全国大学生智能汽车竞赛
讯飞-智慧餐厅
技
术
报
告

参赛学校 江苏理工学院

队伍名称 艾卡一队

领队老师 尹明锋

指导老师 范鑫

参赛学生 王博 陈昌凯 林浩

金圣昕 周林苇

二零二一年

关于技术报告使用授权的说明

本人完全了解2021全国大学生智能汽车竞赛创意组分赛关于保留、使用技术报告的规定，即参赛作品著作权归参赛者本人所有，比赛组委会和科大讯飞（苏州）科技有限公司可以在相关主页上收录并公开参赛作品的技术报告以及参赛作品的视频、图像资料，并将相关内容进行编纂收录。

参赛队员签名：王博 金圣昕 林浩
陈昌凯 周林苇

指导老师签名：尹明津 范鑫

日　　期：2021.3.19

目录

第 1 章 项目背景及意义

1.1 背景及意义

全国大学生智能汽车竞赛是以智能汽车为研究对象的创意性科技竞赛，是面向全国大学生的一种具有探索性工程的实践活动，是教育部倡导的大学生科技竞赛之一。竞赛以立足培养，重在参与，鼓励探索，追求卓越为指导思想，培养大学生的创意性科技竞赛能力。

根据比赛要求，本方案在统一仿真平台 Gazebo 中实现，以激光雷达扫描赛道构建地图，根据设定的目标点生成全局路径，并在行驶过程中通过激光雷达，识别障碍物，以局部路径算法规划局部路径，控制车模行驶，完成避障。

本设计方案在分析关键性问题的基础上，根据相关问题的研究现状，提出解决问题的技术方案及其方案实现，技术路线如图 1-1 所示。

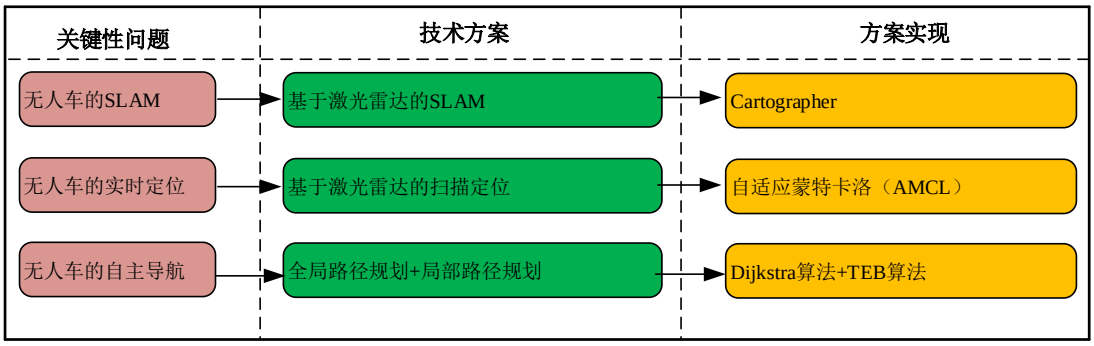

图 1-1 技术路线图

本设计方案在仿真平台 Gazebo 上能够比较流畅的行驶，规避障碍物，完成自主导航的比赛任务，本技术报告的章节内容如图 1-2 所示。

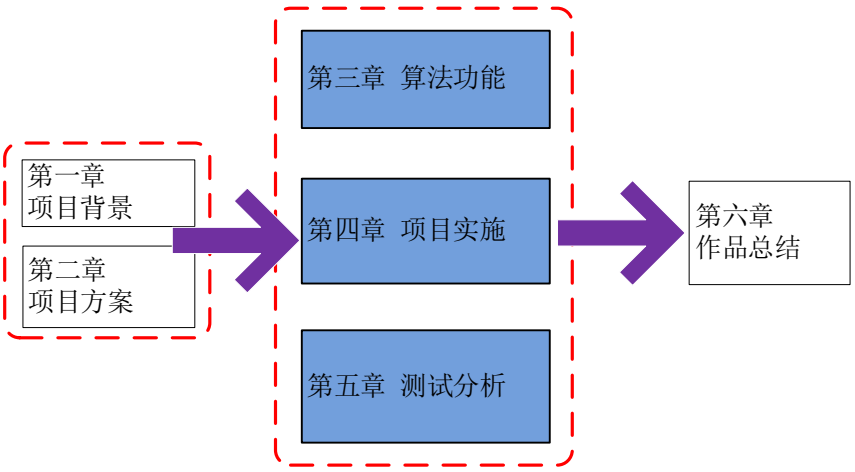

图 1-2 章节内容结构图

1.2 关键问题

讯飞创意组组线上仿真比赛平台统一用 Gazebo。赛道模型和无人车三维模型由组委会于赛前统一提供。线上比赛需要先把赛道模型导入 Gazebo，采用 ROS 中建地图的方式构建赛道地图，通过自主导航算法实现无人车完成从起点到终点的运动。仿真平台的传感器可以使用仿真车模自带的 IMU，激光雷达或摄像头，仿真平台自主导航算法不限。因此，依据比赛任务，本次线上仿真比赛的关键性问题分别为：未知环境的地图构建、无人车的实时定位、无人车的自主导航，下面就这些关键性问题展开详细阐述。

(1) 无人车的 SLAM

无人车的 SLAM(Simultaneous Localization and Mapping)是指无人车在未知环境中，在运动中使用携带的传感器不间断的采集周围环境信息来定位自身的位置，再根据自身位置构建环境地图。SLAM 可以直接有效地获取环境地图，为无人车路径规划提供直观的可视化信息，是实现无人车自主导航的关键技术。

(2) 无人车的实时定位

无人车的实时定位是确定无人车在赛道环境中所处位置的过程，具体地说，定位是利用先验环境地图信息、无人车位姿的当前估计以及传感器的观测值等输入信息，经过一定的处理和变换，产生更加准确的对无人车当前位姿的估计，可靠的定位是无人车最为基础且重要的一项功能，是无人车实现自主导航的前提条件

(3) 无人车的自主导航

无人车的自主导航分为两个阶段：首先根据起始点和目标点利用路径规划算法规划出一条无碰撞的目标路径，接下来根据目标路径利用轨迹规划算法规划出连续的速度序列，驱使无人车能够跟踪目标路径前往目标点，同时在无人车的移动过程中会不断地进行实时定位，对轨迹的输出进行优化。

第 2 章 项目方案设计

第十六届智能车讯飞智慧餐厅组比赛任务是根据组委会提供的无人车 URDF 模型和赛道模型，让无人车通过车载传感器建立赛道地图，通过导航算法规划从起点到终点的路径，并能自主跟随路径躲避锥桶等障碍从起点运动到中终点。

分析下来，比赛任务就分为建图、定位、路径规划、轨迹跟踪这几个部分。

通过近一个月的实验，本方案最终使用 Cartographer 算法构建出赛道的二维栅格地图，使用自适应蒙特卡洛（Adaptive Monte Carlo Localization）算法实现车模在整个赛道地图中的定位，路径规划使用 Dijkstra 全局路径规划算法生成从起点到终点的全局路径，在车模沿着全局路径行驶过程中使用时间弹性带算法（Time Elastic Band）规划局部路径并发布速度指令的话题信息给运动控制器，对车模进行控制，使车模完成轨迹跟踪，规避赛道中的障碍物，从而完成比赛任务。

图 2-1 方案流程图

2.1 相关研究

(1) 无人车的 SLAM

SLAM 概念[1]首先由 R.Smith 和 PCheesman 提出，阐明了 SLAM 问题的基本框架，提出了解决方案的关键问题和地图构建算法，采用卡尔曼滤波器来估计机器人位置及姿态与环境地图的特征信息。根据使用的 SLAM 算法来分，SLAM 方法可以分为两大类：基于概率估计的方法和基于非概率估计的方法。其中，基于概率估计的 SLAM 方法有：基于扩展卡尔曼滤波的方法(Extened Kalman Filter，EKF)、基于粒子滤波(Particle Filter, PF)的方法等。基于非概率的方法有：基于图优化的方法、基于扫描匹配的方法等。

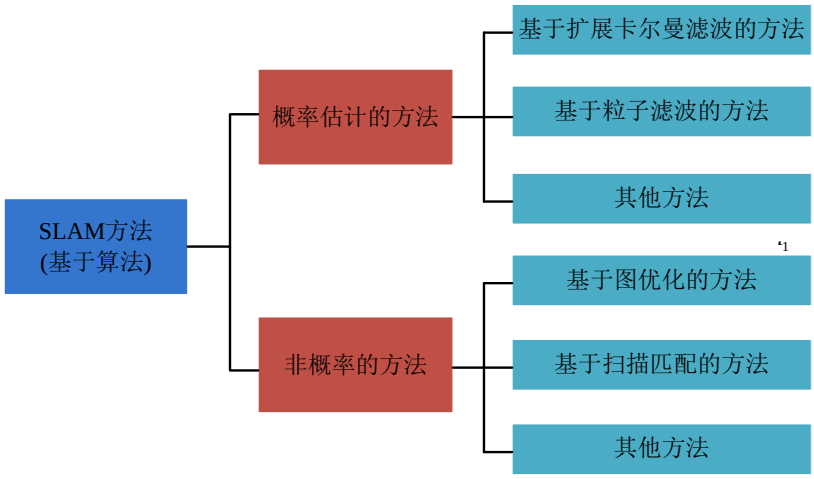

图 2-2 SLAM 基于算法的分类

早期 SLAM 问题主要是基于贝叶斯滤波框架，贝叶斯滤波是基于概率估计的 SLAM 方法的基础。EKF-SLAM 是利用扩展卡尔曼滤波对机器人的位置进行估计，随着观测数据的增加，计算量不断增大，计算成本高；并且 EKF-SLAM 依赖于正确的数据关联，数据关联失败会导致见图定位出错。因此，EKF-SLAM 不适用于大范围的环境构图。

