性能、极高的可靠性和独特设计，特别适合工业设各测控单元互连，因此备受工业界的重视，并已公认为最有前途的现场总线之一。

4.2 车体速度与舵机控制

因为有电子调速器的存在，所以就可以只需要产生一个 PWM 信号来驱动电子调速器，从而让电子调速器产生三相交变电流，驱动电机。得益于微控制器的高精度 PWM 模块，选取使用 FlexPWM 产生高精度的 PWM 控制电子调速器，使得 PWM 更加精准，控制效果更好。 同样的也产生一路 330HZ 的高精度 PWM 用于舵机的控制。

4.3 车体运动状态的获取

对于车体控制来说，实时监控车体的运动方向、速度、加速度尤为重要。车体控制需要获取这些基本信息进行分析综合，以计算获得更好的控制策略。

在车体控制器中的板载 IMU 芯片虽说性能、精度上均比不上车体上配套的 IMU，但是在电机控制分析与车体紧急情况保护中可以发挥其作用。通过微控制器读取 IMU 的六轴数据可以发现，芯片提供的数据夹杂有较严重的噪声，在芯片处理静止状态时数据摆动都可能超过 2%。除了噪声，各项数据还会有偏移的现象，也就是说数据并不是围绕静止工作点摆动。因此获得准确的数据前要先对数据偏移进行校准，再通过滤波算法消除噪声，经低通滤波、卡尔曼滤波等数字滤波器数据处理后，加以磁罗盘进行角度融合修正，最终获取车体姿态的四元数。

在 MEMS 传感器中，陀螺仪能够在短时间内生成高精度姿态估计数据，而加速度计和磁力计生成的数据相对较差。陀螺仪由于随时间的漂移会引起积分误差，导致姿态精度逐渐降低。相比之下，加速度计和磁力计的测量误差不会增加。因而陀螺仪的动态特性优于加速度计和磁力计。根据不同传感器的优势与劣势，我们采用了扩展卡尔曼算法(EKF)融合加速度、磁力计和陀螺仪的信息，以提高姿态角的测量精度和控制系统的稳定性。然而，由于低通滤波器的阻带衰减很慢，噪声变大。此外，如果滤波器的参数是固定的，则难以实现最佳估计值。进行姿态确定，提高姿态数据的融合精度。

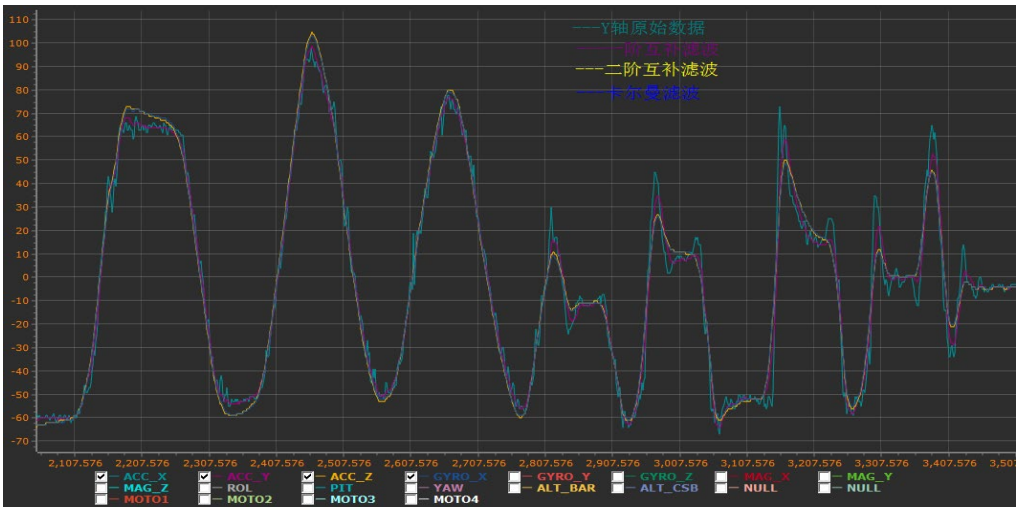

图 4.2 各类滤波器效果图

在速度采集方面有多种方式。其一便是最简单的直接读取来自有感无刷电机的霍尔传感器的信号，通过 MCU 自带的 ENC 进行正交解码而获取其单位时间内的转速，从而间接的测得车体的运动速度。但是由于车体在硬件结构上有硬件上的差速器，使得当转向时，有感无刷电机的转速并不等效于车体运动速度。尤其在四轮摩擦力相差较大的情况下，车体运动速度与采集的电机速度相差甚远。其二是根据板载的 IMU 读取车体的加速度，通过等时长累加积分，可以获取车体的速度大小。但是根据实验测得，由于电子传感器采集到的数据具有不可避免的误差，使得长时间的加速度积分计算出的速度有较大的积分漂移，且随着积分时长的增大，积分漂移现象尤为严重。所以通过加速度积分只能评估短时间内的速度而不能真正的获取车体运动的实时速度。

当车体基于恒加速度的加速运动时，采集加速度计的前向加速度数据放入 MATLAB 中分析

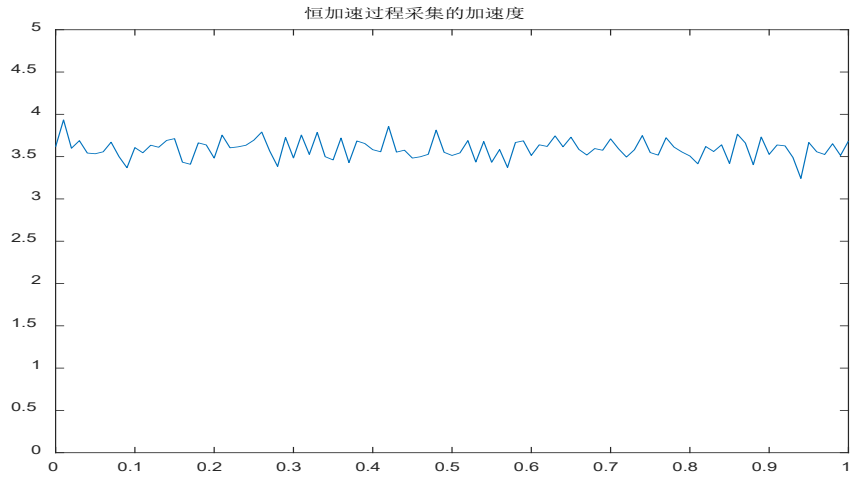

图 4.3 加速度计采集的加速度

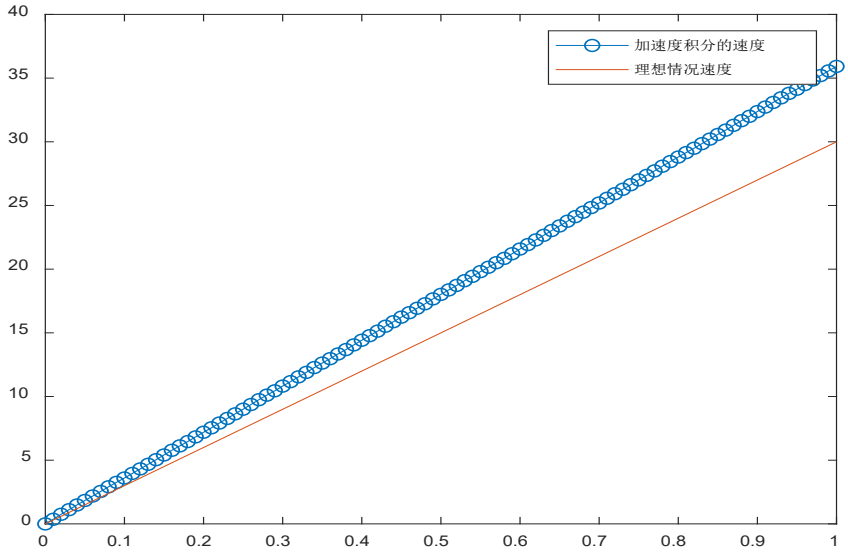

图 4.4 加速度积分效果图

可以从实验结果上分析，可能存在恒常数的影响，加速度计积分会有一定的积分漂移。长时间通过积分计算得到的速度效果不佳。

最后通过前面多次试验改进，我们最终选择在车体后加上一只连接有欧姆龙光电编码器的无动力轮，通过其与地面的摩擦，与车体保持同速。通过利用微控制器的正交解码功能直接读取光电编码器的 A/B 相的正交编码信号，从而获取其转速。通过实验证明，该方式较为良好地反映了真实的车体运动速度。

