全国大学生智能汽车竞赛

航天-智慧物流

**技**

**术**

**报**

**告**

参赛学校 华南农业大学

队伍名称 Taurus

指导老师 邓海东，龙拥兵

参赛学生 申国豪，邱旭林，曾扬晨，黄增航，肖美棋

二零二二年

[第1章 项目背景及意义 4](#_Toc109943367)

[第2章 项目方案设计 4](#_Toc109943368)

[2.1 激光SLAM导航实现 5](#_Toc109943369)

[2.1.1 定位 5](#_Toc109943370)

[2.1.2 建图 5](#_Toc109943371)

[2.1.3 全局规划 5](#_Toc109943372)

[2.1.4 局部规划 5](#_Toc109943373)

[2.2 车道线检测与识别交通灯 6](#_Toc109943374)

[2.3 中央调度系统开发 6](#_Toc109943375)

[2.4 阿克曼运动控制 6](#_Toc109943376)

[第3章 激光SLAM导航实现 6](#_Toc109943377)

[3.1 定位 6](#_Toc109943378)

[3.2 建图 6](#_Toc109943379)

[3.3 全局规划 9](#_Toc109943380)

[3.4 局部规划 9](#_Toc109943381)

[3.4.1 TEB局部路径规划算法 9](#_Toc109943382)

[3.4.2 纯追踪算法原理 11](#_Toc109943383)

[3.4.2.1 纯追踪算法原理 11](#_Toc109943384)

[3.4.2.2 曲率调整前视距离 13](#_Toc109943385)

[3.4.2.3 对局部目标点卡尔曼滤波 13](#_Toc109943386)

[3.4.3 算法调试与实现效果 14](#_Toc109943387)

[第4章 车道线检测与识别交通灯 15](#_Toc109943388)

[4.1 车道线检测及跟随 15](#_Toc109943389)

[4.1.1 车道检测-检测器-pspnet 15](#_Toc109943390)

[4.1.2 网络骨干替换-MobileNetV2 15](#_Toc109943391)

[4.1.3 训练Trick 16](#_Toc109943392)

[4.1.4 最小二乘法拟合多点追踪模型 16](#_Toc109943393)

[4.1.5 求解质心单点追踪模型 17](#_Toc109943394)

[4.1.6 运动控制 17](#_Toc109943395)

[4.1.7 可视化界面调试 18](#_Toc109943396)

[4.2 识别交通灯 19](#_Toc109943397)

[第5章 中央调度系统开发 19](#_Toc109943398)

[5.1 socket通信 20](#_Toc109943399)

[5.1.1 通信模块封装 20](#_Toc109943400)

[5.1.2 多线程同步发送与接受消息 20](#_Toc109943401)

[5.1.3 自定义通信规范 21](#_Toc109943402)

[5.1.4 视频流传输 21](#_Toc109943403)

[5.2 可视化界面 22](#_Toc109943404)

[5.2.1 界面结构介绍 22](#_Toc109943405)

[5.2.2 各功能介绍 23](#_Toc109943406)

[第6章 阿克曼运动控制 24](#_Toc109943407)

[6.1 概述 24](#_Toc109943408)

[6.2 实现方案设计分析 26](#_Toc109943409)

[6.2.1 Odom数据底层处理 26](#_Toc109943410)

[6.2.2 Odom数据ROS层处理 26](#_Toc109943411)

[6.2.3 前轮转向标定 26](#_Toc109943412)

[第7章 总结 27](#_Toc109943413)

# 项目背景及意义

近年来，人工智能这一科技浪潮正在深刻改变着世界，智能机器人作为人工智能的一个综合性载体已经开始渗透进日常生活的方方面面。随着我国国民经济的高速发展和先进自动控制设备的广泛采用，以及人力成本的日益增加，对自动化搬运环节的需求也越来越大，但是目前国内智慧物流方面的技术人才比较匮乏。

本次大赛以工业智慧物流为主题，以无人驾驶技术在工业上的应用为基础，围绕人工智能领域典型传感器的使用、操作、编程、调试等内容，考察选手对当下热门的运动控制、无人驾驶算法、视觉识别算法的应用能力，强化选手对智能感知技术在工业中应用的综合技能，为在工业领域推广应用人工智能技术打下良好的基础。

# 项目方案设计

项目中要求我们使用轻舟机器人实现避障、定点运输货物、卸货、识别红绿灯和车道线检测通过S弯。小车上有车体、上位机Nano处理器、单目摄像头、激光雷达、核心板stm32 f103、IMU、直流减速电机、电源等。

## 激光SLAM导航实现

### 定位

在小车定位方面，由于单纯使用里程计作为小车定位，误差会随着累积。所以我们采用AMCL功能包实现对小车位于全局地图的位姿定位，功能包需要时间逐渐订阅由激光雷达发布到/scan与由小车差速编码器解算得出的小车里程计话题/odom，并最终输出小车里程计坐标系odom与map坐标系的tf变换。

### 建图

在构建地图方面，我们使用catrographer功能包进行仿真地图构建，功能包订阅由传感器节点发布的/scan与/odom话题。通过teleop控制节点，手动控制小车在仿真地图内行进，最终由得到由目标环境的二维栅格地图。为了解决传感器误差对地图精确度的影响，我们使用了ps工具对地图进行修改，增加了地图分辨率，使地图更加精确。

### 全局规划

在全局规划方面，我们使用 move\_base 导航包，提供 A\*算法，利用 map\_server所提供的由小车通过激光雷达所创建出来的地图和小车当前自身的位置以及目标的位置等数据来规划出一条最优的路径。