近年来，在传感器技术的发展推动下，SLAM 算法应用高频率的激光雷达等距离传感器[2]，通过扫描得到的一系列数据帧与现有的地图进行匹配实现机器人的定位，并结合多分辨率的地图，如 S.Kohlrecher 等人提出的 Hector-SLAM 算法[3]，以及 E.Pedrosa 等人提出的在线的同步定位通过使用动态似然估计的快速扫描匹配算法构建地图[4]。此外，W.HESS 等[5]提出了 Cartographer-SLAM 算法，通过分支定界的方法并对扫描进行回环检测，在构建大范围环境地图精度高、鲁棒性好。

(2) 无人车的实时定位

定位问题是无人车实现自主移动要解决的首要问题，无人车必须了解自身与周围环境之间的位置关系，以便做出更准确的判断和选择。由于具有速度快、精度高等优点，利用激光雷达进行位姿估计的方法已经广泛应用于移动机器人领域。基于激光雷达位姿估计是的将姿势信息集成到其他基于激光雷达的算法中更容易，在没有任何校准过程的情况下，系统的准确性、可靠性，易于实施和降低成本得到改善。

杨明等[6]提出了一种基于二维激光雷达的实时位姿估计方法，包括基于

Hough 变换的切线角度直方图算法和迭代切线加权最近点算法，通过仿真数据和室外环境实际数据的大量实验结果表明，具有速度快、精度高、实用性广和对噪声、遮挡和类孔径问题鲁棒性高等特点。畅春华等[7]基于二维激光雷达提出新算法，首先将环境数据聚类为不同的障碍物，之后利用障碍物的特征匹配关联相邻 2 帧的障碍物，最后通过优化位置序列间的方向角实现机器人的位姿估计，显著提高了计算效率。

(3) 无人车的实时导航

目前机器人常用的导航方式包括惯性导航、电磁导航、视觉导航、激光导航等。这些导航方式采用不同的传感器和导航策略，应用于不同环境状况。

史风栋等[8]利用利用激光的准直性和不发散性对需要导航物体所处的位置进行精确定位，机器人通过激光传感器发射激光束，这些激光束在遇到障碍物时会反射回来，传感器通过采集这些反射回来的激光束，来确定机器人当前的位置和方向。激光导航不受电磁干扰，对光线比较敏感，导航精度比较高，安装简单。激光传感器可应用于 2D 或者 3D 的室内外环境中机器人的导航定位。

2.2 方案对比

Gmapping 是基于粒子滤波的算法，是一个基于 2D 激光雷达使用 RBPF（Rao-Blackwellized Particle Filters）算法完成二维栅格地图构建的 SLAM 算法。随着场景增大所需的粒子增加，因为每个粒子都携带一幅地图，因此在构建大地图时所需内存和计算量都会增加。因此不适合构建大场景地图。并且没有回环检测，因此在回环闭合时可能会造成地图错位，虽然增加粒子数目可以使地图闭合但是以增加计算量和内存为代价。

所以不能像 Cartographer 那样构建大的地图。Gmapping 和 Cartographer 一个是基于滤波框架 SLAM 另一个是基于优化框架的 SLAM，两种算法都涉及到时间复杂度和空间复杂度的权衡。Gmapping 牺牲空间复杂度保证时间复杂度，这就造成 Gmapping 不适合构建大场景地图。严重依赖里程计，无法适应无人机及地面不平坦的区域，无回环（激光 SLAM 很难做回环检测），大的场景，粒子较多的情况下，特别消耗资源。故在此不采用 Gmapping 的算法建图。Gmapping 构建的二维栅格地图如下图 2-3。

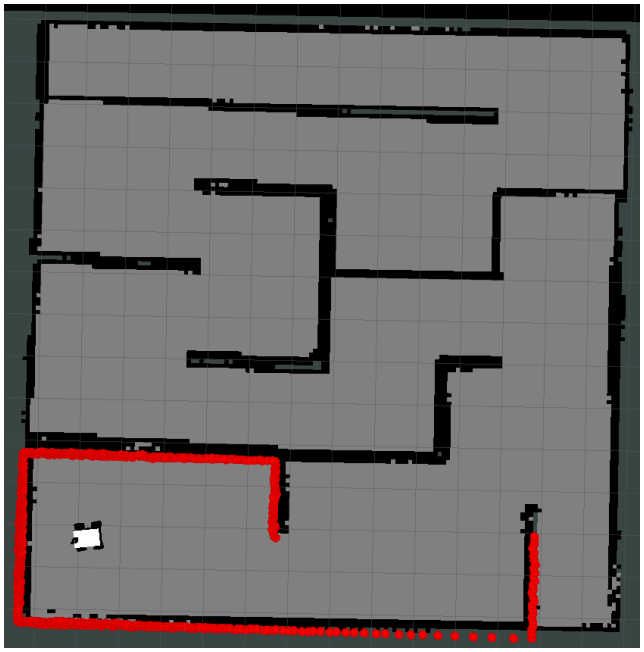

图 2-3 Gmapping 构建的二维栅格地图

Hector_slam 采用高斯牛顿的扫描匹配方法，对当前时刻的点云数据和前一时刻的点云数据进行匹配，找到激光点集映射到已有地图的刚体转换。优点：不需要使用里程计，可以用于地面不平坦区域及空中飞行器、使用多分辨率地图能避免局部最小值；缺点：要求雷达更新频率较高，测量噪声小或者机器人运动速度低、无法利用精确的里程计信息，故在此不采用 Hector_slam 的算法建图。

Hector_slam 构建的二维栅格地图如下图 2-4。

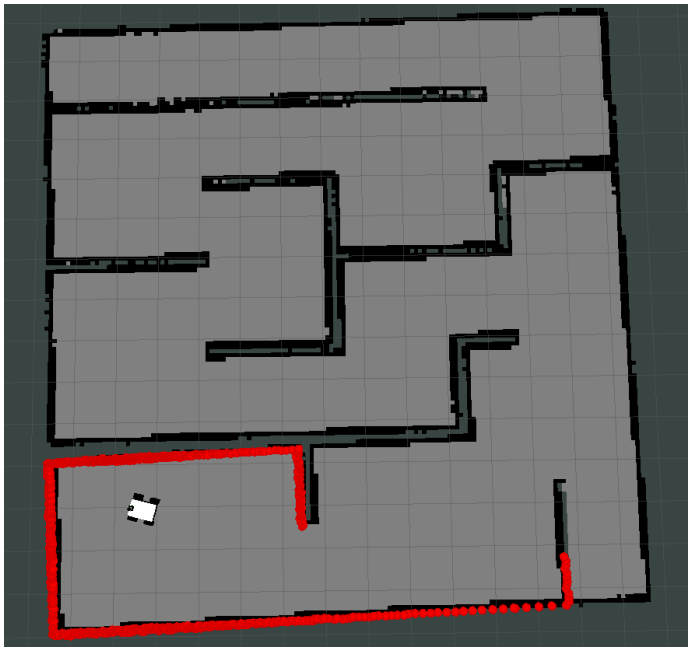

图 2-4 Hector_slam 构建的二维栅格地图

由上图2-3和图2-4两种算法所建的地图来看，其建图与下图2-5 Cartographer所建的二维栅格地图相比，明显建图效果较差。

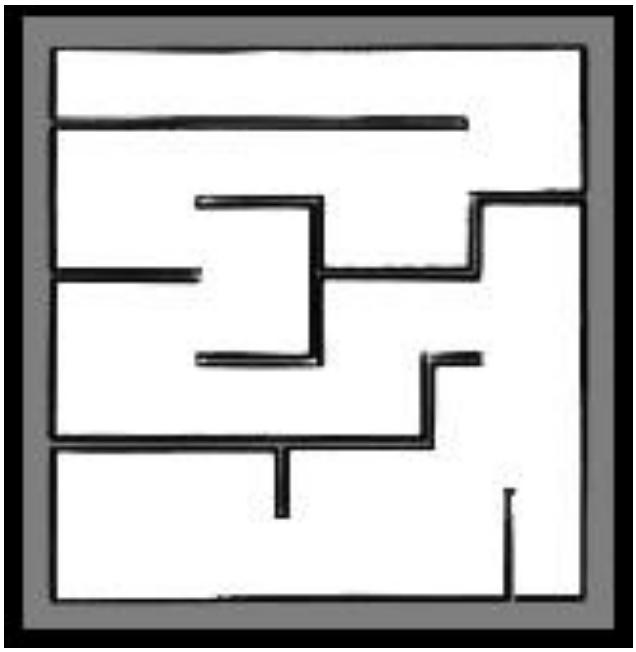

图 2-5 Cartographer 构建的二维栅格地图

第 3 章 项目算法功能

3.1 Cartographer 建图

Cartographer 是谷歌宣布开放的即时定位与地图建模库，开发人员可以使用该库实现机器人在二维或三维条件下的定位及建图功能。Cartograhper 的设计目的是在计算资源有限的情况下，实时获取相对较高精度的 2D 地图。考虑到基于模拟策略的粒子滤波方法在较大环境下对内存和计算资源的需求较高，Cartographer 采用基于图网络的优化方法。目前 Cartographer 主要基于激光雷达来实现 SLAM，谷歌希望通过后续的开发及社区的贡献支持更多的传感器和机器人平台，同时不断增加新的功能。