4.4 车体紧急情况处理

当外部原因干扰或是程序失效下，MiniPC 端发送数据会出现错误或是通讯断开的情况，此时车体可能会失去控制或是出现严重颠簸情况，需要进行紧急的车体运动干预处理。当车体控制器检测到异常的加速度情况：例如一段时间内车体持续以高速运行、有不合理的加速度或是碰撞产生的反向加速度，车体控制器会根据不同的情况进行减速或是紧急刹车的处理，且会持续向主机通讯并蜂鸣器报警。其次，在车辆调试过程中当操作人员察觉出现异常情况，也可通过 SBUS 接收来自遥控器的信号，加以人为干预，使得车体具有更好的操控性能和保护机制。

4.5 硬件上的改进

在调试车辆时，会经常插接电池。然而电池的 T 插接口在频繁接入拔出的过程中可能会出现接口松动的情况，甚至在刚上电的瞬间会出现电火花。为了解决这个问题，我们更换成更加稳定、更高额定电流的 XT60 与 XT90 接头接口。电调采用 XT90 接口以获得更低的导通电阻，控制部分采用 XT60 接口从而避免与电调接口混乱、实验表明，更换接口后，可以有效的解决插拔时产生电火花，接触不良的现象也得到避免。

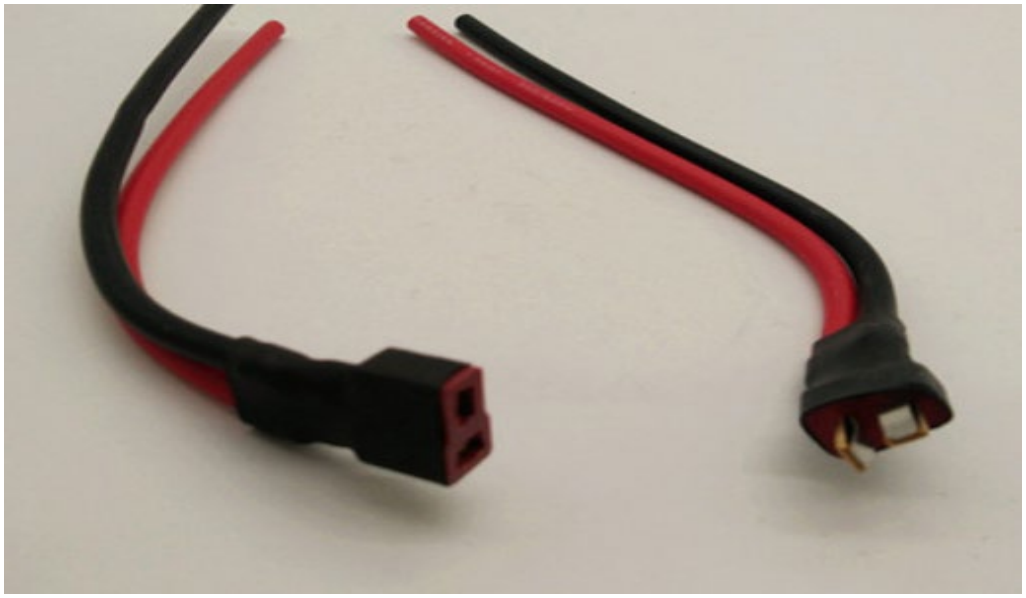

图 4.5 T 型电源接口

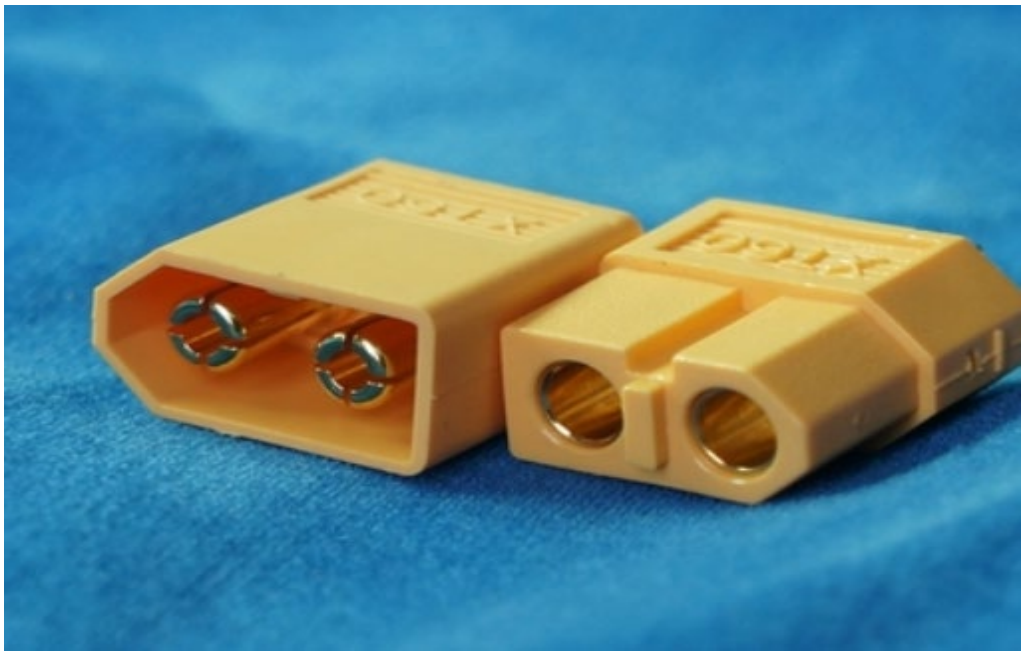

图 4.6 XT60 型电源接口

第五章 无人车室外环境路径规划

路径规划需要综合机器人本体情况、环境情况以及相关算法，它是机器人实现未知区域探索的重要方法，指在动态或静态环境下，依照评价准则，在不发生危及机器人自身安全的前提下，使机器人由起始位置探索、规划出一条能够安全到达目标位置的轨迹。

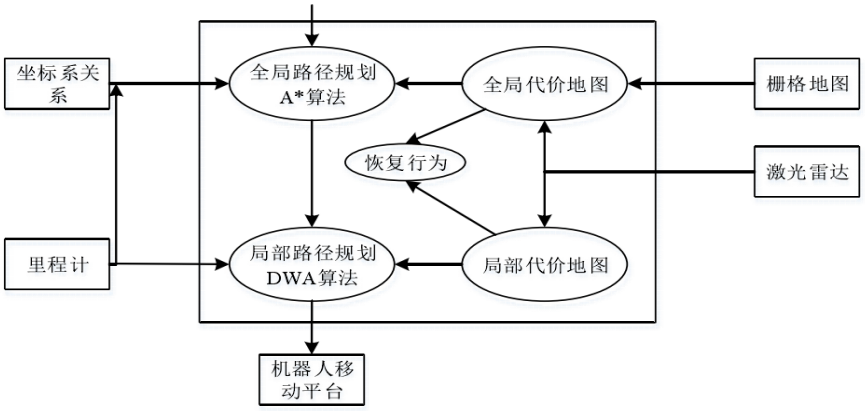

图 5.1 路径规划框图

路径规划的方法有全局路径规划和局部路径规划两种。通常情况下，机器人利用现有的地图，使用全局规划，避开地图上已知固定障碍物信息或可预见移动障碍物；而在未知环境或动态环境中，如存在可移动的障碍物、行人等情况时，机器人则利用传感器信息，进行局部路径规划以避开障碍物。

5.1 移动机器人的运动模型

运动模型在求解无人车 SLAM 问题和导航问题中起着重要作用。一个刚性移动的无人车常用 6 个变量来描述无人车的移动，包括它的三维直角坐标和其对应的三个欧拉角（横滚，俯仰，偏航）。本文所使用的无人车是在室外平坦区域，因此可以将无人车的运动限定在二维平面。此时可以将无人车的运动学状态用向量描述，如公式（5.1），简称位姿。

$$X_t = (x, y, \theta) \tag{5.1}$$

其中 x,y 表示全局坐标系的坐标，θ 表示无人车相对于全局坐标系 x 的方位角。

无人车的运动学模型在求解 SLAM 问题中起着状态变换模型的作用，这个模型就是概率学上的条件概率密度，即公式（5.2）。

$$p(x_t \mid x_{t-1}, u_t) \tag{5.2}$$

公式中x_t和x_{t-1}都描述的上文无人车的位姿。u_t则是控制量。该公式描述的物理意义是在 t-1 时刻，无人车的位姿为x_{t-1}，对其施加控制量u_t后，机器人当前时刻位姿x_t的后验分布概率。其中在工程实践中，u_t是由里程计提供的。