### 局部规划

在局部路径规划算法方面，我们原先采用TEB局部路径规划算法，但是由于其参数众多，在经过数十天的调参，仍不能使小车规划出一条较为可行及较好跟随的局部路径，小车经常会在某个位置不停地调整自身姿态，因此我们及时弃用了TEB，转而选择了纯追踪算法。

## 车道线检测与识别交通灯

车道线检测上我们使用了轻量级语义分割的神经网络pspnet，并用质心公式提取出车道线中心点，然后通过比例解算得到角度，再进行阿克曼运动控制，为了方便调试我们还设计了一个可动态调参的ui方便可视化。

交通灯识别上我们使用了传统计算机视觉，再加上aruco码进行图像裁剪，减少误识别，最后使用了中值滤波提高了鲁棒性。

## 中央调度系统开发

中央调度系统使用socket+pyqt5+Qtdesigner工具开发，技术实现主要包括socket通信开发以及ui可视化界面开发。服务端在Linux系统中运行，通过TCP/IP协议与客户端交互，同时在视频流上，对图像进行了裁剪和压缩，再使用UDP/IP进行通信。

## 阿克曼运动控制

相较于四轮差速底盘，阿克曼模型通过前轮的机械转向，让底盘中的四个轮子在基本不发生侧向滑动的情况下能够顺畅地转弯。这都有益于解决底盘控制稳定性、里程计解算、轨迹跟踪等问题；stm32通过DMA外设与ROS进行通讯，减少了对于stm32资源的占用，为机器人的定位导航提供稳定、可靠的传感器信息。

# 激光SLAM导航实现

## 定位

定位主要依靠轮式里程计和imu进行积分，由于阿克曼模型小车里程计主要依靠后轮编码器，而前轮转角对角速度影响非常大，导致角速度测量不准，我们使用imu进行角速度的测量，里程计进行线速度测量。同时无论是里程计和imu都存在积分误差，需要使用雷达进行误差修正，在常见的开源方案中主要使用AMCL进行定位修正，但是由于AMCL粒子收敛慢，效果不好，我们转而使用了ICP点云匹配算法。

### AMCL自适应蒙特卡洛定位算法

在前期机器人定位上，我们使用了AMCL(adaptive Monte Carlo Localization)自适应蒙特卡洛定位包[3]。其原理是在地图中布置粒子，后获取机器人的运动来移动粒子，并使用每个粒子所处位置模拟一个传感器信息跟观察到的传感器信息作对比，从而赋给每个粒子一个概率。之后根据生成的概率来重新生成粒子，概率越高的生成的概率越大。在迭代有限次之后，所有的粒子会收敛到一起，最终机器人的确切位置被推算出来。

优点：不受场景的限制，算法简捷快速，同时也可以兼顾算法的精度问题。鲁棒性高。有开源代码，实现方便。

缺点：粒子收敛的速度慢，定位效果不好。在分区赛中我们使用的就是AMCL算法，当小车速度比较快时，定位不佳导致小车在直道也会摇晃，浪费时间的同时比较危险。

### ICP迭代最近点定位算法

为了改善定位效果，我们决定放弃AMCL算法，使用scan-to-map的匹配算法，将激光雷达扫描数据直接与地图进行匹配，直接获得小车的位姿进行定位更新，最终获得较好的效果。

在匹配算法上，我们使用迭代最近点（Iterative Closest Point, 下简称ICP）算法，是最为经典的数据配准算法。通过求取源点云和目标点云之间的对应点对，基于对应点对构造旋转平移矩阵，并利用所求矩阵，将源点云变换到目标点云的坐标系下，估计变换后源点云与目标点云的误差函数，若误差函数值大于阀值，则迭代进行上述运算直到满足给定的误差要求。最终获得源点云变换到目标点云的旋转平移矩阵。

#### 估计对应点（Correspondences estimation）

将激光雷达作为源点云，将二维栅格地图的每个像素点转换为一个点得到目标点云，为了加速计算，可以不需要计算 Target 点云中每个点到 Source 点云中一点的距离。可以设定一个阈值，当距离小于阈值时，就将其作为对应点。

#### 迭代优化

找到对应点后，我们就构建了两组对应的点集:

求欧式变换R,t使得：

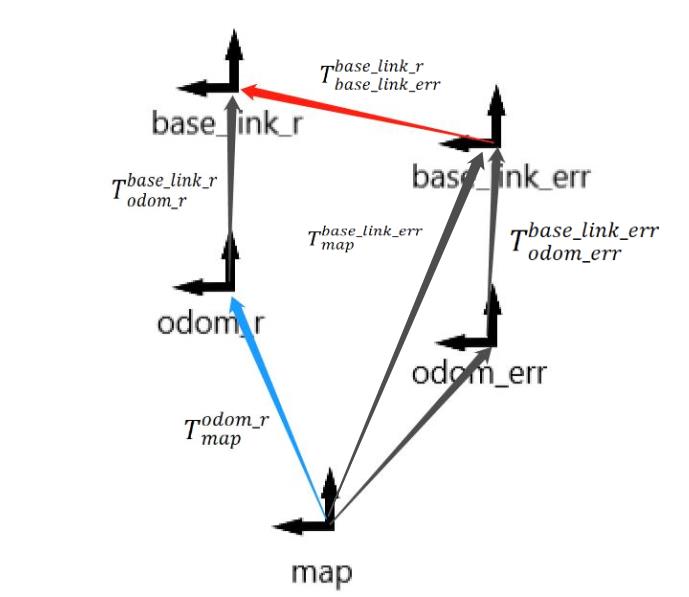
由于噪声和错误匹配,P和Q并不能对应到空间中的同一点，会存在一定的误差，同时匹配算法必须得到一个确定的解，所以我们要最小化的目标函数为：