Cartographer 结合了 local 和 global 两种方式进行 2d SLAM，local 方式就是前端匹配中通过 submap 进行 scan matching。 global 方式就是回环检测，因为是对全局的 submap 进行匹配。local 部分（也就是前端），接收处理后的输入数据，对激光数据进行 scan matching ，scan matching 通过匹配 lidar 数据得到一个位姿，将当前帧激光插入到子图当中。如果传感器没有运动，是静止状态，那这一帧数据需要去除）这样一帧一帧的激光数据处理，我们不断更新得到子图。global （后端）这部分主要是一个回环检测，后端优化，在这其中提到了加速的方法采用的分枝定界法。

Cartographer 在前端匹配环节区别与其它建图算法的主要是使用了 submap 这一概念，每当或得一次 laser scan 的数据后，便与当前最近建立的 submap 去进行匹配，使这一帧的 laser scan 数据插入到 submap 上最优的位置。先有一定数量的 laser scan 构建 submap，由 submap 拼接成地图，所谓的回环检测，就是间隔一定数量的扫描进行一次所有 submap 的图优化（SPA，运用了分支定界原理进行加速），但这种用有误差的估计量去作为约束去优化估计量，总有种自己估计优化自己的嫌疑，跟视觉 SLAM 喜欢运用词袋模型检测是否回到之前来过的地方的算法。累计误差较前两种 SLAM 算法较低，能天然的输出协方差矩阵，后端优化的输入项。成本较低的雷达也能跑出不错的效果。所以采用 Cartographer 算法建图。

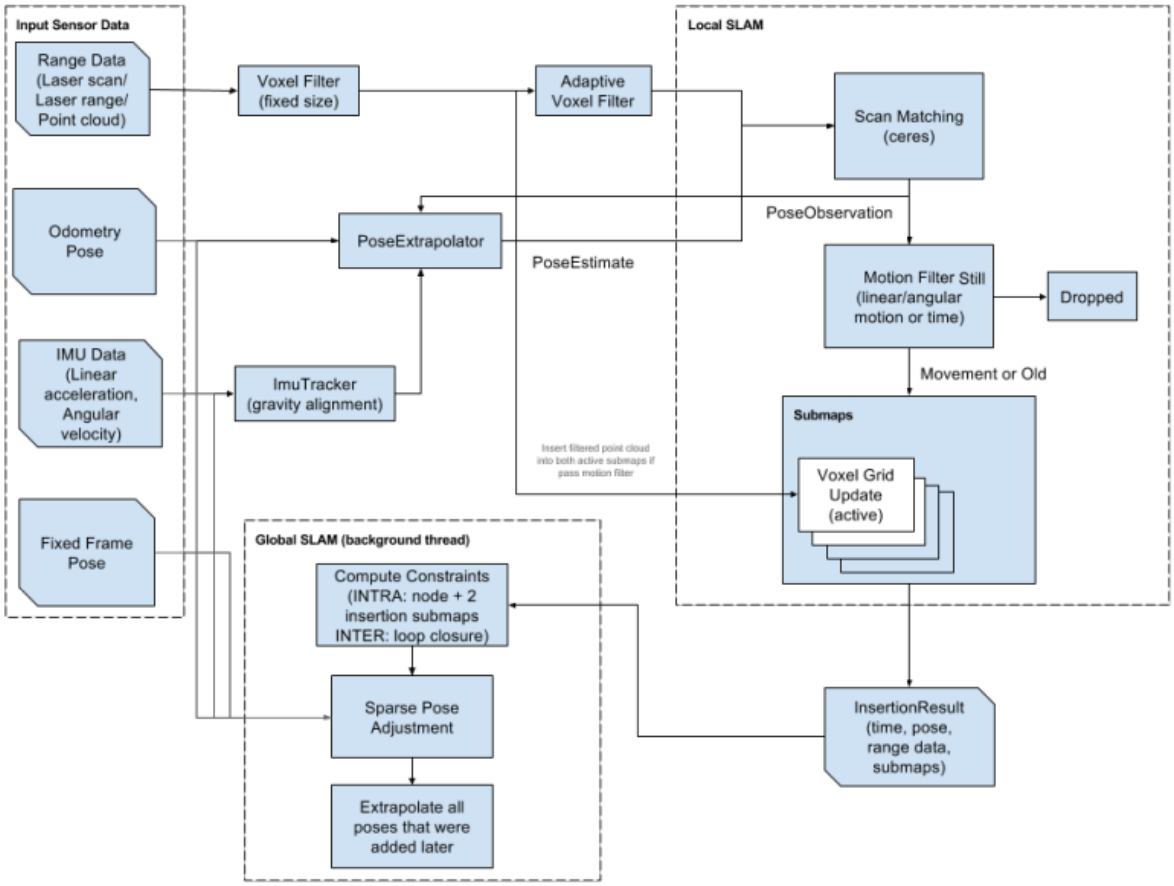

图 3-1 Cartographer 的流程图

3.2 基于 AMCL 的定位

蒙特卡洛：是一种计算位置概率的思想或方法，比如一个矩形里面有个不规则形状，计算不规则形状的面积就可以采用将粒子均匀的撒在矩形上，然后统计不规则形状里的粒子的个数和剩余地方的豆子个数。矩形面积已知，所以就通过估计得到了不规则形状的面积。拿机器人定位来讲，它处在地图中的任何一个位置都有可能，这种情况下我们判断一个位置的置信度的方式就是也使用粒子，哪里的粒子多，就代表机器人在哪里的可能性高。

常用的蒙特卡洛定位(MCL)算法进行分析，针对 MCL 算法存在粒子丢失问题，采用自适应蒙特卡洛定位(AMCL)算法作为本文机器人的定位方法。针对导航过程中存在颠簸和打滑导致定位不准问题，设计了一种利用数据融合算法将惯

导和里程计数据进行融合，实时纠正机器人位姿的方法。

自适应蒙特卡洛的自适应体现在：解决了机器人绑架问题、解决了粒子数固定的问题。算法流程如图 3-2 所示。

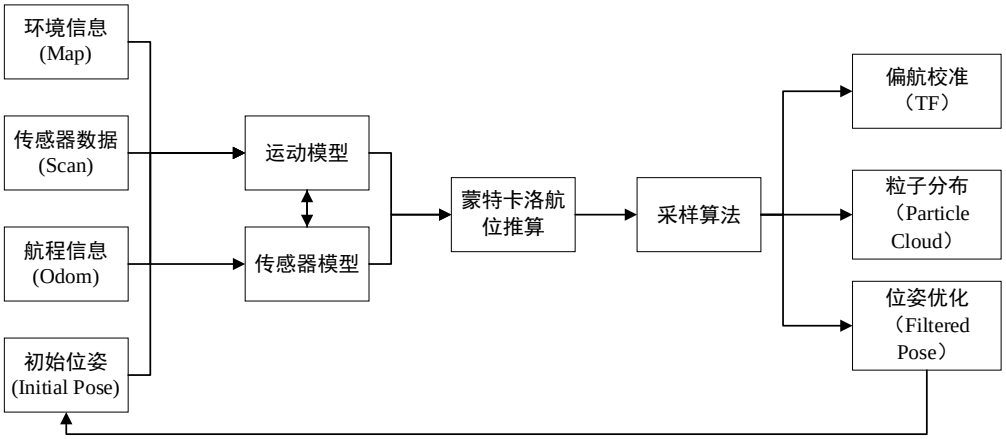

图 1-2 自适应蒙特卡洛流程图

3.2.1 代价地图

要使车模在赛道中自主导航需要在所建地图的基础上进行路径规划，该地图不仅需要反映实际地图的信息，还需要反映经过每个点的代价。Navigation 中通过代价地图（costmap）来实现，代价地图由静态地图层、障碍地图层和膨胀组成。代价地图有全局代价地图（global costmap）和局部代价地图 local costmap）。

（1）第一层：静态地图层（StaticLayer），是由 Cartographer 所构建出的二维栅格地图；

（2）第二层：障碍物层（ObstacleLayer），包括了 2D 和 3D（voxel 层）的障碍；

（3）第三层：膨胀层（InflationLayer），膨胀层将障碍进行膨胀，以计算每个 costmap 单元格的代价。这三层组合成了 master map（最终的 costmap），供给路线规划模块使用。

costmap 初始化流程如图 3-3 所示：

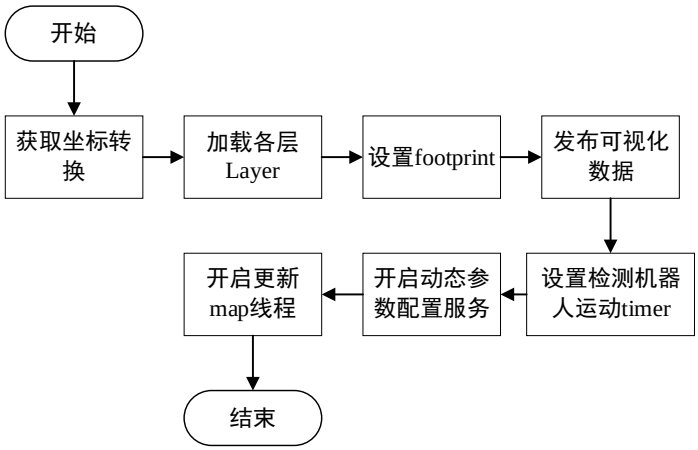

图 3-3 costmap 初始化流程图

costmap 初始化首先获得全局坐标系和机器人坐标系的转换，加载各个 StaticLayer，ObstacleLayer，InflationLayer；设置机器人的轮廓；实例化了一个 Costmap2DPublisher 来发布可视化数据；通过一个 movementCB 函数不断检测机器人是否在运动；开启动态参数配置服务，服务启动了更新 map 的线程。