5.2 速度模型

速度模型通常假设无人车控制电机的速度指令有两个：一个是 X 轴的平移速度，一个是旋转速度，通过这两个速度分量来控制无人车的运动。使用v_t表示 t 时刻机器人的平移速度，w_t表示 t 时刻机器人的角速度，则 t 时刻施加给机器人的控制量u_t可表示为公式（5.3）。一般规定逆时针旋转时w_t取正值，向前运动则v_t取正，反之取负。

$$u_t = \begin{pmatrix} v_t \\ w_t \end{pmatrix} \tag{5.3}$$

在控制量u_t的驱动下，无人车从 t-1 到 t 时刻将以某点圆心做圆周运动，当角速度 w 为 0 时，机器人做直线运动。此情况下，t 时刻无人车的位姿（x,y,θ）可由公式（5.4）得出。

$$\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} v\Delta t\cos\theta \\ v\Delta t\sin\theta \\ 0 \end{pmatrix} \tag{5.4}$$

真实的无人车速度既不可能以一速度跳变到另一速度，也不会在每一个时间间隔内保持不变。利用小时间间隔 Δt，在每段时间间隔内用瞬时速度近似无人车的真实速度是解决非匀速运动的常用手段。

5.3 里程计模型

运动建模中用的最多的模型是里程计模型。里程计通过整合轮子的编码信息得到，该模型用距离测量代替速度控制。实际测试证明，虽然里程计模型也存在着误差，但是该模型通常要比速度模型更加准确，并且里程计模型的计算是在机器人运动控制之后测量得出的，因此里程计模型有着比速度模型更加准确和易于实现的优点。

在里程计运动模型中，距离的测量以机器人本体为中心，而不是以世界坐标系为中心。即相邻时刻间测量的是机器人自身的相对运动。该信息是由机器人内部搭载的里程计提供。具体来说就是里程计能够给出机器人在[t -1,t]时间内，其位姿从x_{t-1}到x_t的变化量。即可提供从$\bar{x}_{t-1} = \begin{pmatrix} \bar{x} & \bar{y} & \bar{\theta} \end{pmatrix}$到 t 时刻

$\bar{x}_t=\left(\bar{x}' \quad \bar{y}' \quad \bar{\theta}'\right)$的相对前进。公式中的“-”表示位姿的改变是相对于其自身的，相对于世界坐标系的运动则是未知的，此时u_t可由公式(5.5)给定。

$$u_t=\begin{pmatrix}\bar{x}_{t-1}\\ \bar{x}_t\end{pmatrix} \tag{5.5}$$

为了计算里程计模型下的相对运动，控制信息 tu 的计算可分解为三个步骤：第一次旋转、直线运动和二次旋转。每一个这样的运动都可以具有唯一的参数向量$\left(\delta_{rot1} \quad \delta_{trans} \quad \delta_{rot2}\right)$。该参数向量可以表示机器人从 t-1 时刻的$x_{t-1}$到 t 时刻的$x_t$运动。其中第一次旋转$\delta_{rot1}$可由公式（5.6）给出。

$$\delta_{rot1}=a\tan 2\left(\bar{y}'-\bar{y},\bar{x}'-\bar{x}\right)-\bar{\theta} \tag{5.6}$$

机器人的相对运动间的平移距离δ_{trans}可由公式（5.7）给出。

$$\delta_{trans}=\sqrt{(\bar{x}-\bar{x}')^2+(\bar{y}-\bar{y}')^2} \tag{5.7}$$

第二次旋转可由公式（5.8）给出

$$\delta_{rot2}=\bar{\theta}'-\bar{\theta}-\delta_{rot2} \tag{5.8}$$

5.4 A*算法

A*算法是一种启发式的搜索算法，它与 Dijkstra 相比主要区别在于它在节点的代价函数中引入了启发函数 F(n)得到新的评价函数，如公式（5.9）所示

$$F(n)=G(n)+H(n) \tag{5.9}$$

式中，n 为当前节点，并且 n=(x_n, y_n)，即移动机器人从起点移动到当前点所需要的代价。其作用是使机器人向终点的可能方向移动，以减少不必要的访问位置点，从而减小算法的时间和空间复杂度。启发函数的选择直接影响到算法的执行效率，直至路径规划算法的完备性。在机器人路径规划中，有两种表示两点之间距离的方法，一种是欧式距离，它以两点之间直线最短为原则，因此，欧式距离为两点的连线得到的几何距离。另一种是曼哈顿距离，即机器人在栅格地图中只能沿着栅格进行移动，而这些栅格就像曼哈顿的街道一样，因此称作曼哈顿距离，由于机器人建立的是栅格地图，因此使用曼哈顿距离来表示两位置点之间的实际距离更为恰当。在曼哈顿距离中，当前位置点与其扩展位置点的距离关系如图 5.2 所示。

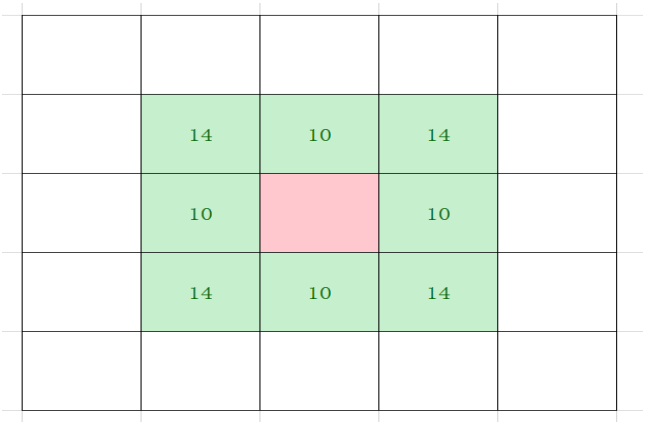

图 5.2 当前点与扩展位置点的距离关系

粉红色栅格表示当前位置，8 个栅格绿色栅格称为子位置在当前位置的扩展位置点。图中子位置点左下角数字为其与父位置点之间的距离，本文中采用近似的整数来计算，取$\sqrt{2}d = 14$，其中 d 为扩大倍数，取 d=10。

在公式（5.10）中 F(n)是从起始位置点 S 经过位置点 n 到达目标位置点 G 最小代价的评价函数。G(n)是当前路径中从起始位置点到点 n 的实际移动代价。

$$G(n) = \begin{cases} dist(S,n) & (if\ parent(n) = S) \\ g(parent(n)) + dist(parent(n),(n)) \end{cases} \quad (5.10)$$

式中，$parent(n)$ 是当前位置点的父位置点。

H(n)是最小代价启发式估计函数，如公式（5.11）

$$H(n) = d * (abs(n.x - G.x) + abs(n.y - G.y)) \quad (5.11)$$

第一步的搜索结果如图 5.3 所示

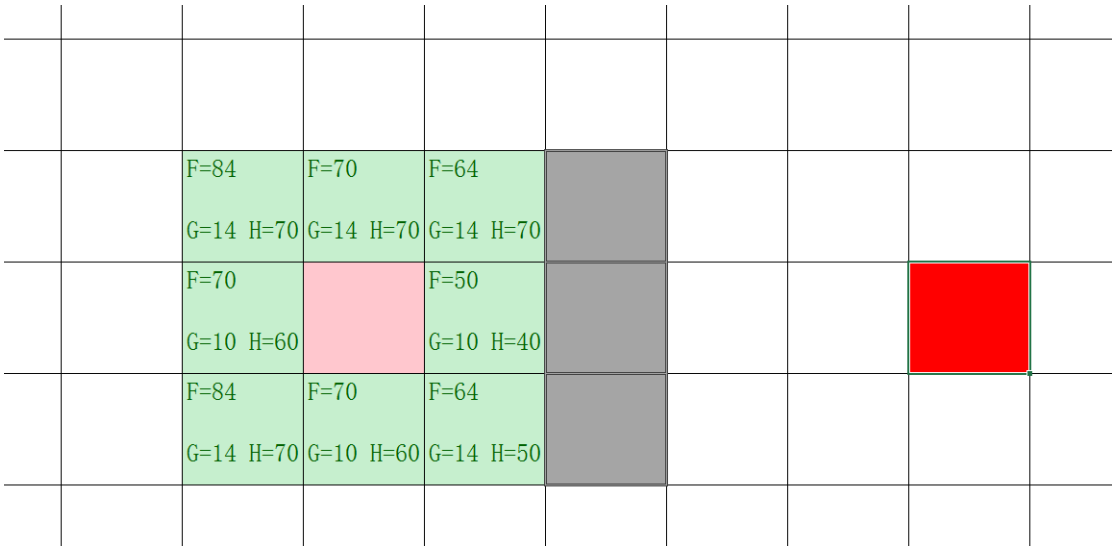

图 5.3 搜索结果示意图

在图 5.3 灰色的部分为障碍物，红色的部分为目标位置点。在栅格中写入 F、G、H 的评分， F 被打印在左上角， G 在左下角，H 则在右下角。

选取 F 值最低的子位置点进行扩展，并将此位置点加入到关闭列表中，依次类推，直到目标节点 G 被加入到关闭列表中，然后从目标点 G 开始，依次根据指针朝其父位置点移动，最终回到起始点，产生一条最短的路径。