求解R和t的方法主要有SVD和非线性优化，由于时间有限，本项目直接使用PCL库中的ICP实现，PCL中的ICP算法是基于SVD（Singular Value Decomposition）实现的，可以得到源点云到目标点云的坐标变换。

#### 坐标变换与定位实现

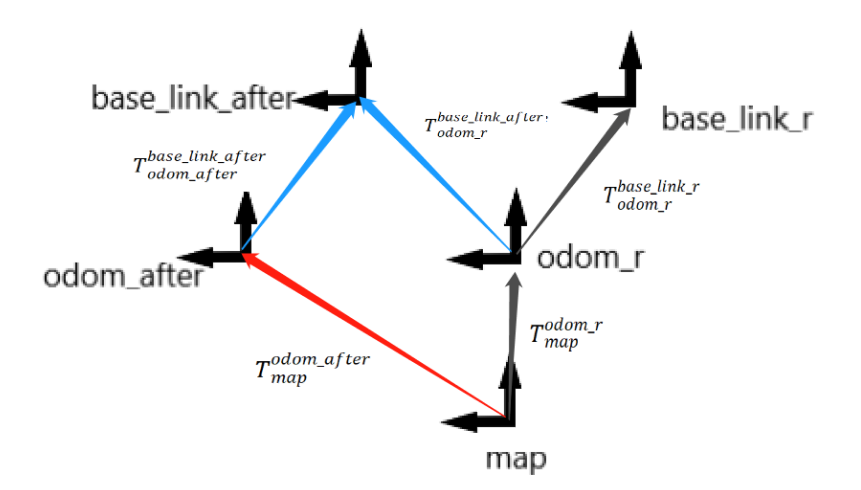
用前面的方法就能得到当前小车坐标点到小车真实坐标的变换矩阵，要实现修正小车定位，我们沿用了AMCL中发布map到odom坐标变换的方式。

为了方便表示变换，下文用欧式变换来表示坐标变换，，其中R为旋转矩阵，t为平移矩阵，并定义表示b到a的坐标变换。为了区分，我们将变换前的odom和base\_link坐标命名为odom\_err和base\_link\_err，变换后的命名为odom\_r和base\_link\_r，map坐标系为全局坐标系保持不变，其中为ICP匹配得到的坐标变换，需要求解map到Odom的变换，即。坐标关系如下图所示。



化简得：

即为修正定位需要发布的变换，但是由于ICP匹配需要进行多次迭代，定位效果存在延迟（实测ICP平均计算时间为0.1s），并且匹配的频率较低（雷达频率为10HZ左右），不能满足使用需求，我们需要将这段时间小车运动加入到变换中。



=

化简得：

该方案的定位定位频率理论上可以达到200hz（与imu一致），且非常稳定，定位修正速度快。

## 建图

实现激光SLAM导航的前期工作需要我们首先对使用的工具的测试。接着我们对搭建的场地进行gmapping的建图，发现建出来的图比较扭曲，会对我们后续导航起到很大的影响，于是便将方向转向了使用cartographer算法的建图，按照线上赛的配置进行轻舟机器人实时建图时，建出来的图虽然比gmapping的效果好，可还是会出现部分区域的重影以及位置上面的不准确。为了不影响后续导航的路径平滑以及准确性，我们考虑了两种建图方法并且都做了对应的实现，其一是通过画图工具将相关尺寸画出来，我们通过一定分辨率，将地图上的障碍通过像素点描绘出来，虽然画出来的效果不错，比较清晰，不过与实际位置还是存在偏差，需要后续根据实际情况来进行不断的修修补补，如图 3‑1所示：

图片包含 图示

描述已自动生成

图 3‑1

其二是通过仿真来建图，gazebo上面可以根据坐标点来对每个墙面设置安放的位置，这样也可以大大的提高建图的准确性，并且建好的仿真环境还可为后续导航巡回，实现整套流程等的测试，在仿真上面能够很好的实现，跑通后再用在轻舟机器人上面，这样既可以在仿真上大大提高了功能的准确性，又可以大大降低了轻舟机器人的来回重启调试达到想要的效果的麻烦，减少了机器人的工作量，对硬件上面烧坏的负担与拖延进度做到了很大程度的降低，同时大大提高了我们总体的效率以及各方面进度的协调配合。

在仿真环境的搭建以及测试上面，仿真环境以及车辆模型完成后具体呈现如图 3‑2所示，建立好cartographer的建图相关配置后，开始在仿真下进行地图的构建，使用cartographer地图保存方式将建好到的地图保存为pbstream格式后，再将其转成ROS下使用的栅格地图格式yaml以及pgn格式的地图，如图 3‑3所示：

