智慧物流技术说明文档

1. 概述

本文为南京工业大学“御风”战队“2021全国大学生智能汽车竞赛 --智慧物流创意组”的技术说明文档，主要对项目各项功能技术要点实现进行具体说明。

1. 硬件部分

硬件采用官方提供的设备，其中硬件层连接如图1所示

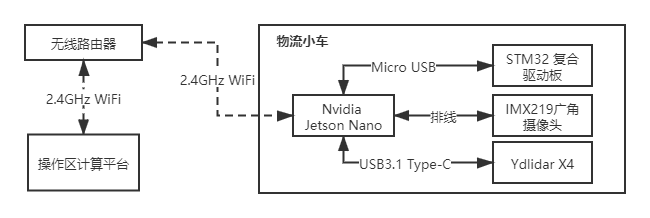


图 1 硬件层连接

1. 软件部分

团队智能车的计算平台使用 Ubuntu 18.04 操作系统，各模块基于 ROS-Melodic 开发。下位机代码使用了官方提供的Minibalance项目，对应的上位机驱动则基于 qingzhou\_ws 项目修改而成。

软件层连接如图2所示

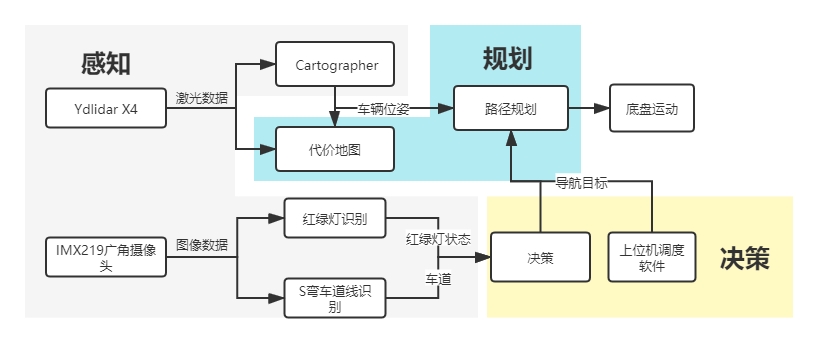


图 2 软件层连接

* + 1. 上位机调度软件
       1. 运行环境及界面

团队智能车的上位机采用Qt进行编写，运行环境为Linux系统，其中图形化界面如图3所示。界面包括轻舟机器人位姿显示、初始等待区、装货区、卸货区位置发送命令按钮以及摄像头图像显示功能，上位机调度软件界面如图3所示。