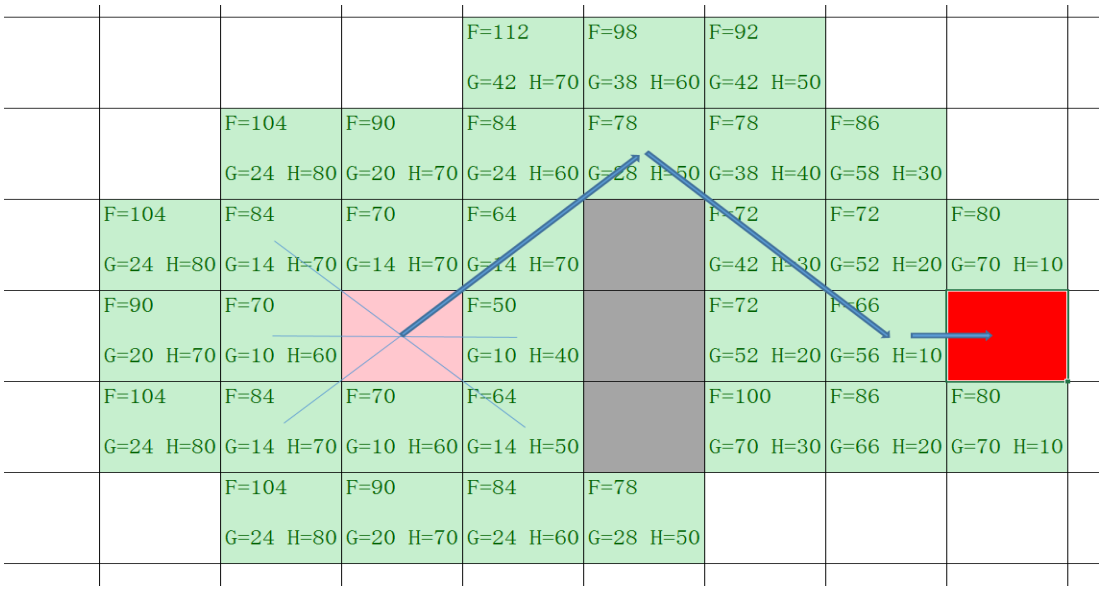

图 5.4 搜索结果

在图 5.4 中蓝色箭头线段即为最终产生的路径。由于采用启发函数对从当前位置点移动到目标位置点的代价进行估计，其搜索范围变小，效率得到提高。

5.5 DWA 局部路径规划

DWA(Dynamic Window Approach)算法，即动态窗口法，它是机器人进行局部路径规划的一种重要算法，在 1997 年由 D.Fox，W.Buegrad，S.Thrun 等人提出。在 ROS 中主要采用的是 DWA 算法。

DWA 算法的主要思路是：

- 在速度空间(v,ω)中对线速度v和角速度w进行多次采样，获得多组值；
- 模拟计算机器人在这样的速度条件下一段时间的机器人的运动轨迹；
- 引入评价函数，使用评价函数对模拟预测的多组轨迹进行评价，并且选取最优轨迹所对应的速度条件，使用它来控制机器人的运动。

DWA 算法的突出优点在于其动态窗口，它能够根据机器人驱动机构的加减速性能把速度采样空间限定在一个可行的动态的范围内。

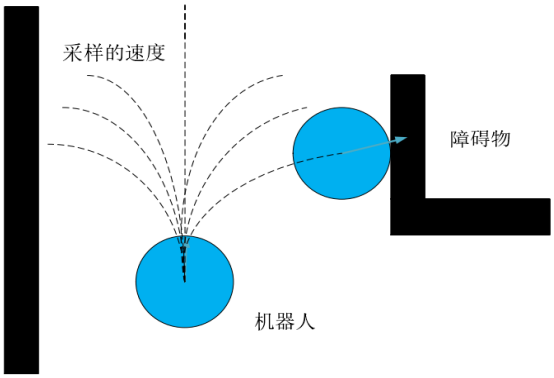

图 5.5 采样空间示意

5.6 无人车静态环境路径规划

本无人车使用 gmapping 算法建立的走廊地图作为已知地图，并且在该环境下无移动障碍物。

首先启动无人车的 ROS 系统，再使用 roslaunch 调用路径规划算法和地图文件。并在 rviz 中显示出来，随后使用 rviz 的 2D Pose Estimate 来修正无人车在地图中的位置与位姿，然后再设定目标位置，那么无人车便会根据设定的目标点规划出一条路径并结合 AMCL 修正无人车的位姿，并控制无人车运行到目标点。