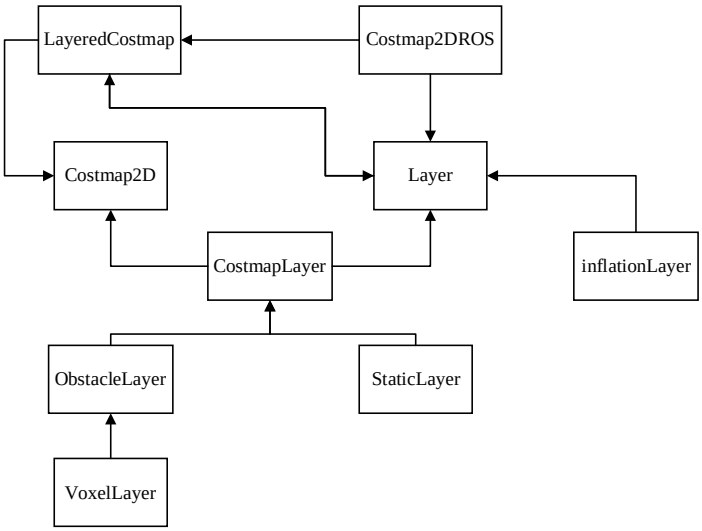

图 3-4cost_map 接口

Costmap 的 ObstacleLayer 和 StaticLayer 都继承于 CostmapLayer 和 Costmap2D，因为它们都有自己的地图，Costmap2D 为它们提供存储地图的父类，CostmapLayer 为它们提供一些对地图的操作方法。而 inflationLayer 因为没有维护真正的地图所以只和 CostmapLayer 一起继承于 Layer，Layer 提供了操作 master

map 的途径。

LayerdCostmap 为 Costmap2DROS（用户接口）提供了加载地图层的插件机制，每个插件（即地图层）都是 Layer 类型的。

3.3 路径规划

3.3.1 基于 Dijkstra 的全局路径规划

在代价地图的基础上，可以根据地图信息进行路径规划。全局规划是在已知地图的基础上规划一条最优路径。Navigation 中共有三种全局规划器：carrot 规划器、navfn 规划器和 global 规划器。navfn 规划器使用 dijkstra 算法来在起点和终点之间寻找最小代价路线。global 规划器建立了更灵活的替代 navfn 的选择，这些选择包括：（1）支持 A*算法（2）切换二次近似（3）切换网格路径。

A*算法是启发式搜索算法，启发式搜索即在搜索过程中建立启发式搜索规则，以此来衡量实时搜索位置和目标位置的距离关系，使搜索方向优先朝向目标点所处位置的方向，最终达到提高搜索效率的效果。

A*算法的基本思想如下：引入当前节点 x 的估计函数 f(x),当前节点 x 的估计函数定义为：

$$f(x)= g(x)+h(x)$$

其中 g(x)是从起点到当前节点 x 的实际距离量度（代码中可以用两点之间距离代替）；h(x)是从节点 x 到终点的最小距离估计，h(x)的形式可以从欧几里得距离或者曼哈顿距离中选取。

算法基本实现过程为：从起始点开始计算其每一个子节点的 f 值，从中选择 f 值最小的子节点作为搜索的下一点，往复迭代，直到下一子节点为目标点。

在全局规划中，本方案采用 Dijkstra 算法，Dijkstra 算法一种经典的基于贪心的单源最短路算法最短路径算法，用于计算一个节点到其他节点的最短路径。

Dijkstra 算法步骤如下：

图中的顶点分两组，第一组为已求出最短路径的顶点集合，用 S 表示。初始时 S 只有源点，当求得一条最短路径时，便将新增顶点添加进 S，直到所有顶点加入 S 中，算法结束。第二组为未确定最短路径顶点集合，用 U 表示，随着 S 中顶点增加，U 中顶点逐渐减少。

1. 初始时，S 中只包含起点 s，U 包含除 s 外的其他顶点，且 U 中顶点的距

离为起点 s 到该顶点的距离。若 s 与 v 相邻，则 U 中的顶点 v 的值为 s 到 v 的距离，若 s 和 v 不相邻，则 v 的值为∞；

2. 从 U 中选出距离最短的顶点 k，将顶点 k 加入到 S 中，从 U 中移除顶点 k，该顶点 k 作为新一轮距离计算的起始点；

3. 更新 U 中各个顶点到起点 s 的距离；

4. 重复步骤 2 和 3，直到遍历完所有顶点。

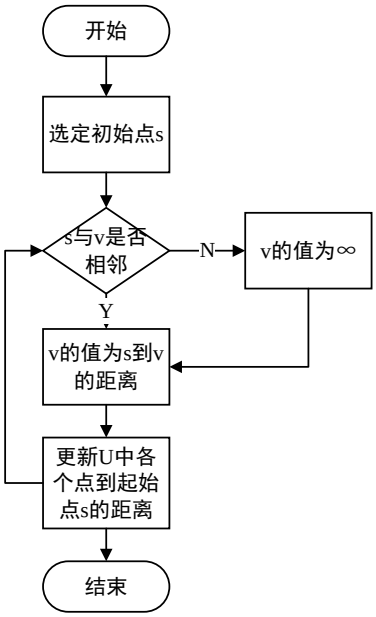

图 3-5 Dijkstra 算法流程图

3.3.2 基于 TEB 局部路径规划

时间弹性带算法（Time Elastic Band，TEB）是 2D 导航堆栈的 base_local_planner 的插件，该方法针对全局路径规划器生成的初始轨迹进行后续修正，从而优化机器人的运动轨迹，属于局部路径规划。

TEB 以 EB(Elastic-Band)算法为基础，EB 算法依靠传感器来实时地感知周围环境，根据传感器的观测信息来优化规划路径中的各个航点，EB 算法引入了力场的观念，各外部约束分别被描述为势场，周围环境中的障碍物提供外部的斥力，而相邻的目标航点提供内部的张力，整个轨迹被“拉”成一条橡皮筋。但是该方法忽略了机器人运动过程中应满足的几何约束，即在运动过程中保持瞬时航向角的连续性。而 TEB 方法在此 EB 的基础上引入了图优化的思想，将机器人的位姿状态和相邻状态的时间差作为图中待优化的节点，各个位姿状态下的运动学约束

以及环境约束作为待优化的边，并对位姿状态以及对应的时间差进行求解，从而得到满足约束的最优控制量。本质上是将轨迹规划问题视为了一个多目标优化问题。

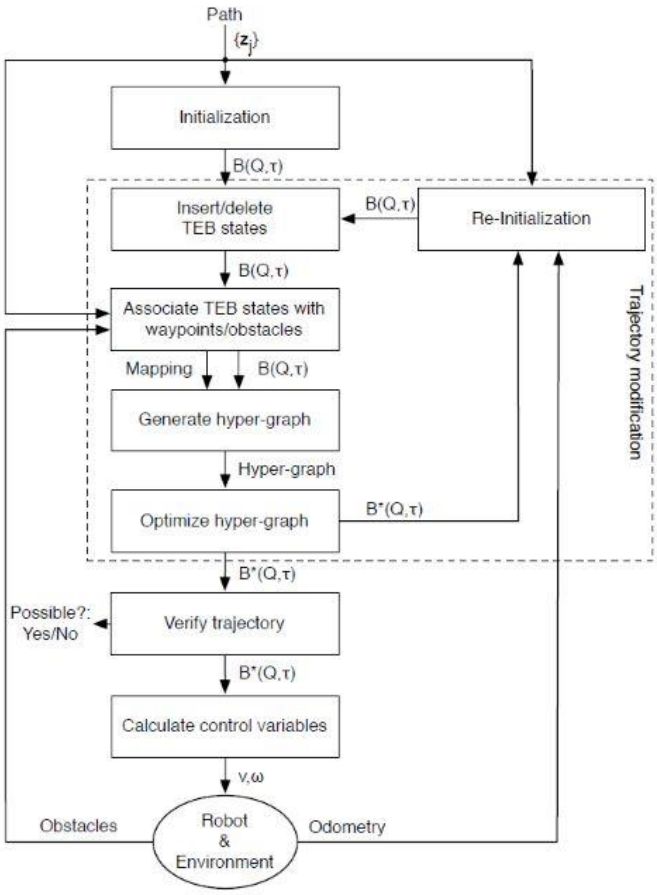

图 3-6 TEB 算法流程图

3.3.3 基于 DWA 局部路径规划

DWA 算法全称为 Dynamic Window Approach，其原理主要是在速度空间（v,w）中采样多组速度，并模拟这些速度在一定时间内的运动轨迹，再通过一个评价函数对这些轨迹打分，最优的速度被选择出来发送给下位机。

DWA 在机器人运动规划中属于 Local Planner。Local Planner 提供了一个控制器来驱动飞机上的移动基础。这个控制器用来连接路径规划器和机器人。通过使用地图，规划者可以为机器人创造一个运动轨迹，让机器人从一个开始到一个目标位置。在此过程中，规划人员至少在机器人周围创建了一个值函数，表示为