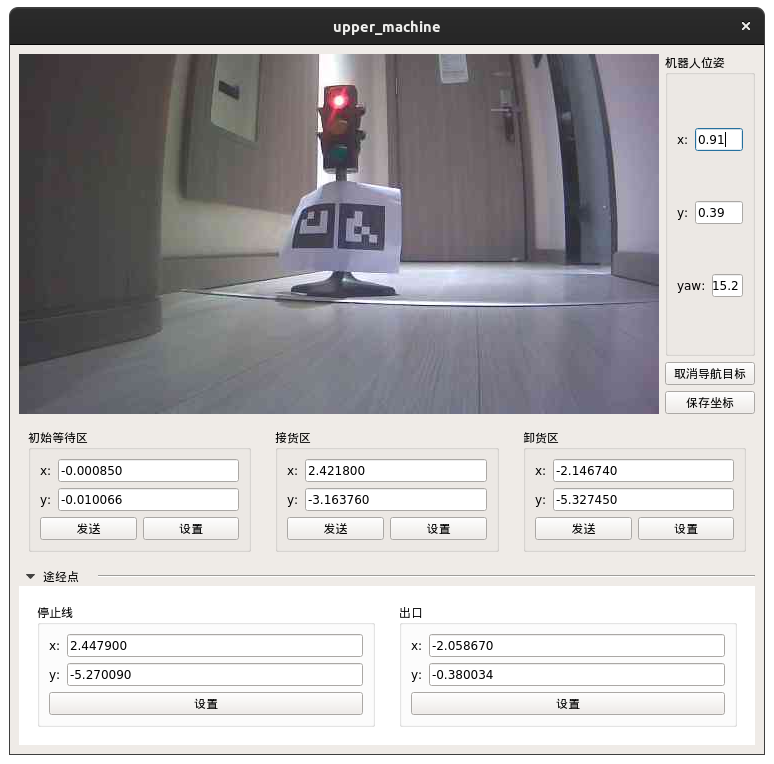


图 3 上位机调度软件界面

* + - 1. 通信

上位机程序使用ROS的消息机制与其他节点通信。其中，点击各导航目标点的“发送”按钮后，程序会将该点的坐标发送至move\_base节点；上位机显示的图传画面的图像来自于摄像头发布节点，使用ROS image\_transport作为协议基础，为保证图传的实时性，我们对图像进行了降采样以及JPEG压缩，以减小其大小、加快传输速度。

* + 1. 建图与定位

建图与定位采用Google开发的cartographer，cartographer主要由 Local SLAM 和 Global SLAM 两部分组成，如图4所示。Local SLAM利用里程计(Odometry)和IMU数据进行轨迹推算，给出小车位姿估计值,将位姿估计值作为初值，对雷达数据进行匹配，并更新位姿估计器的值,雷达一帧帧数据经过运动滤波后，进行叠加，形成子图(submap)。Global SLAM则是通过Loop Closure来进行闭环检测，来消除累积误差：当一个submap构建完成，也就是不会再有新的laser scan插入到该submap时，算法会将该submap加入到闭环检测中。

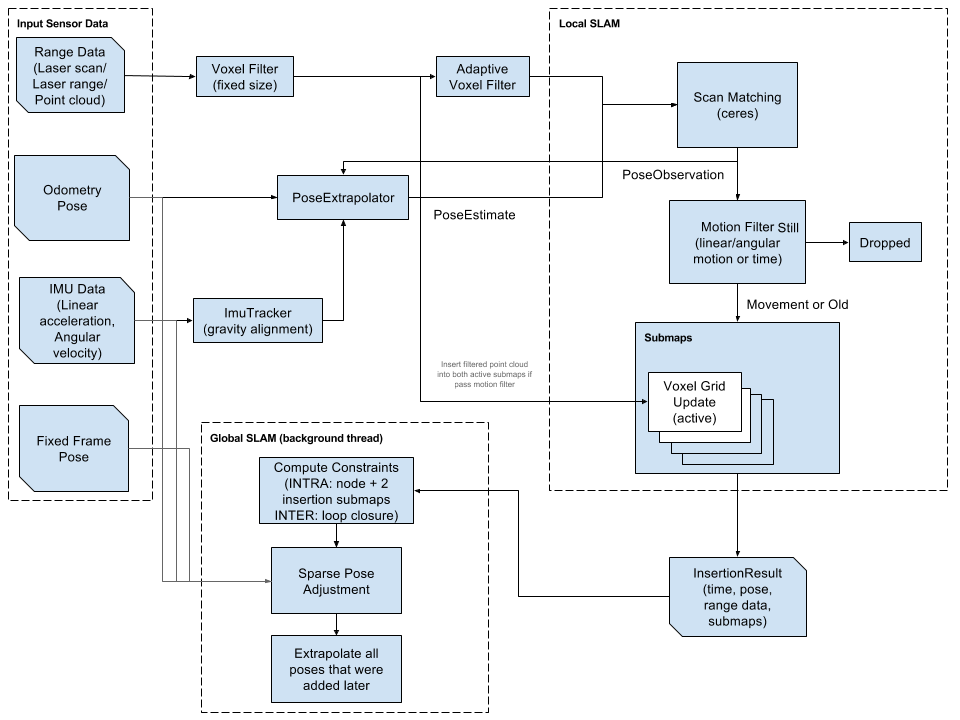


图 4 cartographer 算法系统框图

cartographer相比于gmapping在建图方面有较大优势。

1. gmapping算法对里程计的依赖大，当里程计计算有误差或者更新频率太低会对建图造成很大的影响，构建地图时如果快速移动，很可能出现这种偏移错位的情况。
2. gmapping本身不具有回环检测的功能，仅依靠粒子的多样性。在回环发生后根据正确粒子与错误粒子的权重差距来修正里程计累积的误差，回环越大，粒子耗尽的可能性就越高，地图越难在回环时修正回来。
3. cartographer累计误差较gmapping算法低，能天然的输出协方差矩阵，后端优化的输入项，仅仅依靠雷达也能跑出不错的效果。
4. cartographer带有闭环检测，当地图存在回环时可以很准确地检测到，构成的方形回环很规则。
5. cartographer算法相比gmapping，随着地图构建的扩大，内存消耗和计算量都很大，但是建图的效果更好。
   * 1. 导航与避障
        1. 全局代价地图
           1. 静态地图层