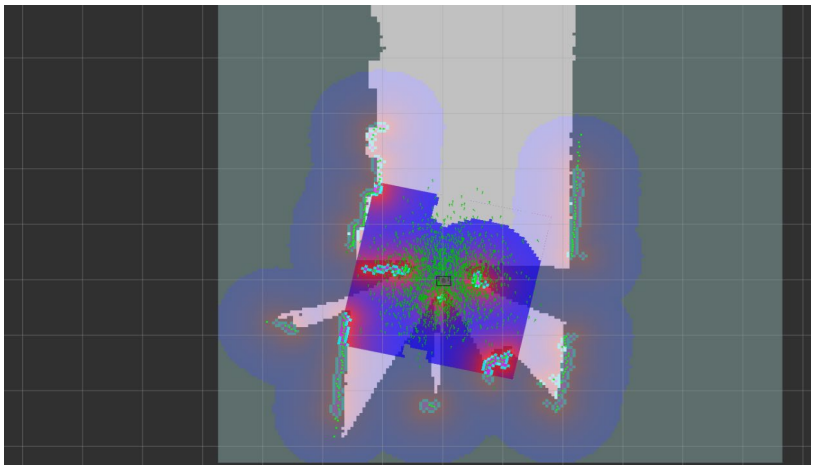

图 5.6 无人车位于初始位置

无人车在启动导航后，使用已知地图数据对自身位姿进行估计，结合 IMU 数据和 AMCL 算法修正自身位姿。

无人车使用全局路径规划器（Global Planner）计算出移动至目标点的轨迹，如图 5.7 所示，图中黑线标出的为无人车的轨迹。

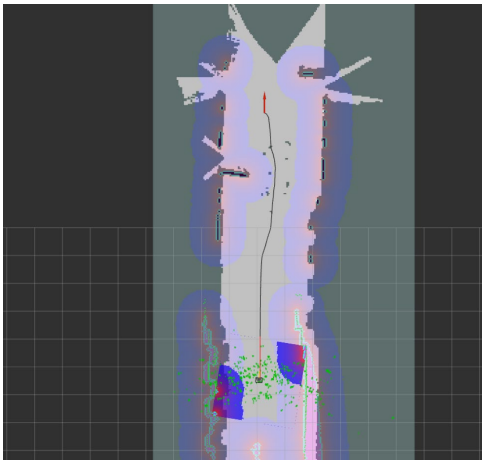

图 5.7 无人车路径规划

如图 5.7，无人车成功的按照规划好的路径移动到目标点。

5.7 无人车动态环境路径规划

本次比赛是能够在原有不存在的障碍物的地图中进行自主导航，因此需要在地图上随机设置障碍物，验证无人车的路径规划算法。

为了测试无人车使用局部规划算法进行动态避障并移动到目标点的能力，使用纸箱作为障碍物，如图所示放在机器人全局规划路径上用来测试无人车的自主导航能力。

图 5.8 障碍物设置

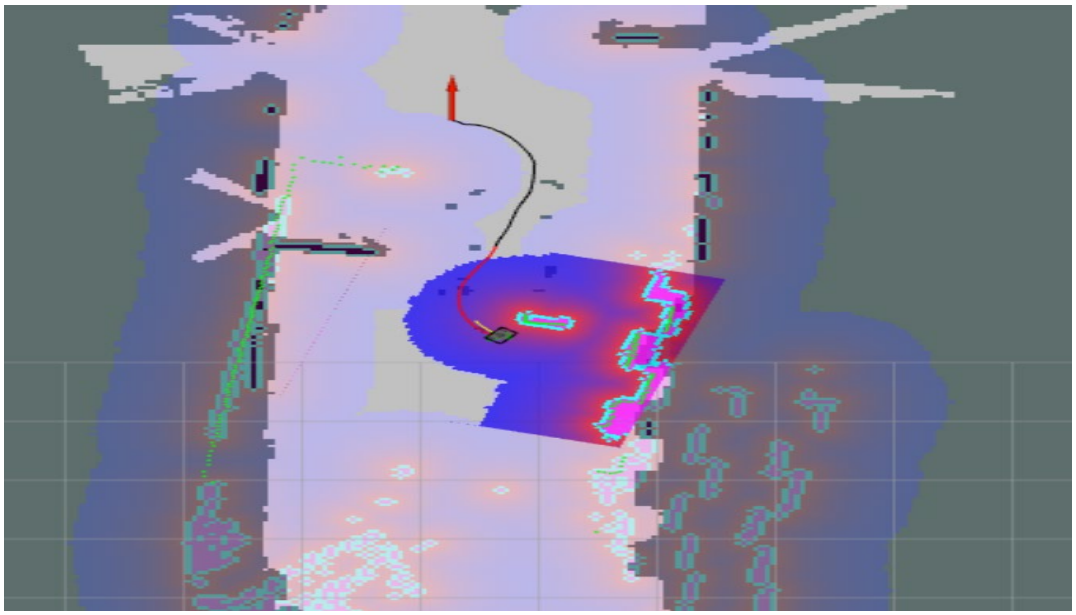

图 5.9 路径规划

如图 5.9 所示,机器人在运动至箱子附近时激光雷达扫描到机器人前方的障碍并规划局部路径（如黄线所示）绕过障碍向目标点移动。

实验结果表明，机器人检测到地图上原本不存在的障碍物后，成功规划局部路径，避开障碍物，成功运动到指定位置。

第六章 视觉 SLAM 初探

目前常见的机器人 SLAM 系统一般具有两种形式：基于激光雷达的 SLAM(激光 SLAM)和基于视觉的 SLAM(Visual SLAM 或 VSLAM)。激光 SLAM 是本次比赛的主要方向，但为了更好的学习 SLAM 技术，我们也对视觉 SLAM 进行了了解并且尝试在车辆上增加视觉 SLAM 传感器已获得更好的行驶策略。 首先我们先了解了相对于激光 SLAM，视觉 SLAM 有什么优劣。经过资料查询与整理，我们汇总得到下表。

表格 6.1 优劣势汇总

优劣	激光 SLAM	视觉 SLAM
优势	可靠性高，技术成熟	结构简单，安装方式多元化
	建图直观，精度高，不存在累计误差	无传感器探测距离限制，成本低
	地图可用于路径规划	可提取语义信息
劣势	受 Lidar 探测范围限制	环境光影响大、暗处（无纹理区域）无法工作
	安装有结构要求	运算负荷大，构建的地图本身难以直接用于路径规划和导航
	地图缺乏语义信息	传感器动态性能还需提高，地图构建时会存在累积误差
	信息更新速率无法满足高速运动平台	噪声较大，需通过大量算法弥补

相比于激光雷达，作为视觉 SLAM 传感器的相机更加便宜、轻便，而且随处可得（如人人都用的手机上都配有摄像头），另外图像能提供更加丰富的信息，特征区分度更高，缺点是图像信息的实时处理需要很高的计算能力。幸运的是随着计算硬件的能力提升，在小型 PC 和嵌入式设备乃至移动设备上运行实时的视

觉 SLAM 已经成为了可能。本次比赛使用的 MINIPC 配备有 Intel Core i3-7130U 处理器，其性能足够处理部分视觉 SLAM 算法。

但是回归到本次比赛的环境，提供地图，只需定位于避障，要求速度。激光雷达可以提供很精细的地图信息，配合 IMU 可以很好的进行定位，同时由于围布图案单一，较难利用视觉进行定位，因此定位任务可以有激光雷达配合 IMU 进行。但同时，由于需要在较高速度下对障碍进行避让，此时激光雷达的信息更新速率限制成为了一个明显的瓶颈，此时视觉 SLAM 传感器的优势尽显无疑。因此我们计划将激光 SLAM 与视觉 SLAM 配合使用。

6.1 视觉传感器

视觉 SLAM 使用的传感器目前主要有单目相机、双目相机、RGBD 相机三种，其中 RGBD 相机的深度信息有通过结构光原理计算的（如 Kinect1 代），也有通过投射红外 pattern 并利用双目红外相机来计算的（如 Intel RealSense R200），也有通过 TOF 相机实现的（如 Kinect2 代）。

结合我们对视觉传感器的任务要求与硬件平台，我们的需求是可以在户外使用，动态性能好，无需大量计算量，最好能直接提供深度数据。经过挑选，我们最终选择了 Intel RealSense D435i 深度相机。

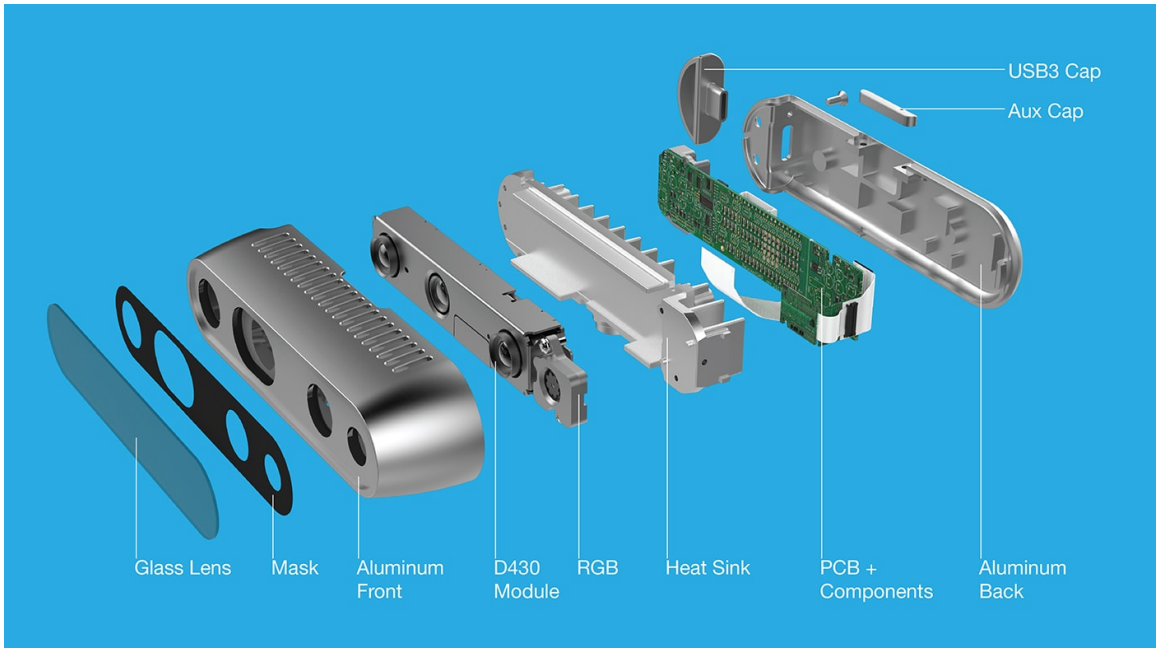

图 6.1 Intel RealSense D435i 深度相机

6.2 镜头标定

我们使用的 D435i 相机使用的结构光+双目红外镜头实现的深度数据采集，因此在到手后需要进行镜头标定才能获得准确的深度数据。

在没有标定前，可以看到本来反射率良好的平面呈现蜂窝状，深度数据几乎不可信。 Intel 官方提供有标定板与标定软件。在进行镜头标定后，可以看到，蜂窝状现象大幅度减少，深度数据基本比较准确。