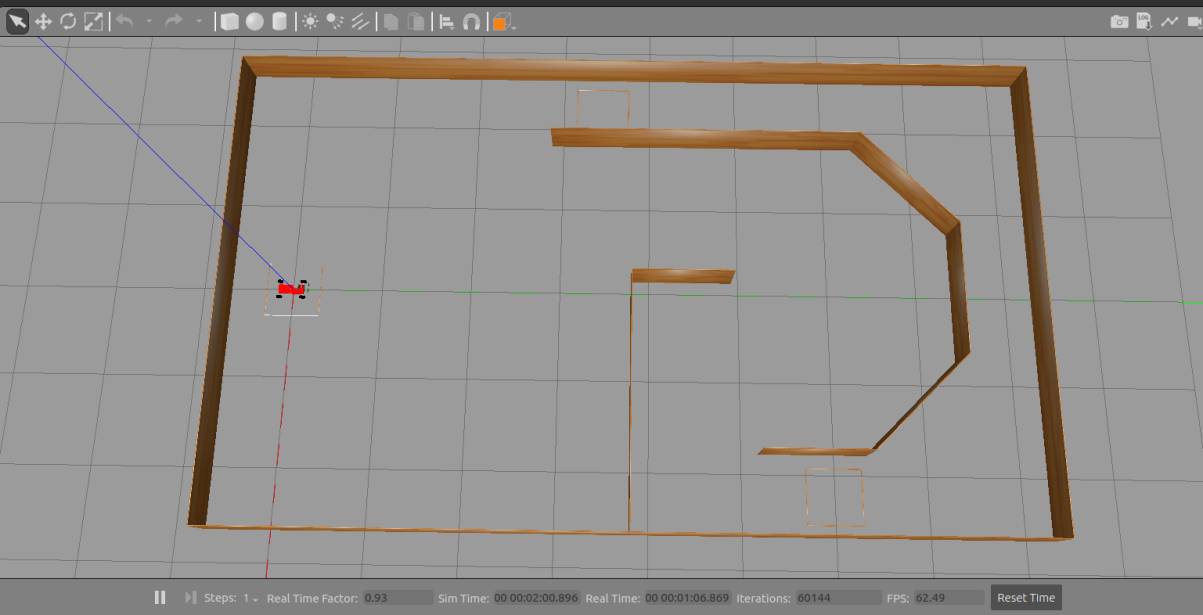


图 3‑2

在仿真环境下使用cartographer建出来的图十分准确，可以作为我们后续导航的主要地图。

## 全局规划

在全局路线规划上，使用全局规划器global Planner所提供的A\*算法求解出最短路径，通过下一节点距离上一点的代价，以及下一结点距离终点的预计代价的总和作为综合代价[4]。通过每次选取综合代价最小点作为下一个等待遍历的节点，在有限次的迭代后最终求解出最有的全局规划路径。

## 局部规划

### TEB局部路径规划算法

TEB,全称Timed-Elastic-Band。关于elastic band（橡皮筋）的定义：连接起始、目标点，并让这个路径可以变形，变形的条件就是将所有约束当做橡皮筋的外力。关于time elastic band的简述：起始点、目标点状态由用户/全局规划器指定，中间插入N个控制橡皮筋形状的控制点（机器人姿态）；为了显示轨迹的运动学信息，我们在点与点之间定义运动时间Time，即为Timed-Elastic-Band算法。

注意：机器人的姿态指的是机器人坐标和朝向，称之为configuration;引入Time的概念，保证机器人运动是实时的，在一定时间内完成。Time Elastic Band算法把路径规划问题描述为一个多目标优化问题，即对最小化轨迹执行时间、与障碍物保持一定距离并遵守运动动力学约束等目标进行优化。因为优化的大多数目标都是局部的，只与机器人的某几个连续的状态有关，所以该优化问题为对稀疏模型的优化。求解稀疏模型多目标优化问题，可通过构建超图（hyper-graph），使用g2o（通用图优化）框架中关于大规模稀疏矩阵的优化算法来求解。机器人状态和时间间隔作为nodes，目标函数和约束函数作为edges，各nodes由edges连接构成hyper-graph。在该hyper-graph中，每个约束为一条edge，且每条edge允许连接的nodes的数目不受限制。Time Elastic Band算法通俗的解释：从给定路径中得到一系列带时间信息的离散位姿（pose），通过图优化的方法将这些离散位姿组成满足时间最短、距离最短和远离障碍物等目标的轨迹，同时满足机器人运动动力学的约束。需要注意的是，优化得到的轨迹并不一定满足所有约束，即给定的约束条件实际上都是软约束条件。

优点：

* 适用于各种常见车模：如差分、全向、阿克曼模型。
* 有很强的前瞻性： 对前方一段轨迹进行优化。
* 动态避障效果好： 对动态障碍有较好的避障效果，可直接使用其封装好障碍类Obstacle。

缺点：

* 计算复杂度较大：可通过牺牲预测距离来降低复杂度。
* 速度和角度波动较大、控制不稳定： 源码中是通过两状态之间的距离和角度差及时间差来计算该控制周期内的速度和角速度，使得在控制过程中速度和角度波动较大。
* 非全局最优： 但是优于DWA。
* 算法通用性高也导致整个算法过于复杂，需要调整的参数过多，很难达到局部最优解。

### 纯追踪算法原理

由于TEB算法有以上的缺点，我们最终决定自己重写轨迹跟踪算法，并达到了不错的效果。我们使用了纯追踪算法，算法原理如下所示。

#### 纯追踪算法原理

对阿卡曼转向几何的简化，将车辆四轮模型简化为两轮模型，并且假定车辆只在平面上行驶，得到个自行车运动学模型，如图 3‑5所示。

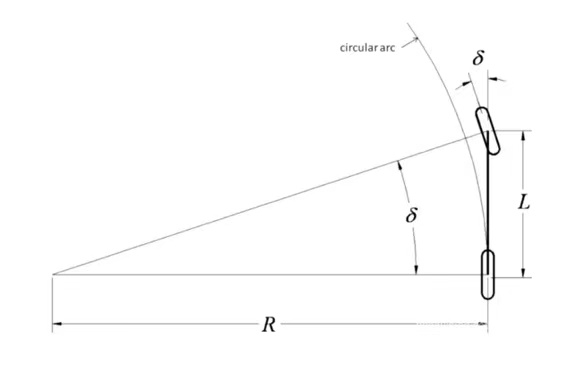


图 3‑4

它是对阿卡曼转向几何的简化，将车辆四轮模型简化为两轮模型，并且假定车辆只在平面上行驶，那么前轮转角、轴距和转向半径将遵循以下关系：

(1)