一个网格映射。这个值函数编码遍历网格单元的成本。控制器的工作是使用这个值函数来确定 dx，dy，dtheta 速度来发送给机器人。

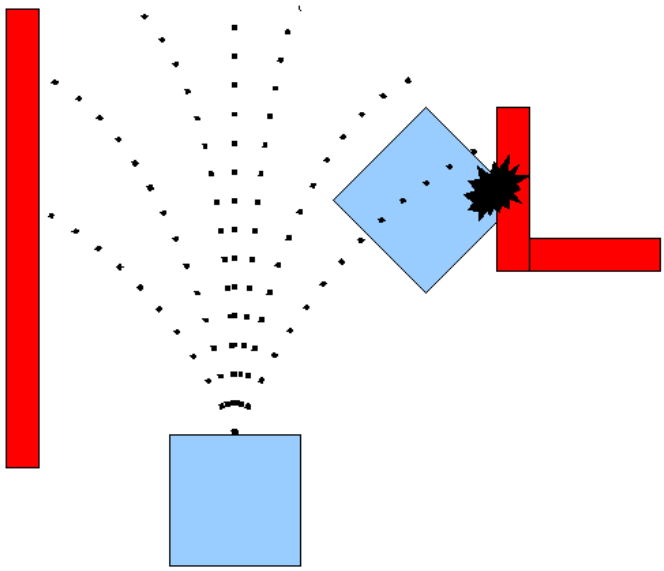

图 3-7 DWA Local Planner

DWA 算法基本思路如下：

1.在机器人控制空间进行速度离散采样(dx,dy,dtheta)。

2.对每一个采样速度执行前向模拟，看看使用该采样速度移动一小段段时间后会发生什么。

3.评价前向模拟中每个轨迹，评价准则如: 靠近障碍物，靠近目标，贴近全局路径和速度；丢弃非法轨迹（如哪些靠近障碍物的轨迹）。

4.挑出得分最高的轨迹并发送相应速度给移动底座。

5.重复上面步骤。

3.4 轨迹跟踪

3.4.1 PID 控制

PID 控制小车轨迹跟踪的基本思想就是找到小车和目标轨迹的偏差量，然后使用 PID 控制器来消除偏差，当小车偏离轨迹就会有偏差，小车在目标轨迹上，偏差就为零。使用 PID 控制器对小车进行横向控制，使小车左右转向来消除偏差，从而完成轨迹跟踪。小车纵向控制由路径曲率决定，曲率越大，速度越慢，曲率越小，速度越快，从而实现直道加速，弯道减速。

PID(Proportional Integral Derivative)控制是最早发展起来的控制策略之一，由于其算法简单、鲁棒性好和可靠性高，被广泛应用于工业过程控制，尤其适用于可建立精确数学模型的确定性控制系统。

在工程实际中，应用最为广泛的调节器控制规律为比例、积分、微分控制，简称 PID 控制，又称 PID 调节，它实际上是一种算法。PID 控制器问世至今已有近 70 年历史，它以其结构简单、稳定性好、工作可靠、调整方便而成为工业控制的主要技术之一。当被控对象的结构和参数不能完全掌握，或得不到精确的数学模型时，控制理论的其它技术难以采用时，系统控制器的结构和参数必须依靠经验和现场调试来确定，这时应用 PID 控制技术最为方便。即当我们不完全了解一个系统和被控对象，或不能通过有效的测量手段来获得系统参数时，最适合用 PID 控制技术。PID 控制，实际中也有 PI 和 PD 控制。PID 控制器就是根据系统的误差，利用比例、积分、微分计算出控制量进行控制的。

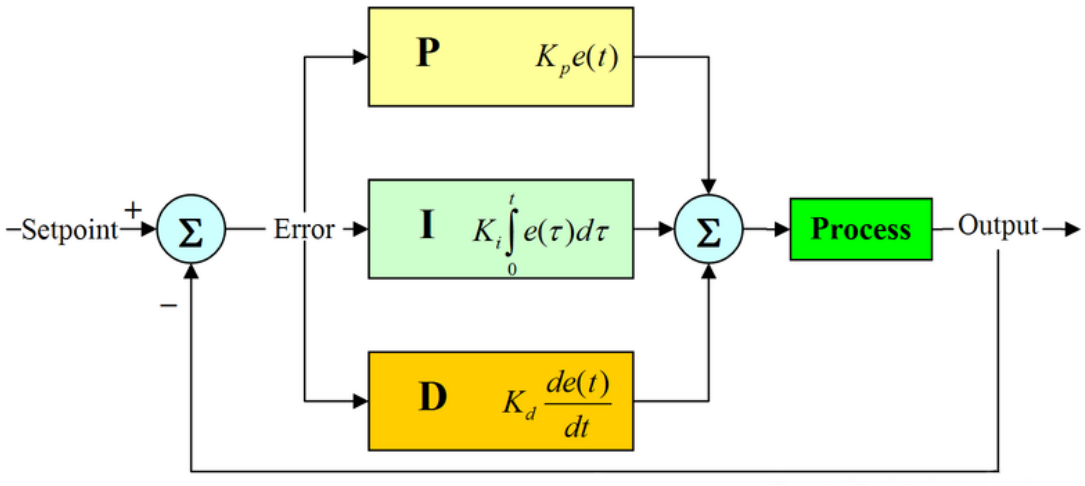

图 3-7 PID 原理图

3.4.2 Pure Pursuit

纯路径跟踪是基于几何追踪的一种方法，该方法只考虑车辆运动学方程。首先将车辆模型简化，如同 3-8 所示：

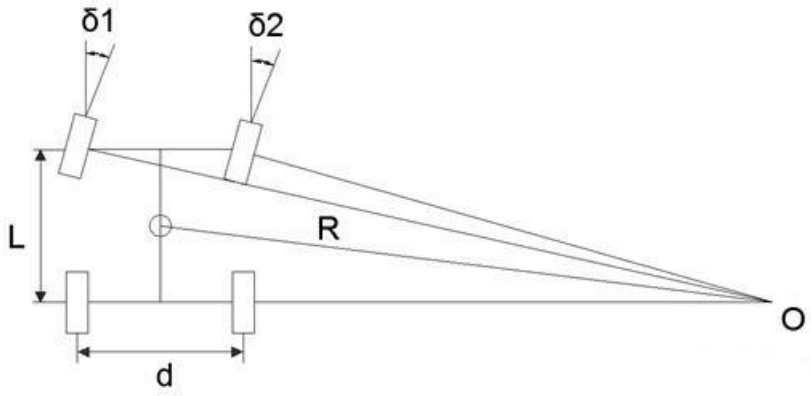

图 3-8 阿克曼转向几何

低速时（如在停车场的操控）轮胎不需要产生侧向力，在这种情况下，轮胎滚动没有侧偏角，此时车辆必须如下图所示进行转向。对转向时（假定小转向角）的正确几何关系，可以由几何关系得到车辆的正确转向角。为了方便描述预瞄点和车辆转向半径的关系，再将车辆简化为自行车模型。

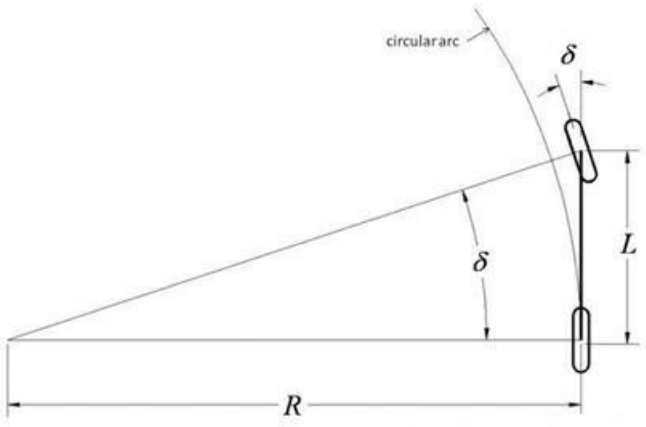

图 3-9 转向图

加上预瞄点之后，根据几何关系，得到车辆要跟踪预瞄点因该输出的正确转角，只要车辆输出改角度，即可保证后轴中心经过预瞄点，从而保证车辆跟踪路径行驶。

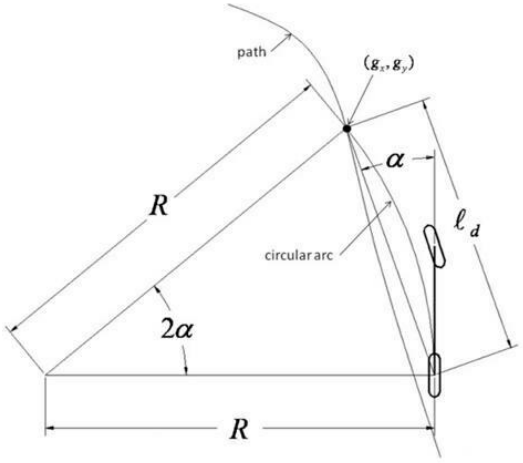

图 3-10 具体几何原理

最终可得到前轮转角和前视距离的线性关系，所以，本质上，纯路径跟踪就是一个 P 控制器。具体运用到差速车上的话，原理也比较简单，不再赘述。

3.4.3 Stanley 算法

Stanley 算法也是一种直接对前轮转角进行调整来消除横向偏差的横向控制算法。其控制原理可以用如下公式来表示：$\delta=\theta+tan-1(ke/v)$