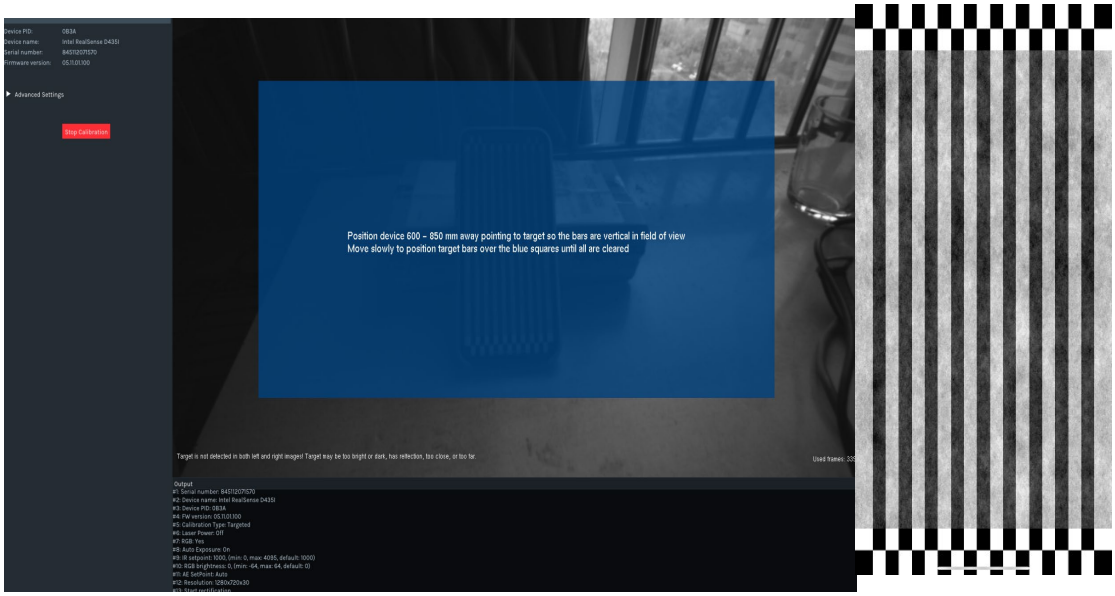

图 6.2 标定板与标定软件

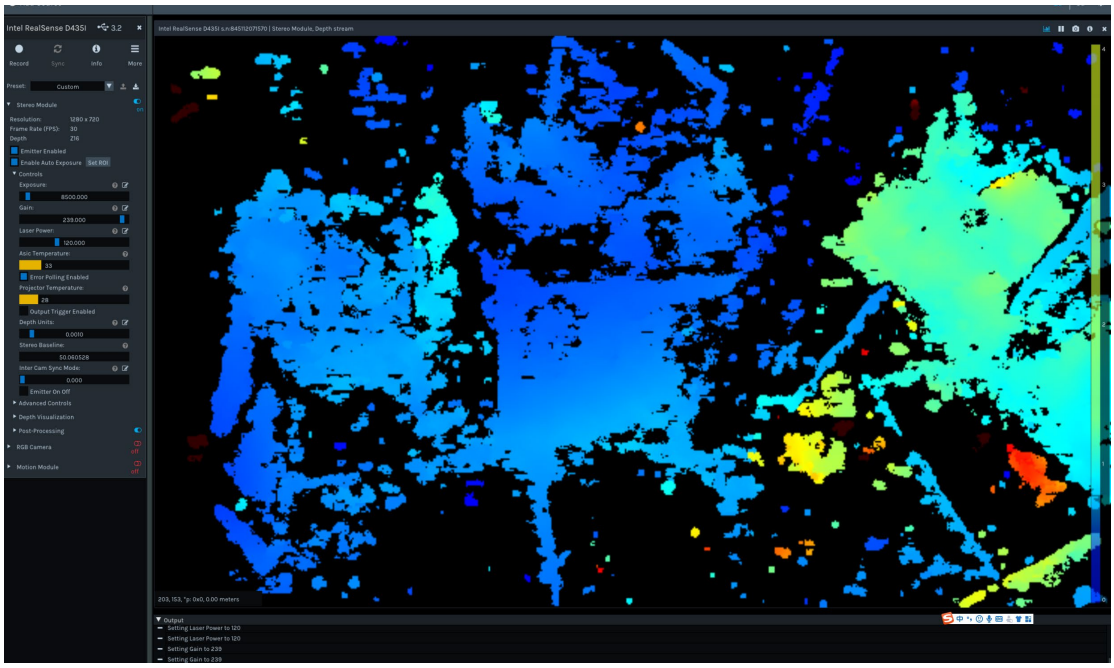

图 6.3 标度前深度图像

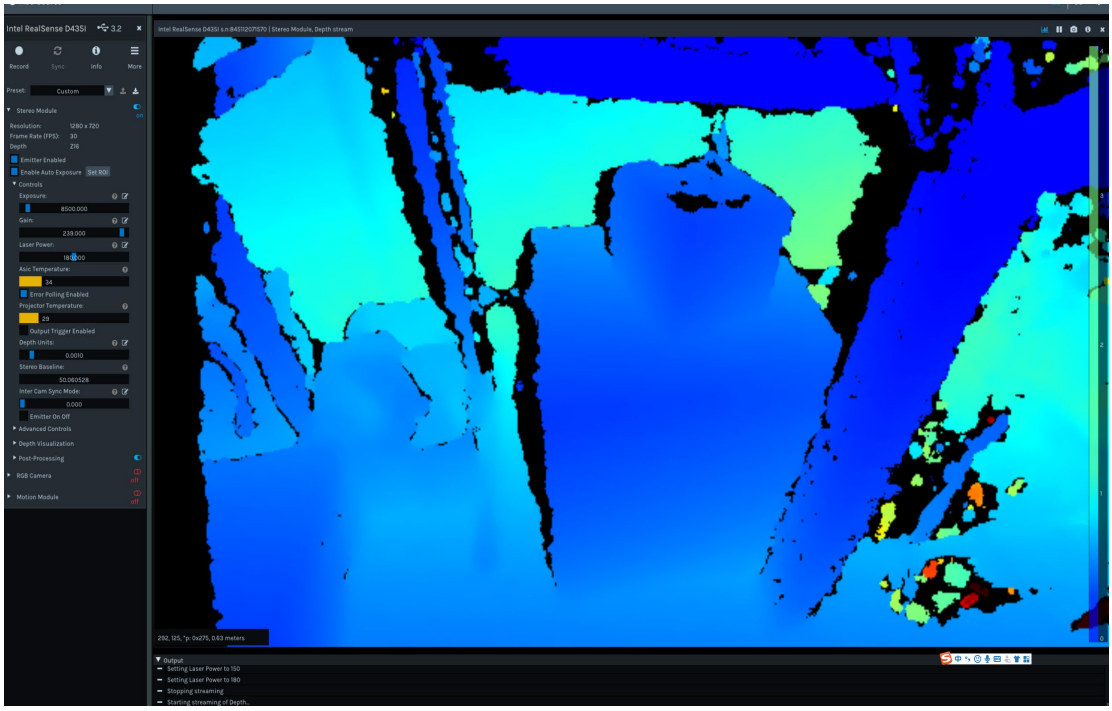

图 6.4 标定后深度图像

6.3 信息处理

根据我们的需求，我们不需要利用 D435i 相机进行建图与导航，因此无需将相机的数据流接入 ROS 系统中进行处理，所以我们选择利用 OpenCV 来获取视觉数据并进行处理。

D435i 相机为 RGB-D 相机，可以输出深度图像与可见光段 RGB 图像两个数据流。但由于深度图像传感器与 RGB 图像传感器并不在同一点，因此深度图像与 RGB 图像的空间坐标系并不重合。我们可以根据两个传感器之间的空间距离经换算得到深度图像与 RGB 图像空间坐标系的转换矩阵，但若使用此方法，需要知道两个相机的镜头内参与空间距离，计算异常繁琐。幸好，Intel 提供的 SDK 中已经可以根据上述的镜头标定数据自动换算出坐标系转换矩阵并利用此矩阵将深度图像与 RGB 图像对齐至同一坐标系，为接下来的处理提供方便。