静态地图层由cartographer建图获得，包含场地中的障碍物快以及四周围挡。

* + - * 1. 膨胀层

膨胀层会对障碍物轮廓进行膨胀扩增，可以显著减小与障碍物的刮蹭，避免撞到墙壁。

* + - 1. 局部代价地图
         1. 障碍物层

该层通过激光雷达扫描到的点云创建，包含了激光雷达扫描平面内的墙壁，障碍物等。

* + - * 1. 膨胀层

膨胀层会对局部障碍物轮廓进行膨胀扩增，可以显著减小与障碍物的刮蹭，避免撞到墙壁

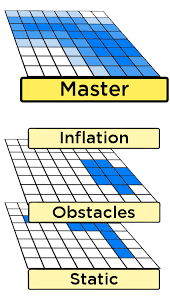


图 5 代价地图分解图

* + - 1. 路径规划
         1. 全局路径规划

全局路径规划采用Dijkstra算法，Dijkstra算法从物体所在的初始点开始，访问图中的节点。它迭代检查节点集中的结点，并将和该节点最靠近的尚未检查的结点加入待检查点集。该结点集从初始结点向外扩展，直到到达目标节点。 Dijkstra算法保证能够找到条从初始点到目标点的最短路径。

* + - * 1. 局部路径规划

由于该车采用阿克曼运动模型，因此更适合采用基于TEB的局部路径规划算法，但由于自动模式下，上位机无法使该车进行倒车，而TEB算法无法在路径规划中规避倒车，因此我们决定不采用TEB算法，而改用DWA算法。动态窗口法DWA是在曲率速度法基础上提出的，将移动机器人的位置控制转化为速度控制，将避障问题描述为速度空间带约束的优化问题。该算法在速度空间中采样多组速度，将有限的速度和加速度的运动约束考虑到动态窗口的设计中，模拟移动机器人以一定的速度在一定时间内的运动轨迹。在得到运动轨迹后，通过一个评价函数对这些轨迹打分，选取最优轨迹对应的速度来驱动移动机器人的运动路径规划，其效果如图6所示。

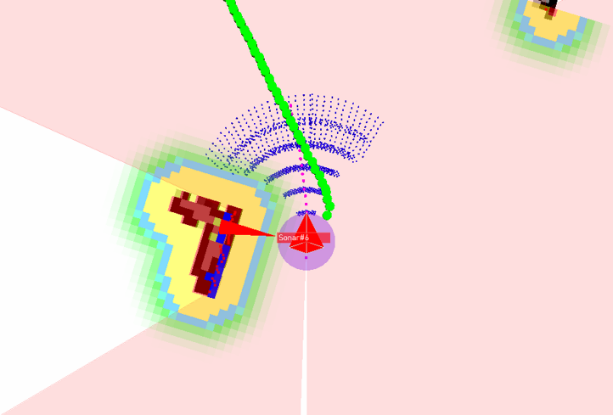


图 6 DWA 局部路径规划

* + - 1. Footprint

由于车子为长方体，因此根据车子的实际尺寸将footprint设为长方形。

* + 1. 红绿灯识别
       1. 绘制ROI区域

红绿灯为立式交通灯，比赛中共出现两种形式的灯光，分别为红灯和绿灯。首先调用opencv\_contrib中的aruco库对aruco码进行识别（由于考虑到识别可能存在误判且场地标签固定的情况因此我们直接通过标签id对识别数据进行过滤），根据aruco码在图像中的位置以及码与红绿灯在图像中的相对位置关系，框出红绿灯的ROI区域以减少外界颜色对进一步灯管颜色识别的干扰，如图7所示。其中在计算码与红绿灯之间相对关系时，考虑到比赛时可能存在aruco码贴反或距离过远只能识别到一个码的情况，所以对其进行分情况处理，鲁棒性更强。