其中是前轮的转向角，L为轴距，R为当前转角下的转向半径。

从自行车模型出发，纯追踪算法以车后轴为切点、车辆纵向车身为切线，通过控制前轮转向角，使车辆可以沿着一条经过目标路经点的圆弧行驶，如图 3‑6所示：

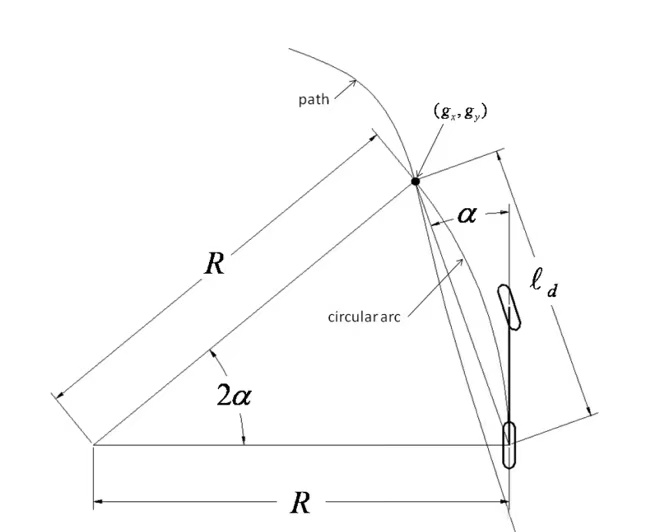


图 3‑5

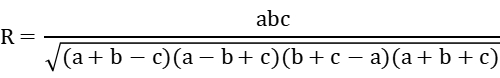
图中(,)是下一个要跟踪的路径点，它位于已经规划好的全局路径中，需要控制车辆使得车辆的后轴经过该路径点，代表车辆当前位置（后轴）到目标路径点的距离，代表当前车身姿态和目标路径点的夹角，根据正弦定理可以推导出以下公式：

(2)

结合(1)(2)两式，可以得到纯追踪算法的控制量的最终表达式:

#### 曲率调整前视距离

同时使用轨迹的曲率调整前视线距离，先得到前方轨迹的三个点，可得到两点间距离a,b,c，再根据公式



可得到三点构成的圆的曲率再，根据曲率变化调整前视距离，得到不错的控制效果。

#### 对局部目标点卡尔曼滤波

在纯跟踪算法中，我们使用固定一定前视距离获取局部目标点，由于路径存在波动，为了使纯跟踪更加平滑，我们对目标点进行卡尔曼滤波。

卡尔曼滤波是非常经典的预测追踪算法，能够在系统存在噪声和干扰的情况下进行系统状态的最优估计，算法的核心思想是,根据当前的仪器"测量值" 和上一刻的 "预测量" 和 "误差",计算得到当前的最优量，再预测下一刻的量,， 里面比较突出的是观点是，把误差纳入计算，而且分为预测误差和测量误差两种通称为噪声，还有一个非常大的特点是，误差独立存在，始终不受测量数据的影响。

纯追踪算法的优点是很大地简化了车辆运动学和动力学模型，输入输出明确，在低速下能获得较好地控制效果；缺点是简化了太多的车辆运动学和动力学模型，所以只要速度和加速度提高，运动学动力学简化条件不再成立，就无法保证横向控制精度。

### 算法调试与实现效果

在调整导航参数时，我们的思路是首先调试好全局路径，再去调局部路径。为了让小车规划的全局路径更加的贴合赛道中间以及平滑，我们首先将全局代价地图上的膨胀半径调大，使赛道两边膨胀区域与边界贴合，经测试，全局膨胀半径为0.5时能达到较好的效果。

在最初的局部路径规划方案中，我们使用TEB作为导航上使用的局部路径规划，轻舟机器人是阿克曼的运动模型，TEB对阿克曼模型具有良好的适用性，并且具有很强的前瞻性和动态避障的效果，同时配置了amcl的参数文件，为导航的定位提供实施方向。可以实现导航功能但是很难调到较好的效果。

由于teb算法有不少缺点，我们最终决定自己重写轨迹跟踪算法，使用了纯追踪算法，为了更好地测试小车能否很好的实现相关功能，我们搭建了仿真环境，通过获得需要使用的相关功能话题，纯跟踪进行了实现。在仿真环境下经过不断的测试，发现小车跟随全局的效果很好，小车达到4m/s速度也能跟随路径行驶，由于轻舟机器人的电机速度有限制，只能到1.2米/秒，我们就打算设置到最大速度进行不断测试，经过多次的测试，发现在这种特定的场景下，纯跟踪比teb更容易达到局部最优解，很高的速度下也能很好的跟随全局的路径很稳定行驶到目标点。

# 车道线检测与识别交通灯

## 车道线检测及跟随

具体实现车道线拟合总共有4个步骤：车道检测，剔除低鲁棒性，算法提取跟踪点，运动控制。

### 车道检测-检测器-pspnet

考虑到车道线跟踪任务要求实时性和鲁棒性，传统视觉和很多较大的网络都无法同时满足两个要求，我们选择了pspnet。

### 网络骨干替换-MobileNetV2

由于实例分割网络通常需要高算力/高精确度，在嵌入式或低算力环境会达不到实时监测的要求，所以轻量化网络是必不可少的工作，其中针对非头部的部分我们可以进行轻量处理。其中重要的部分是骨干网络模型替换，由原本的繁重的VGG网络替换为更加轻量的MobileNet网络是本次S弯车道线检测的重头戏。其中MobileNet的构建又是其中的主要部分。MobileNet采用了1X1的卷积通道上升，而采用深度卷积块进行特征提取，第三步再用1X1的卷积块进行通道下降，这三步构成模块的最基本，能高效的提取S弯的特征进而进行分割处理。最终的权重模块经过轻量化之后模型仅仅9.3MB，相比未轻量化的VGG上百兆的大小，轻量化明显，于此同时准确度依旧维持了高水平，达到了实时检测/实时分割的基本标准。