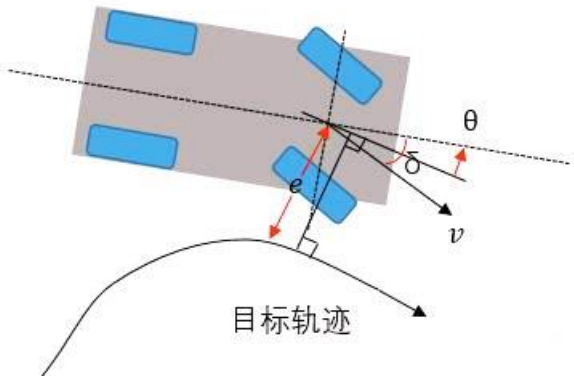

图 3-11 控制原理图

δ 表示前轮转角，v 表示车速，θ 为最近的目标轨迹点的切线与速度方向的

夹角，e 为横向误差，k 为可调节的控制增益量。从上式可看出当航向偏差量 θ θθ 与横向偏差量 e 增加时，控制算法也会相应增大对前轮转角的调节，从而更快地消除较大的误差。本算法源于 2005 届 DARPA 挑战赛中的冠军车辆，Stanley robot 。在该次比赛中，第一次有无人驾驶车辆完成了整个越野赛道。而在所有跑完全程的五支队伍中，由 Sebastian 带领的斯坦福大学 AI 实验室凭借 Stanley 算法的出色发挥，以压倒性的优势摘得桂冠。

第 4 章 项目实施过程

4.1 小车启动流程

本方案的实现流程图如图 4-1 所示，具体步骤如下：

（1） source gazebo_test_ws/devel/setup.bash,

（2） roslaunch gazebo_pkg race.launch,启动 gazebo 仿真世界，加载小车模型；

（3） roslaunch race_navigation cartographer_demo.launch,启动 cartographer 建图；

（4） rosrun teleop_twist_keyboard teleop_twist_keyboard.py，启动键盘控制节点；

（5） rosrun map-sever map-saver -occ 70 -free 30 -f racemap 保存地图

（6） 在 start_game 中启动终端键入 python2 main.py 加载仿真环境和小车模型，并启动导航功能，发布目标点进行导航。

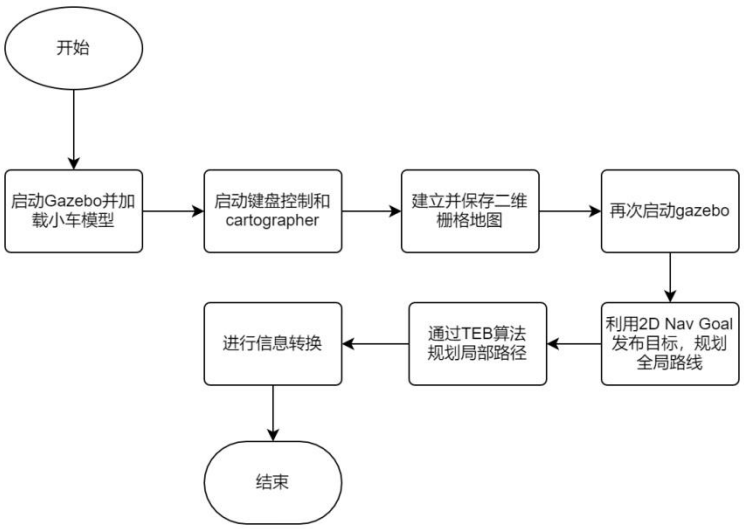

图 4-1 方案流程图

首先，需要在 Gazebo 中加载车模，打开 rviz，订阅小车模型和传感器数据。然后运行 Cartographer 启动文件，运行键盘控制节点，控制小车走完全程，rviz 中可以显示相应的二维栅格地图，看到地图完整地建立之后，运行保存地图的命令。将地图放到功能包里相应的文件夹，将 rviz 配保存在相应的文件夹。然后在

launch 文件中添加相应的启动命令即可完成建图任务下载相应的导航功能包，在工作空间下创建新的功能包，编写导航的 launch 启动文件，加载相应的参数文件。配置成功后即可实现导航，基础的流程不再赘述。

4.2 基于 AMCL 的定位

当小车可以实现基础的导航功能后，需要仔细优化各个功能模块，使得相应的功能更加符合自己的小车。首先优化定位功能。在 AMCL 包中，选型、配置了运动模型、传感器模型、蒙特卡洛航位推算算法、采样算法等等功能模块。传感器模型将传感器数据结构化，运动模型将机器人的运动进行分解和代数化。传感器模型和运动模型相辅相成，紧密合作，为蒙特卡洛滤波做好数据支撑。采样/重采样算法为定位的迭代运算和误差消除提供保障。输入给 AMCL 的有/tf、/scan、/map，其中输入的 tf 有 base_frame 到 odem_frame，输出的 tf 有 base_frame 到 map_frame。

表 4-1 AMCL 订阅与发布的话题

	名称	描述
订阅	scan	激光雷达扫描数据
	tf	tf 变换
	initialpose	用于初始化粒子滤波器的平均值和协方差
	map	当设置了 use_map_topic 参数时， AMCL 订阅此主题以检索用于基于激光的本地化的映射
发布	particlecloud	由过滤器维护的姿态估计集合
	tf	从 odom 发布的 tf 变换

在 amcl.launch 文件里可以修改配置参数，从而优化定位功能。

4.3 导航

Navigation 包是一个获取里程计信息、传感器数据和目标位姿并输出安全

的速度命令到运动平台的 2D 导航包的集合。

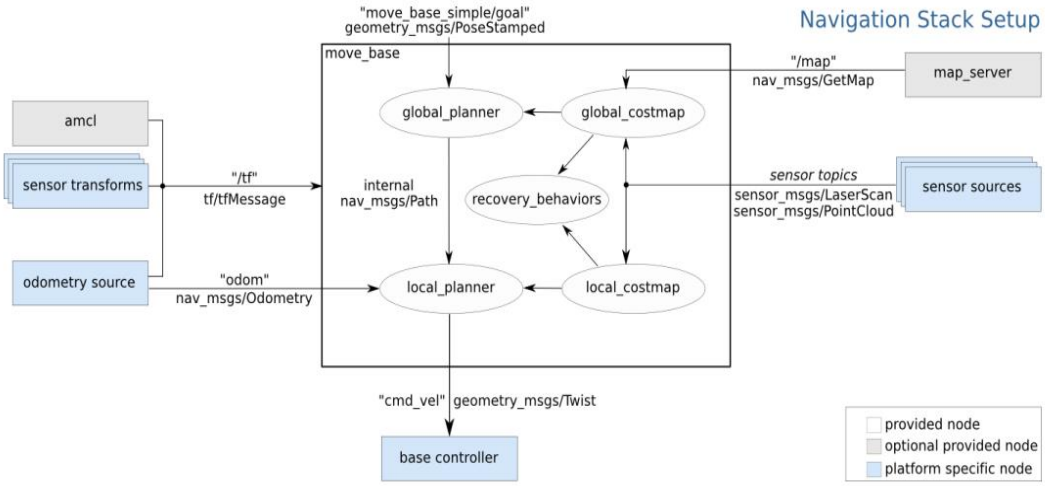

图 4-2 导航框架图

导航总的来看可以分为数据收集层（传感器数据收集）、全局规划层（global_planner）、局部规划层（local_planner）、行为层（结合机器人状态和上层指令给出机器人当前行为）、控制器层。

车模使用 navigation 导航时，move_base 这个模块负责整个 navigation 行为的调度, 包括初始化 costmap 与 planner，监视导航状态适时更换导航策略等。move_base 订阅 map_sever 发布的二维栅格地图，订阅 TF 变换得到车模在地图上的位姿。在 move_base 的使用中定义了 planner_costmap_ros 、controller_costmap_ros 分别对应 global_costmap、local_costmap。

具体的逻辑流程如下：

（1）move_base 首先启动了 global_planner 和 local_planner 两个规划器，负责全局路径规划和局部路径规划，通过 costmap 组件生成自己的代价地图(global_costmap 和 local_costmap)；

（2）通过 Dijkstra 全局路径规划算法，计算出车模从起始点到 2D NavGoal 发布的目标点的全局路线；

（3）通过 TEB 算法，规划出局部避障的路径，并将相应的速度和转向信息发布给运动控制器。导航功能节点图如下图所示：

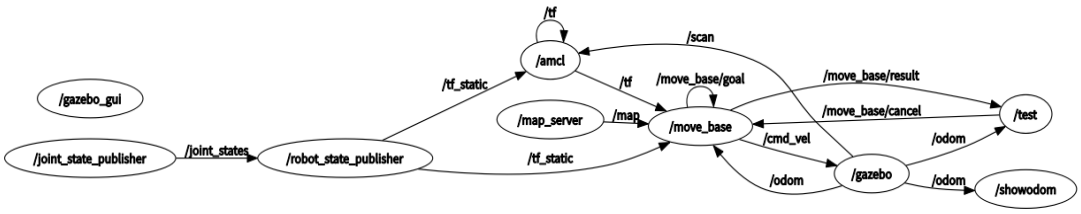

图 4-3 导航节点图

4.3.1 **Dijkstra 的全局路径规划**

为全局代价地图配置静态地图层和膨胀层，全局路径规划器使用全局代价地图产生全局路径。

表 4-2 global_planner 重要参数表

名称	描述
old_navfn_behavior	若在某些情况下,想让 global_planner 完全复制 navfn 的功能,那就设置为 true
use_quadratic	设置为 true,将使用二次函数近似函数,否则使用更加简单的计算方式,这样节省硬件计算资源
use_dijkstra	设置为 true,将使用 dijkstra 算法,否则使用 A*算法
use_grid_path	如果设置为 true,则会规划一条沿着网格边界的路径,偏向于直线穿越网格,否则将使用梯度下降算法,路径更为光滑点
allow_unknown	是否允许规划器规划穿过未知区域的路径,只设计该参数为 true 还不行,还要在 costmap_commons_params.yaml 中设置 track_unknown_space 参数也为 true 才行