图 7 红绿灯ROI

* + - 1. 灯光颜色检测与判断

在通过上述操作获得红绿灯ROI区域后，需要对灯光颜色进行提取，考虑到目标颜色为红色和绿色，如果采用HSV进行提取，需要对色彩空间进行转换，占用资源大，于是决定直接对RGB图像进行通道分离并通道相减，分别获得较为纯净的红色通道(红色通道-绿色通道)以及绿色通道（绿色通道-红色通道），对两个通道设置阈值进行二值化处理，并对其进行膨胀操作，获得两幅二值化图像，如图8，图9所示。计算图像中白色像素点的个数，对两图像中的白色像素个数进行比较，设定一个阈值K，若红色通道中的像素数减绿色通道中的像素数大于阈值K则为红灯，反之为绿灯。考虑到近大远小的关系，可将阈值K与aruco标签在图像中的宽度建立比例关系，从而确定K值。

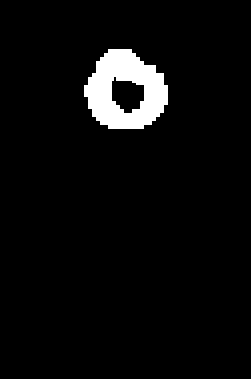
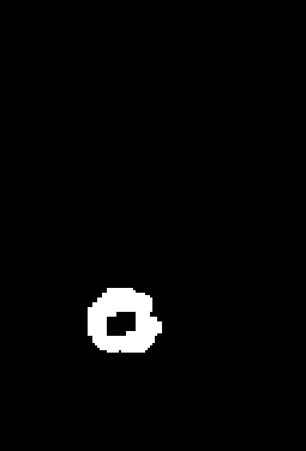
 

图 8 二值化红灯 图 9 二值化绿灯

* + 1. S弯道识别
       1. 车道线检测，绘制ROI区域

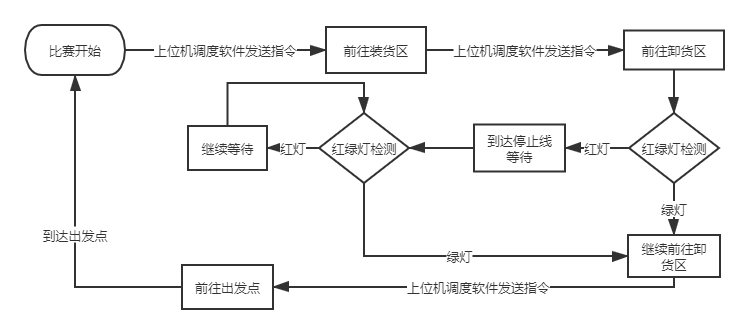
首先对图像尺寸进行裁剪（将图像上方裁去，只保留车道部分）并进行通道分离，通道相减，获得蓝色通道和红色通道的灰度图，使用高斯模糊去噪对图像进行腐蚀膨胀操作并分别进行canny算子边缘检测并利用霍夫变换检测直线，对直线进行过滤，取均值，只保留一条直线(由于箭头线为一段一段的蓝色箭头，因此霍夫变换检测直线只对红色和蓝色的车道边线有效)，将两图像分别检测到的直线的方程带入原图中，绘制ROI区域。

* + - 1. 箭头识别，获得导航目标点

接下来即是对ROI区域中的箭头线进行检测。首先对图像进行通道分离通道相减以及二值化处理，获得蓝色通道的二值化图像，对图像进行高斯模糊，闭运算，使得小块箭头连接在一起成为大块区域，在ROI区域进行Findcontours操作，过滤掉小面积的轮廓，保留大面积的轮廓。轮廓中坐标y值较小的点最为导航目标点，对坐标系进行转换，得到目标点在世界坐标系下的坐标，也就是2D地图下目标点的坐标。(最终由于nano性能不稳定等因素，决定放弃该方案，后期将对S弯进行更深度的研究分析，寻找更优的解决方案)

* + 1. 决策

决策框架如图10所示。



*图 10 决策框架图*

* + - 1. 调度软件发送指令

车辆由操作手在操作台通过调度软件发送指令，前往对应目标点，其中目标点点击顺序依次为装货区，卸货区，出发点。

* + - 1. 红绿灯检测

对红绿灯检测，为减小错误检测对结果的影响，对连续三帧的数据进行判断，如果连续三帧都为绿灯，则才判定为绿灯。