### 训练Trick

由于网络深度不高，采用合理的针对S弯的训练Trick是非常重要的。我们训练世代为800个epoch，batchsize为16并且采用梯度学习率下降从1e-2每50个epoch下降，直至1e-5。输入分辨率为473X473，下采样倍数选择了测试后更优的16而优化器采用adam，相比于sdg更能解决长世代训练的梯度爆炸以及梯度消失。最终模型成功拟合，val/train loss均达到低点0.01，实时检测帧数达到30FPS以上成功满足实时的实例分割S弯的检测要求。

### 最小二乘法拟合多点追踪模型

针对小车模型我们最先设计的是求出多点进行追踪，这在长线（例如S弯）的循迹中是占优的。我们可以针对相对远的点进行求解，求解出相对中心，与此同时针对临近摄像头底部盲区的车道线，由于其不稳定我们可以在适时时弃用，采用最小二乘法拟合出绿色的追踪线，反而更加有效。

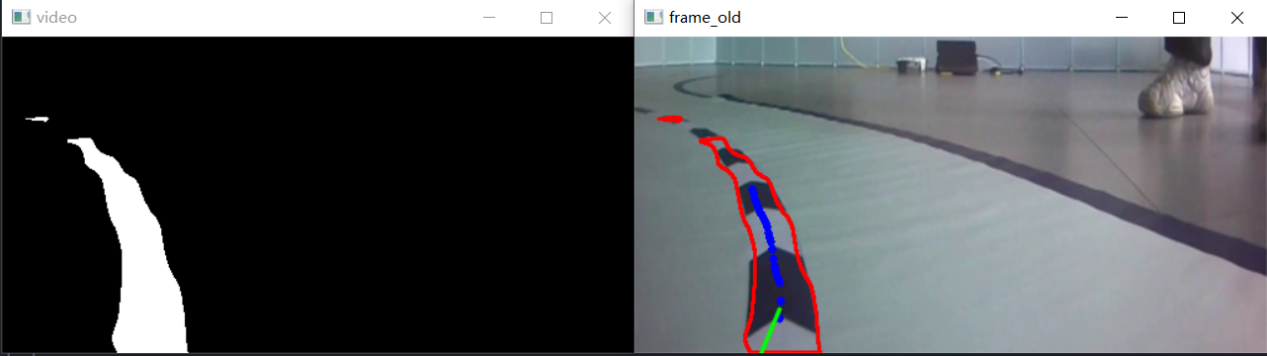


图 4‑1

### 求解质心单点追踪模型

由于长线追踪不稳定，单点模型反而在S弯中测试效果更好。我们可以用网络分割出的赛道的分割实例图，求解质心则可求出车道线中心点。

### 运动控制

得到车道线中心点后，需要控制小车运动跟随车道线运动，我们尝试了两种方案

方案一：pid控制；使用车道线与图像中间像素点的差值直接进行pid控制，优点是实现简单；缺点是需要调整3个参数，并且速度不同参数不同，并且不符合阿克曼小车运动模型。

方案二：解算得到角度进行运动控制；根据车道线与图像中间像素点的差值和偏差角度与摄像头角度成比例的关系，公式为

得到摄像头坐标系下的偏差角度，再根据阿克曼运动模型

(vel\_a为角速度，angle为角度差，vel\_x为x方向线速度，L为车辆轴距)

可计算得到小车角速度，由于得到的角速度为摄像头坐标系下的值，真实控制需要小车坐标系下的值，但是单目摄像头无法算出距离无法进行坐标变换，所以加上一个系数p进行调整。这样就使得要调整的参数由pid的3个降低到一个，且不受速度影响，大大降低调参工作量。最终我们使用了方案二。

在得到角速度进行控制时我们发现由于识别角度存在波动，导致舵机会在小范围内频繁波动，为了解决这个问题，我们使用了一阶指数平滑法对数据进行处理，最终得到了较好的效果。

速度平滑效果如图 4‑2下所示

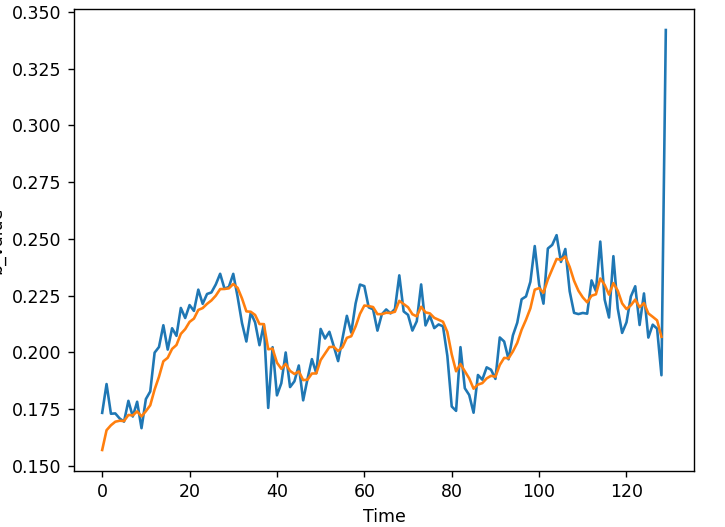


图 4‑2

### 可视化界面调试

为了进一步减少调试时间，提高队员调试体验，并能够可视化结果，我们做了一个可视化界面，能实时显示小车摄像头数据和控制及反馈角度，并且能动态调整参数，极大缩短了参数调整时间。调试界面如图 4‑3所示。