如图 4-4 所示为使用 Dijkstra 算法规划的全局路径和全局代价地图。

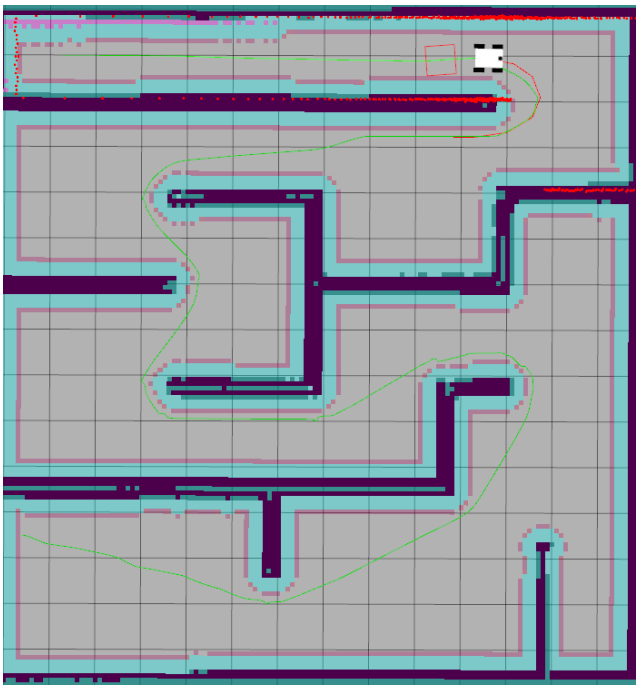

图 4-4 基于 Dijkstra 的全局路径

4.3.2 基于 TEB 的局部路径规划

为局部代价地图配置障碍地图层和膨胀层，局部路径规划器使用局部代价地图进行动态避障。

表 4-3 局部路径规划重要参数

名称	描述
odom_topic	机器人原始 odom 坐标系
teb_autosize	优化期间允许改变轨迹的时域长度
dt_ref	局部路径规划的解析度一般为 1/control_rate
max_vel_x	最大移动速度
max_vel_x_backwards	后退时最大线速度
max_vel_theta	最大角速度
acc_lim_x	线加速度
acc_lim_theta	角加速度

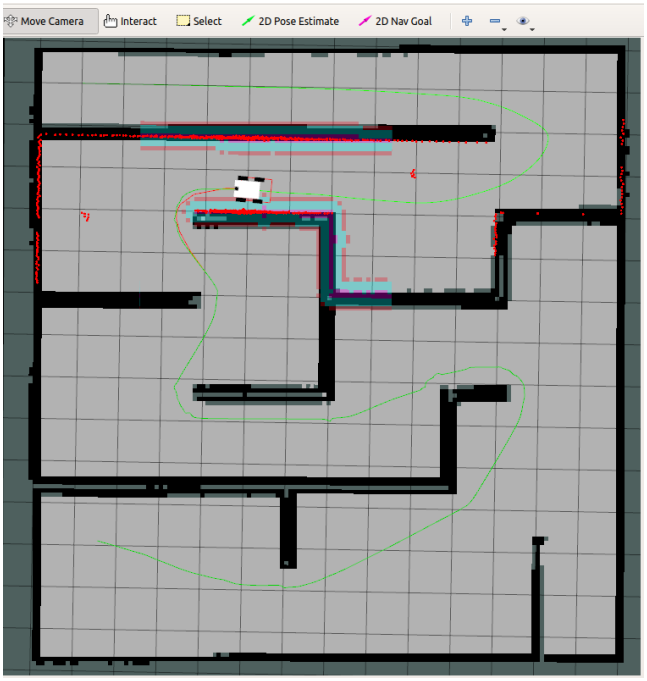

图 4-5 基于 TEB 的局部路径规划

整体的 TF 树如下图所示：

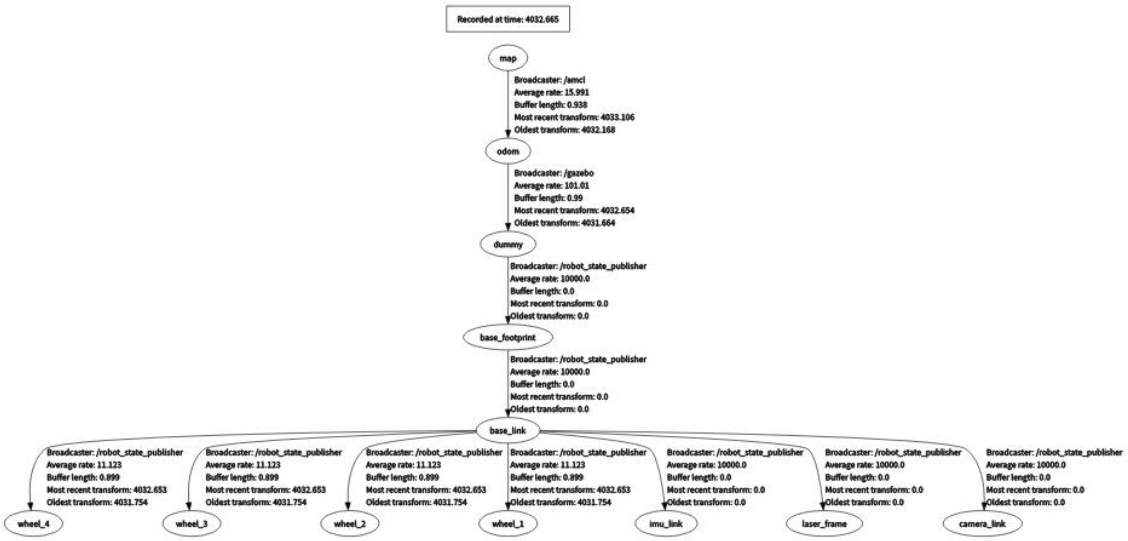

图 4-6 TF 树

4.3.3 基于 PID 的路径跟踪算法

在训练版本的小车中，只使用 DWA 或者 TEB 发布的速度指令很难跑出很好的成绩，于是采用自行编写路径跟随算法。通过分析纯路径跟踪和 Stanley 的理论知识，发现他们都是基于几何路径追踪，并没有参考汽车的动力学模型。于是最终决定使用最为熟悉、最简单的 PID 控制算法来控制小车的横向误差。

具体实施方法为：订阅 teb 发布的局部路径，该话题实质上发布的是一系列坐标点。不过该坐标点是在 odom 坐标系下的，需要将其转换到小车的 base_link 坐标系下以方便计算小车中线与路径的偏差。

在这儿说明一下路径中的目标点从里程计坐标系 x_oOy_o 变换到小车坐标系 $x_{\mathrm{b}}Oy_{\mathrm{b}}$ 的具体方法：已知小车相对于里程计坐标系 (odom) 的位姿坐标(x_1, y_1, θ)，目标点相对于里程计坐标系的位置坐标(x_2, y_2)。先将目标点的位置通过平移转换到坐标系 $x_{\mathrm{c}}Oy_{\mathrm{c}}$ 下：

$$\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}_c = \begin{bmatrix}x_2\\ y_2\end{bmatrix}_e - \begin{bmatrix}x_1\\ y_1\end{bmatrix}_e$$

再将$(\Delta x, \Delta y)_c$的坐标旋转 θ 角度就可以将目标点变换到小车坐标系下了。

$$\begin{bmatrix}x\\ y\end{bmatrix}_{\mathrm{b}} = \begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}$$

之后可根据偏差编写方向环。方向 PID 使用 PD 即可。小车纵向控制使用一个简单的 P 控制即可。

不过该方案在训练版车模上效果非常好，可轻松进入 20s。但由于考核版车模加大了惯性，导致车模加减速非常不灵敏，用该算法控制起来非常困难，甚至成绩还不如直接使用优化后的 teb 发布的速度指令。因此本方案最终没有使用。

4.4 问题及解决方案

（1）Gampping 构建地图时会出现构建的地图与实际的地形严重不符的情况，虽然可以使用 PS 等工具修改地图，但为了方便起见，本方案最终使用 Cartographer 建图。

（2）使用全局路径规划时发现近距离可以规划路径，距离太远就规划不出来路径，后在全局代价地图参数文件里加上全局地图的长、宽，并设置稍微大一

点即可解决问题。

（3）关于局部路径规划比如车子停下后退转弯的问题，在之后的调节局部路径规划（TEB）的参数关于角速度和倒车惩罚系数有所改善。

（4）由于地图 u 型弯较多，尝试修改最小转向半径及其权重使得 u 型弯过弯更加流畅。调整膨胀层系数及其权重、与障碍物的最小期望距离、障碍物周围缓冲区等参数使得车子尽可能在过弯时不停下调整角度。

（5）最初地图的路径规划并不是很理想，由此调整了 global_planner 中与代价值相关的参数和代价因子，这让每次的路径趋于一致。

（6）第一个 u 型弯前有较长的加速直道，据此调整了最大加速度和代价地图的刷新发布频率让小车更早的减速，对过弯后的障碍物进行规划避障。

（7）通过调整基于时间轨迹上的权重让在稳定和时间上平衡，使得小车尽可能规划时间更快的路程。

第 5 章 项目数据分析

5.1 分析工具的使用

5.1.1 PlotJuggler

PlotJuggler 是一个基于 Qt 的应用程序，允许用户加载，搜索和绘图数据。提供了更加友好的用户界面。由于 plot 并不兼容 python2，但是为了更好的分析小车运行的状态我们安装了 PlotJuggler。

DataStreaming plugins: 订阅到一个或多个 ROS 主题，并绘制它们的数据流。RosPublisher plugin: 使用交互式跟踪重新发布原来的 ROS 消息。所以用来分析小车在运动过程中数据传输的稳定，以及分析速度和转弯的稳定性。