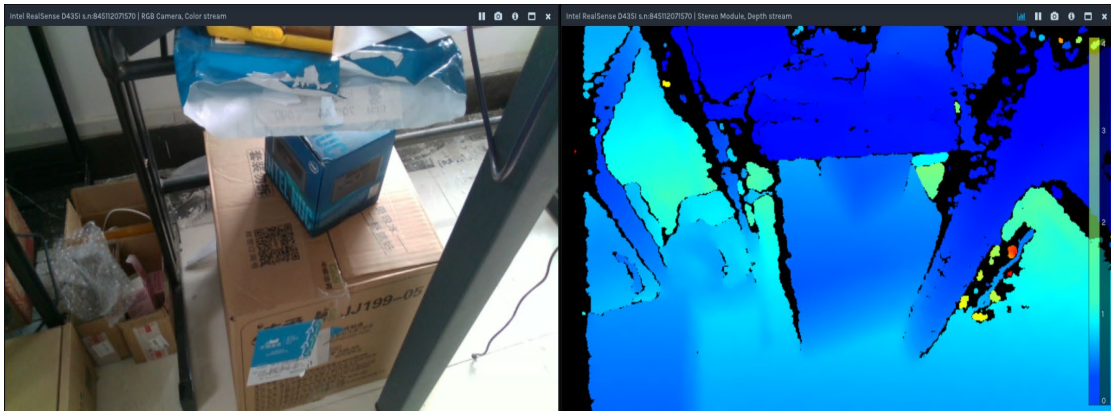

图 6.5 原始 RGB 图像与深度图像

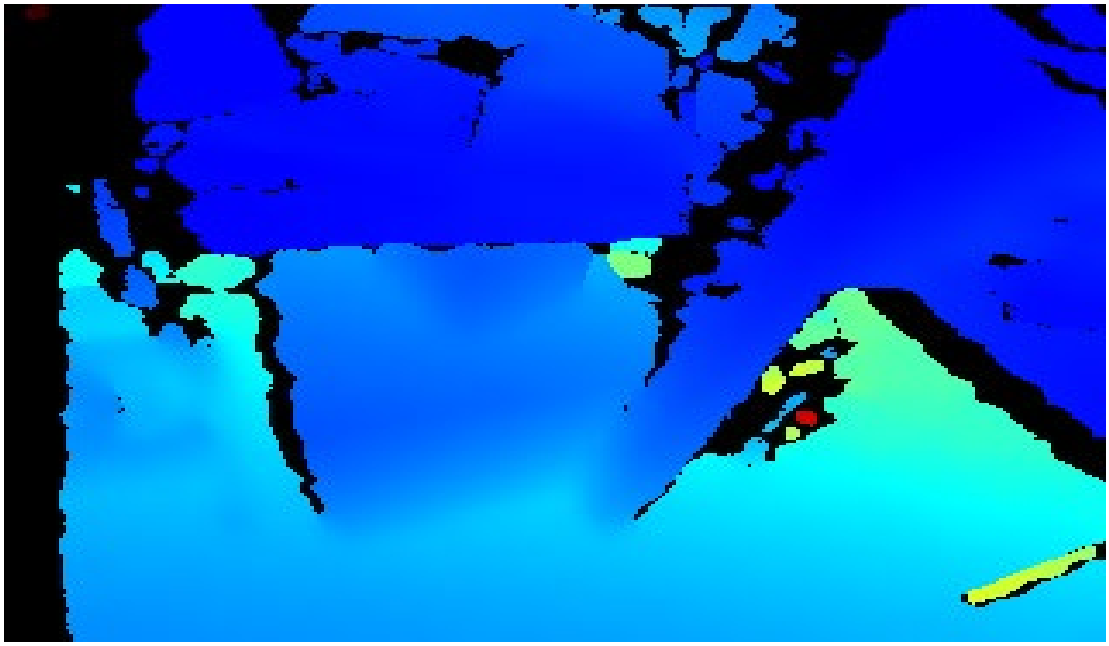

图 6.6 坐标对齐后图像

根据比赛要求，障碍为锥桶。在可见光段，锥桶有很强的颜色特征与形态特征，因此我们可以利用 RGB 图像对锥桶进行寻找。在寻找到锥桶后，可以利用 RGB 图像中锥桶的坐标映射到深度图像上从而获得此锥桶的深度数据。这样，我们就完成了锥桶的探测与定位。

图 6.7 视觉识别效果

6.4 改进方向

由于时间有限，我们并没有对视觉 SLAM 的数据与激光 SLAM 的数据进行融合。目前激光 SLAM 仍占主导地位，负责车体的控制。视觉 SLAM 目前只是利用于进行车体的紧急避障，在检测到障碍距离过近时将直接接手车体控制，对车体实施紧急避障。此方法增加了我们比赛的完成率，可以以更快的速度的避开更多障碍。但此方案只是视觉与激光的简单切换，并没有将两种传感器的信息做到最大的利用。由本章开始的表格可以看出，激光 SLAM 与视觉 SLAM 的优缺点几乎完美互补，因此将两者数据进行叠加，可以获得最大的优化。在比赛前，我们将会在此方面进行更深入的研究。

第七章 总结

本文从机械、硬件、软件等方面详细介绍了车模的设计和制作方案。从机械和硬件方面，我们力求电路的稳定可靠，尽量将车模的机械结构调整到最佳；在软件方面，我们努力追求最有效的行驶路径，不断优化各方面策略，添加 RGBD 相机的避障策略与冗余 IMU 是我们的创新点。同时，考虑到车模无法准确测量速度从而影响定位，我们利用 3D 打印制作了第五轮测速支架，尽可能提高车模运行的稳定性。在车模制作过程中，我们学到很多知识，逐渐对单片机控制、电路设计、机械设计、3D 打印、激光 SLAM 以及视觉 SLAM 和 RGBD 相机视觉处理等有了比较深入的了解与认识。

很感谢实验室能够给我们这样的平台，感谢各位指导老师和学长学姐的指导。经过半年的辛苦准备，我们也积累了很多经验，深刻认识到坚持不懈的重要性。从大雪飘飘到炎炎夏日，从一开始的不断撞击障碍到现在的流畅运行，考验的不是谁更聪明，而是谁可以坚持努力，不断地发现问题解决问题，坚持为比赛付出时间和精力。

最后，再次感谢湖北工业大学蓝电中心智能车实验室这个大家庭，感谢一直支持和关注智能车比赛的学校和学院领导以及各位老师，感谢钢铁侠公司的车模赞助与技术支持，也感谢比赛组委会能组织这样一项很有意义的比赛。

[参考文献]

[1]全国智能车竞赛室外光电组机器人操作系统 ROS 原理与应用
[2]合肥工业大学第十三届创意组 ApolloGo 队设计报告
[3]高翔，张涛，刘毅著.视觉 SLAM 十四讲：从理论到实践.北京：电子工业出版社，2017
[4]Fox D, Burgard W, Thrun S. The Dynamic Window Approach to Collision Avoidance[J]. IEEE Robotics & Automation Magazine, 1997, 4(1):23-33.
[5]曲丽萍.移动机器人同步定位与地图构建关键技术的研究[D].哈尔滨工程大学,2013.
[6]苑全德.基于视觉的多机器人协作 SLAM 研究[D].哈尔滨工业大学,2016.
[7]陈白帆.动态环境下移动机器人同时定位与建图研究［D］．中南大学，2009.
[8]张文玲.移动机器人同时定位与地图创建自适应算法研究［D］．中国科学技术大学， 2009.
[9]张长勇,王兴财,步亚,等.移动机器人搬运物料目标定位优化仿真[J].计算机仿真， 2018, 35(02): 257-261.
[10]孙博雅.移动机器人 SLAM 技术[J].电子技术与软件工程, 2018(2): 95-95.
[11]宋楚轩.室内移动机器人的定位导航技术[J].中国新通信, 2018, 20(02): 73.