图 4‑3

## 识别交通灯

识别交通灯上我们使用传统计算机视觉的颜色识别，对图像进行高斯模糊后转到HSV颜色空间，然后裁剪相应的颜色空间分别得到红灯和绿灯，然后进行判断。同时我们，进行了aruco码识别，在能识别到aruco码时先根据比例裁剪出红绿灯区域，减少误识别，为了进一步防止误识别，我们对结果进行了中值滤波，更加准确，最终得到较好的效果。

# 中央调度系统开发

本项目使用socket+pyqt5+Qtdesigner工具开发，技术实现主要包括socket通信开发以及ui可视化界面开发。服务端在Linux系统中运行，通过TCP/IP协议与客户端交互。

## socket通信

Window和ros使用TCP/IP协议进行通信，服务端在ubuntu系统，客户端在window系统。Socket 作为进程通信机制，是 2 个程序之间进行双向数据传输的网络通讯端点，用一个地址和一个端口号标识Socket 通常也称作“套接字”，用于描述 IP 地址和端口。socket通信结构如图 5‑1所示。

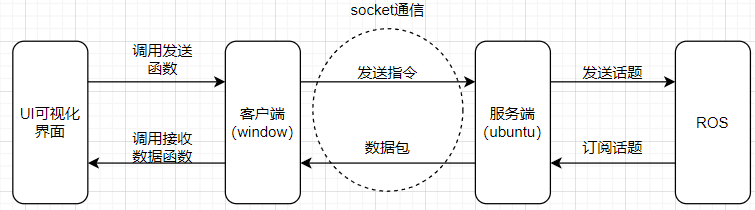


图 5‑1

### 通信模块封装

如果将通信模块与ui界面在同一个文件中，功能耦合性高，会导致程序越发臃肿，不方便新功能的加入，为提高程序易读性和可维护性，我们将客户端的功能封装在socket\_client模块中，该模块提供了发送和接受消息的接口，以便ui界面端随时读取，并写了较为完善的README文件，方便他人使用，提高了程序复用性。

### 多线程同步发送与接受消息

为了实现同时发送与接受消息，使用了多线程分别进行发送信息和对接受的信息进行处理。我们在连接成功后，使用python的threading创建了新线程处理接受的数据，先根据信息中的话题名对数据进行分类并保存，界面端每次接受数据都会对数据进行重置，保证每次读取到的都是最新数据且不会重复读取同一次数据。

服务端与ROS直接连通，收到话题后可通过socket发送给客户端，同时收到客户端的消息后会以话题形式发出。为提高程序易读性和可维护性，客户端的功能封装在socket\_client模块中，该模块收到服务端的数据后会更新消息，以便ui界面端随时读取，同时提供了发送消息的接口，供ui界面端使用,并写了较为完善的README文件，方便他人使用。

### 自定义通信规范

为了实现ros中话题与window互通，我们使用JSON格式定义了一种话题传输规范，JSON(JavaScript Object Notation, JS ) 是一种轻量级的数据交换格式，且易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率，非常适合用于socket通信。

目前支持字符串、速度、导航目标点和里程计数据，为了减少通信数据量，缩减了话题中部分冗余数据只保留最关键的话题名和重要数据，提高通信效率。并且封装在独立的json文件中，非常方便增加新的数据类型，程序可维护性高。

### 视频流传输

在视频流的传输中，使用上述的消息类型和TCP协议会占用大量带宽，导致无法实时传输，考虑到视频流传输内存大以及实时性的要求，我们使用了UDP协议进行通信，同时为了减少通信负担，我们先将图像进行裁剪再压缩，同时在帧头和帧尾添加校验位，防止数据错误，最终达到了30fps（摄像头帧率）且无肉眼可见延迟。

## 可视化界面

可视化界面由pyqt5+QtDesigner开发，先用QtDesigner进行整体界面的设计，再用PyQt5对各个控件功能进行设置。同时在线上赛作品的基础上，应对国赛做出了不少调整。

### 界面结构介绍

Ui界面主要包括状态显示栏，消息发送端及可视化数据三部分组成，我们使用QtDesigner进行界面的整体设计，再使用QSS对界面进行美化，美化过程中参考了大量图片和颜色，进行了多次测试，得到一套较为美观的配色。然后使用PyQt5对各种按键进行配置，实现最终的效果。为了符合比赛要求，将命令发送放在正中间并加大按键和字体，防止比赛时紧张导致误触，中央调度系统主页如图 5‑2所示。



图 5‑2

### 各功能介绍

左边消息总表会提示是否连接正常以及发送的消息，小车状态栏显示小车控制速度(cmd\_vel)、里程计速度(odom)，车道线及红绿灯识别状态、频率和导航的状态，方便调试。比赛时上位机操作手可通过状态栏观察是否到达从而判断是否发下一个命令，节省与队员沟通时间。

右上角的框可以实时显示小车摄像头数据，右边的速度趋势图显示表里程计以及发布的速度信息并进行可视化的功能，能根据需要改变读取的数据信息，并对数据以图的形式进行可视化显示。我们使用Matplotlib对读取到的数据以折线图的形式进行可视化显示，能够很好地反映小车当前的运动情况，对调节电机PID、设置局部路径规划的小车参数以监控小车运动有重大帮助。