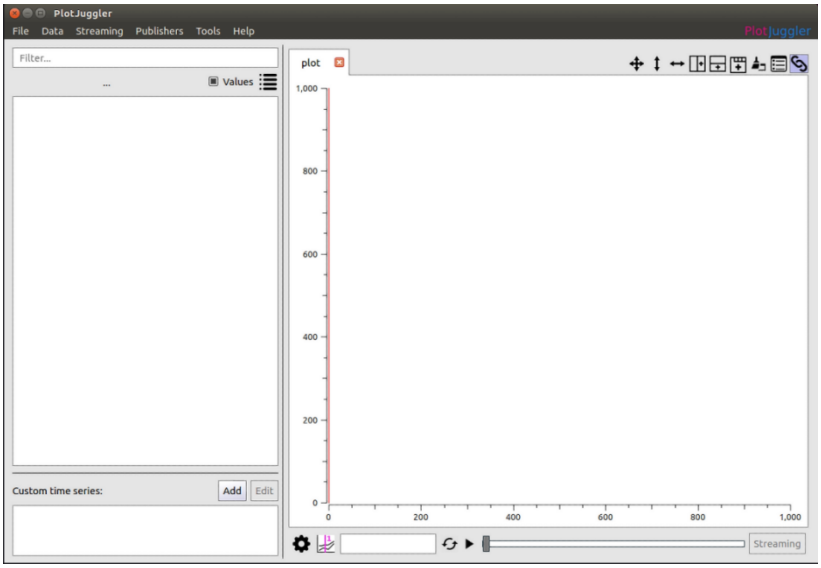

图 5-1 PlotJuggler

5.1.2 rqt 工具

rqt 里面有很多十分有用的用于 ROS 调试的插件

rqt_graph：用来查看节点是否连接，以及是否有需要禁用的节点，防止违反比赛规则。

rqt_robot_steering:可以发布一个话题 cmd_vel,发布 Twist 话题消息，可以可视化的修改速度，转角变量，用于测试一些控制指令十分方便。

rqt_tf_tree:用于可视化 ROS TF 框架树，在判断是否正确配置运动控制器以及 amcl 时候十分有用。

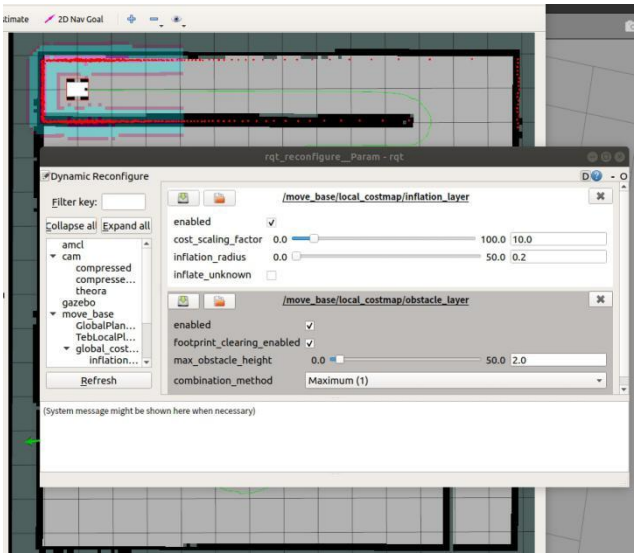

图 5-2 rqt 动态调参

5.2 数据分析

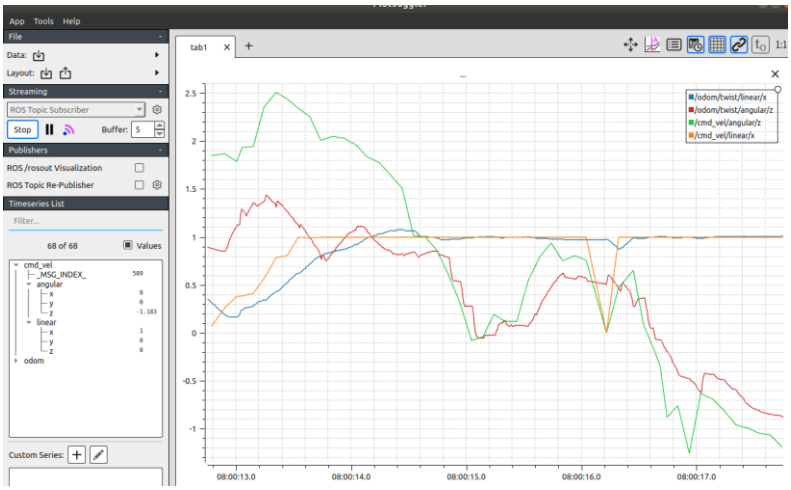

图 5-3 PlotJuggler 速度曲线

在这张图中我们可以看到小车由于惯性加大，小车的响应非常延迟。基本上不能对 cmd_vel 发布的速度做到很好的追踪。因此要对小车进行精确的控制非常的困难。一般的轨迹跟踪算法，如 Pure Pursuit，Stanley，PID，等基于几何追踪的算法计算出来的小车转角基本上不能直接用，还需要加入增益控制。也有加入车辆动力学方程的模型预测控制（MPC）算法效果可能会好一点。不过由于时间问题，本方案并没有尝试，而是直接选用 teb 发布的控制指令进行轨迹跟踪。

在分析小车轨迹与规划轨迹是否贴合，即分析小车跟随情况时可以编写一

个脚本订阅小车的位姿再发布为路径信息即可在 rviz 中显示，以方便分析。

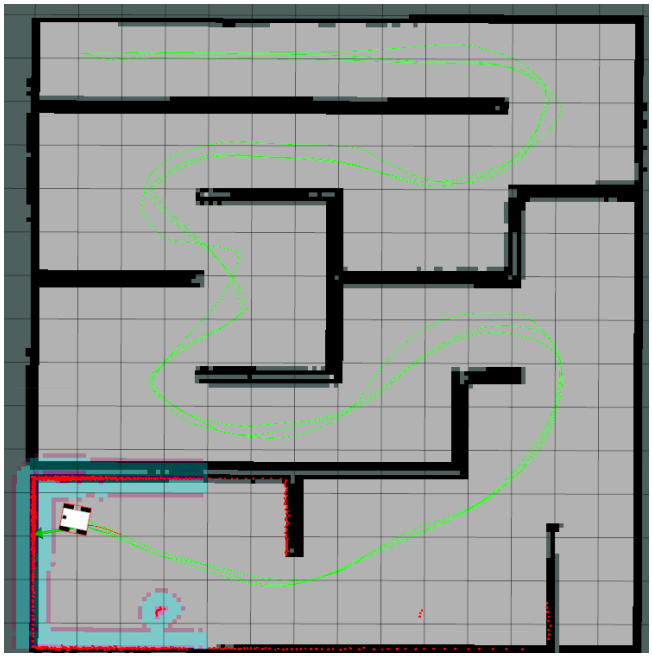

图 5-4 小车行驶轨迹

第 6 章 项目作品总结

通过这两个月来对 ros 的学习，我们对 ros 系统有一定的了解和操作能力，在学习时遇到了很多困难，很多时侯自己一人的能力是无法解决这些问题，而队友恰恰可以给你帮助，通过这个比赛，我认识到了团队的重要性，学会了相互之间的配合。我们凭着团队合作坚持不懈地去思考、学习、调试，最终将问题一个个解决。本届比赛，对路径规划算法要求较高，对小车在尽量大的速度上同时不发生飘移，撞墙的控制上也有较大的要求。在最初，我们选用了 dwa 算法，但通过一段时间的调试，发现与小车不太相匹配，时间上也没有达到我们预期的效果。之后我们尝试用了自己写的算法，虽然最终的效果并不是很满意，但我们从中学到了很多东西，让我们对算法有了进一步的了解。最后我们选择了 TEB 算法，经过调试，大大缩短了完赛时间。尽管还没有达到我们预期的目标，但在短短一周的时间内优化到了极致，在这一周短暂的时间里,极大的提高了我们的能力，锻炼了我们的意志。

参考文献

[1] 谭民, 王硕. 机器人技术研究进展[J]. 自动化学报, 2013, 39(07):963~972.

[2] Li Y, Olson E B. Extracting general-purpose features from LIDAR data[C]//Proccedings of 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA:IEEE, 2010:1388~1393.

[3] Kohlbrecher S, von Stryk O, Meyer J, et al. A flexible and scalable SLAM system with full 3D motion estimation[C]//Proccedings of 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, Kyoto, Japan:IEEE, 2011:155~160.

[4] Pedrosa E, Lau N, Pereira A. Online SLAM Based on a Fast Scan-Matching Algorithm[C]//Proccedings of Portuguese Conference on Artificial IntelligenceSpinger, 2013:295~306.

[5] Hess W, Kohler D, Rapp H, et al. Real-time loop closure in 2D LIDAR SLAM[C]//Proccedings of 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden:IEEE, 2016:1271~1278.

[6] 杨明, 董斌, 王宏, 等. 基于激光雷达的移动机器人实时位姿估计方法研究[J]. 自动化学报, 2004,(05):679~687.

[7] 畅春华, 赵汗青, 秦博. 基于激光雷达的移动机器人实时位姿估计算法[J]. 装甲兵工程学院学报, 2011, 25(04):54~57.

[8] 史风栋, 刘文皓, 汪鑫, 等. 室内激光雷达导航系统设计[J]. 红外与激光工程, 2015, 44(12):3570~3575.

[9] 内蒙古科技大学.第十五届全国大学生智能汽车竞赛室外光电组技术报告.2020.8.6.