附录 A 源代码

```
void RosFilter<T>::run()
  {
    ROS_INFO("Waiting for valid clock time...");
    ros::Time::waitForValid();
    ROS_INFO("Valid clock time received. Starting node.");

    loadParams();

    if (printDiagnostics_)
    {
      diagnosticUpdater_.add("Filter diagnostic updater", this,
&RosFilter<T>::aggregateDiagnostics);
    }

    // Set up the frequency diagnostic
    double minFrequency = frequency_ - 2;
    double maxFrequency = frequency_ + 2;
    diagnostic_updater::HeaderlessTopicDiagnostic
freqDiag("odometry/filtered",

diagnosticUpdater_,

diagnostic_updater::FrequencyStatusParam(&minFrequency,

&maxFrequency,

0.1, 10));

    // We may need to broadcast a different transform than
    // the one we've already calculated.
    tf2::Transform mapOdomTrans;
    tf2::Transform odomBaseLinkTrans;
    geometry_msgs::TransformStamped mapOdomTransMsg;
    ros::Time curTime;
    ros::Time lastDiagTime = ros::Time::now();

    // Clear out the transforms
    worldBaseLinkTransMsg_.transform =
tf2::toMsg(tf2::Transform::getIdentity());
```

```
    mapOdomTransMsg.transform =
tf2::toMsg(tf2::Transform::getIdentity());

    // Publisher
    ros::Publisher positionPub =
nh_.advertise<nav_msgs::Odometry>("odometry/filtered", 20);
    tf2_ros::TransformBroadcaster worldTransformBroadcaster;

    // Optional acceleration publisher
    ros::Publisher accelPub;
    if (publishAcceleration_)
    {
      accelPub =
nh_.advertise<geometry_msgs::AccelWithCovarianceStamped>("accel/filte
red", 20);
    }

    const ros::Duration loop_cycle_time(1.0 / frequency_);
    ros::Time loop_end_time = ros::Time::now();

    // Wait for the filter to be enabled
    while (!enabled_ && ros::ok())
    {
      ROS_WARN_STREAM_ONCE("[" << ros::this_node::getName() << ":] This
filter is disabled. To enable it call the service " <<
ros::this_node::getName() << "/enable");
      ros::spinOnce();
      if (enabled_)
      {
        break;
      }
    }

    while (ros::ok())
    {
      // The spin will call all the available callbacks and enqueue
      // their received measurements
      ros::spinOnce();
      curTime = ros::Time::now();

      // Now we'll integrate any measurements we've received
      integrateMeasurements(curTime);
```

```
      // Get latest state and publish it
      nav_msgs::Odometry filteredPosition;

      if (getFilteredOdometryMessage(filteredPosition))
      {
        worldBaseLinkTransMsg_.header.stamp =
filteredPosition.header.stamp + tfTimeOffset_;
        worldBaseLinkTransMsg_.header.frame_id =
filteredPosition.header.frame_id;
        worldBaseLinkTransMsg_.child_frame_id =
filteredPosition.child_frame_id;

        worldBaseLinkTransMsg_.transform.translation.x =
filteredPosition.pose.pose.position.x;
        worldBaseLinkTransMsg_.transform.translation.y =
filteredPosition.pose.pose.position.y;
        worldBaseLinkTransMsg_.transform.translation.z =
filteredPosition.pose.pose.position.z;
        worldBaseLinkTransMsg_.transform.rotation =
filteredPosition.pose.pose.orientation;

        // the filteredPosition is the message containing the state and
covariances: nav_msgs Odometry

        if (!validateFilterOutput(filteredPosition))
        {
          ROS_ERROR_STREAM("Critical Error, NaNs were detected in the
output state of the filter." <<
                " This was likely due to poorly coniditioned process,
noise, or sensor covariances.");
        }

        // If the worldFrameId_ is the odomFrameId_ frame, then we can
just send the transform. If the
        // worldFrameId_ is the mapFrameId_ frame, we'll have some work
to do.
        if (publishTransform_)
        {
          if (filteredPosition.header.frame_id == odomFrameId_)
          {
```

```
worldTransformBroadcaster.sendTransform(worldBaseLinkTransMsg_);
          }
          else if (filteredPosition.header.frame_id == mapFrameId_)
          {
            try
            {
              tf2::Transform worldBaseLinkTrans;
              tf2::fromMsg(worldBaseLinkTransMsg_.transform,
worldBaseLinkTrans);

              if (!RosFilterUtilities::lookupTransformSafe(
                    tfBuffer_,
                    baseLinkFrameId_,
                    odomFrameId_,
                    ros::Time(filter_.getLastMeasurementTime()),
                    odomBaseLinkTrans,
                    true))
              {
                ROS_ERROR_STREAM_DELAYED_THROTTLE(1.0, "Unable to
retrieve " << odomFrameId_ << "->" <<
                   baseLinkFrameId_ << " transform. Skipping
iteration...");
                continue;
              }

              /*
               * First, see these two references:
               *
http://wiki.ros.org/tf/Overview/Using%20Published%20Transforms#lookup
Transform
               *
http://wiki.ros.org/geometry/CoordinateFrameConventions#Transform_Dir
ection
               * We have a transform from mapFrameId_->baseLinkFrameId_,
but it would actually transform
               * a given pose from baseLinkFrameId_->mapFrameId_. We
then used lookupTransform, whose
               * first two arguments are target frame and source frame,
to get a transform from
               * baseLinkFrameId_->odomFrameId_. However, this
transform would actually transform data
```

```
               * from odomFrameId_->baseLinkFrameId_. Now imagine that
we have a position in the
               * mapFrameId_ frame. First, we multiply it by the inverse
of the
               * mapFrameId_->baseLinkFrameId, which will transform
that data from mapFrameId_ to
               * baseLinkFrameId_. Now we want to go from
baseLinkFrameId_->odomFrameId_, but the
               * transform we have takes data from
odomFrameId_->baseLinkFrameId_, so we need its
               * inverse as well. We have now transformed our data from
mapFrameId_ to odomFrameId_.
               *However, if we want other users to be able to do the same,
we need to broadcast
               * the inverse of that entire transform.
              */

              mapOdomTrans.mult(worldBaseLinkTrans,
odomBaseLinkTrans);

              mapOdomTransMsg.transform = tf2::toMsg(mapOdomTrans);
              mapOdomTransMsg.header.stamp =
filteredPosition.header.stamp + tfTimeOffset_;
              mapOdomTransMsg.header.frame_id = mapFrameId_;
              mapOdomTransMsg.child_frame_id = odomFrameId_;


worldTransformBroadcaster.sendTransform(mapOdomTransMsg);
            }
            catch(...)
            {
              ROS_ERROR_STREAM_DELAYED_THROTTLE(5.0, "Could not obtain
transform from "
                                                << odomFrameId_ <<
"->" << baseLinkFrameId_);
            }
          }
          else
          {
            ROS_ERROR_STREAM("Odometry message frame_id was " <<
filteredPosition.header.frame_id <<
```

```
                              ", expected " << mapFrameId_ << " or " <<
odomFrameId_);
          }
        }

        // Fire off the position and the transform
        positionPub.publish(filteredPosition);

        if (printDiagnostics_)
        {
          freqDiag.tick();
        }
      }

      // Publish the acceleration if desired and filter is initialized
      geometry_msgs::AccelWithCovarianceStamped filteredAcceleration;
      if (publishAcceleration_ &&
getFilteredAccelMessage(filteredAcceleration))
      {
        accelPub.publish(filteredAcceleration);
      }

      /* Diagnostics can behave strangely when playing back from bag
       * files and using simulated time, so we have to check for
       * time suddenly moving backwards as well as the standard
       * timeout criterion before publishing. */
      double diagDuration = (curTime - lastDiagTime).toSec();
      if (printDiagnostics_ && (diagDuration >=
diagnosticUpdater_.getPeriod() || diagDuration < 0.0))
      {
        diagnosticUpdater_.force_update();
        lastDiagTime = curTime;
      }

      // Clear out expired history data
      if (smoothLaggedData_)
      {
        clearExpiredHistory(filter_.getLastMeasurementTime() -
historyLength_);
      }
```

```
      ros::Duration loop_elapsed_time = ros::Time::now() -
loop_end_time;

      if (loop_elapsed_time > loop_cycle_time)
      {
        ROS_WARN_STREAM_DELAYED_THROTTLE(1.0, "Failed to meet update
rate! Took " << std::setprecision(20) <<
          loop_elapsed_time.toSec() << " seconds. Try decreasing the
rate, limiting sensor output frequency, or "
          "limiting the number of sensors.");
      }
      else
      {
        ros::Duration sleep_time = loop_cycle_time - loop_elapsed_time;
        sleep_time.sleep();
      }

      loop_end_time = ros::Time::now();
    }
  }
```