为了方便调试，我们添加了远程发布导航目标点，发布速度和遥控功能，能切换阿克曼底盘和全向轮底盘两种模式。全向轮模式下，向左和向右会发布y方向速度实现横移，满足全向轮要求，而阿克曼底盘向左和向右会发布z轴角速度，实现舵机转动，同时能控制速度的增加和减小，更符合实际需求。

# 阿克曼运动控制

## 概述

底层实现采用stm32f103RCT6单片机，软件层面，为更有效地利用MCU的资源、简化应用软件的设计、缩短系统开发时间，更好地保证系统的实时性和可靠性，在stm32上采用了嵌入式实时操作系统——FreeRTOS，旨在利用FreeRTOS的任务管理、时间管理、信号量、消息队列、内存管理、协程等功能，充分发挥MCU的性能，为机器人的定位、导航提供稳定、可靠、响应迅速的硬件设施及高度可信赖的传感器信息。

程序设计大致分为四个部分：硬件驱动层(Applicaton/UserBsp)、算法层(Applicaton/MathLib)、通讯层(Applicaton/Com)、应用控制层(Applicaton/AppCtrl)。

* 硬件驱动层(Applicaton/UserBsp)：即驱动stm32的外设（USART、DMA等）和必要的硬件设施（带编码器直流减速电机、舵机、IMU、PS2、OLED显示屏），并为外界提供API接口。
* 算法层(Applicaton/MathLib)：为程序执行提供必要的算法支持，如电机控制所需的PID、IMU原始数据处理、姿态解算所需的均值滤波、滑动滤波、滑动中值滤波、RC低通数字滤波、IIR数字滤波、卡尔曼滤波、互补滤波等。
* 通讯层(Applicaton/Com)：当需要使用软件来控制IO电平模拟通讯协议时使用，如读取MPU9250使用IIC时。
* 应用控制层(Applicaton/AppCtrl)：即FreeRTOS下的多条线程。

图示

描述已自动生成

图 6‑1

硬件层面，为避免电路板烧毁等意外情况而导致影响机器人调试进度，根据官方电路板原理图而自行设计了电路板，同时购买了多份可能损耗的硬件设施，如舵机、电机等。

## 实现方案设计分析

底层实现任务主要有二：一是接收ROS发送过来的控制指令，二是发送传感器信息（odom、imu、mag）。针对这一要求，主要在odom数据在ROS层处理，还是在stm32上处理设计了两套方案。

### Odom数据底层处理

即在底层直接积分求得机器人的位姿(x,y,)。

优点：避免通讯丢包问题而导致不必要的系统误差。

缺点：在ROS层无法得到传感器的原始数据；数据量较大。

### Odom数据ROS层处理

ROS层获取左右电机编码器的脉冲值，再进行积分：

优点：数据量小；能够直接获取传感器原始数据，方便后续标定等工作。

缺点：通讯可能产生的延迟、丢包问题；

最终，权衡之下我们选取了第二套方案。

### 前轮转向标定

在实际调试过程中，发现实际获取得到的机器人角速度与下发的目标角速度并不吻合，经过问题排查发现，由于前轮舵机角度和前轮转角是非线性的，并且存在装配误差，前轮转角和下发的控制角度并不吻合，于是编程设计了——Odom标定包（Odom\_Calib），通过获取IMU的角速度，对前轮转角进行修正，最终得到前轮转角与舵机角度的对应关系，再通过最小二乘拟合的方法，得到其函数关系，效果如下图 6‑2所示。

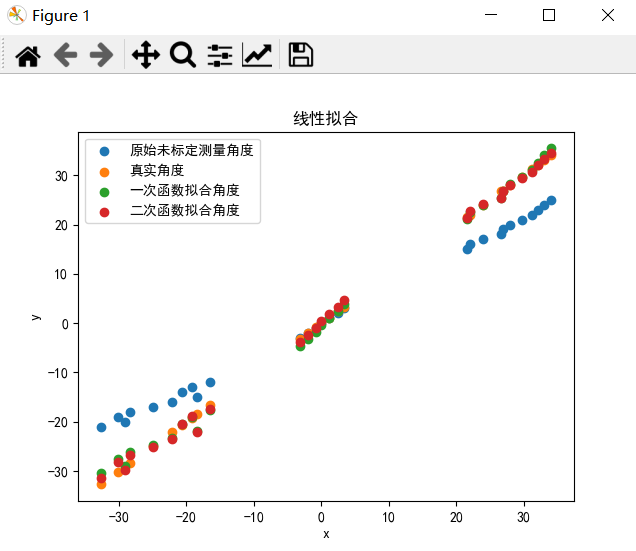


图 6‑2

# 总结

本次比赛要求是小车到达指定目标点完成运货、卸货，并在途中通过识别红灯停止行进、识别绿灯继续行进以及检测车道线通过S弯。本报告主要介绍了我们的技术方案：采用AMCL功能包，实现对小车位于全局地图的位姿定位，使用 A\*算法进行全局的路径定点导航规划，纯追踪算法进行局部路径规划，使用PSPNet神经网络进行车道线检测。

我们从这次比赛中受益良多，学会了如何团队合作，每个人负责不同的部分，比赛任务才能顺利完成。在调车的过程中，遇到了许多大大小小的问题，我们一起讨论、集思广益，最终这些问题才能一一化解，这十分考验我们的实践能力和耐心，我们每个人都从中积累了宝贵的经验，更加懂得了如何将人工智能技术应用到我们的生活中。在后续的比赛中，我们也将更加努力拼搏，相信在不久的赛场上，我们的小车会跑出别样的风采！