无人驾驶竞速车虚拟仿真赛

技 术 报 告

学　　校：　　中国矿业大学（北京）

队伍名称：　　矿世竞速车队

参赛队员：　邹宇杰　王雪蓓　林锐东

带队教师：　　赵峰

目　录

第 1 章 方案概述. 1
第 2 章 问题描述. 4
第 3 章 技术方案. 5
第 4 章 方案实现. 11
第 5 章 测试分析. 14
第 6 章 作品总结. 17
参考文献. 18

第1章 方案概述

在机器人操作系统（ROS）下，对 Lidar 和 IMU 姿态传感器获取的智能车周围环境信息与车体的姿态信息进行处理，再通过 SLAM 算法构建环境地图，使用 AMCL 算法进行定位，借助 ROS 中 move_base 功能包，实现路径规划和自主导航。

1.1 技术实现路线

本方案在 racecar 模型基础上通过在 gazebo 仿真环境中配置 gmapping 以实现 slam 建图，并配置 move_base + teb 路径规划的导航栈以实现导航。

1.1.1 仿真环境

Gazebo 是一款 3D 动态模拟器，能够在复杂的室内和室外环境中准确有效地模拟机器人群。与游戏引擎提供高保真度的视觉模拟类似，Gazebo 提供高保真度的物理模拟，其提供一整套传感器模型，以及对用户和程序非常友好的交互方式。

本方案使用赛方统一提供的传感器、机器人、地图场景文件。

1.1.2 建立栅格化地图

Gmapping 是一种主要使用激光和里程计数据创建栅格地图的一种 slam 建图方法。其采用自适应重采样技术来减少粒子退化的影响，同时在粒子分布时引入当前　观测值，使得粒子估计的不确定性降低。[1]

本例中使用 rf2o 与车轮里程计信息以实现实时结算机器人当前的姿态和位置。

1.1.3 navStack 配置

本例中通过配置实现了 tf 转换广播，使用 imu、lidar、camera 与车轮里程计提供里程信息，借助 ekf_pose 实现 rf2o 融合 imu_data 提供 odom 数据。使用 map_sever 提供地图并以 move_base 作为基本控制器，实现了导航栈的配置。

1.1.4 路径规划

Teb 是一种实现了用于移动机器人导航和控制的在线最佳本地轨迹规划器。其对由全局计划程序生成的初始轨迹在运行时进行了优化，从而将轨迹执行时间（时　间最优目标）最小化。通过解决稀疏的标量化多目标优化问题，可以有效地获得最优　轨迹。

本例中通过改变 teb 规划参数及修改优化权重，以解决目标冲突时的路线规划冲突问题实现了更优的路径规划。

1.2 创新点及效果

1.2.1 选择性使用 IMU 数据构建 ekf 融合里程计

Gazebo 中运行的 IMU 所提供的 imu_data 有较大的噪声干扰，直接使用 ekf 融合 lidar 与 imu 数据时会有较大的误差，经对比测试，选取姿态角中的偏航角（yaw） 参与融合，取得了较好的效果。

1.2.2 采用 teb 路径规划取代传统 DWA 方法

teb(Timed Elastic Band) 规划器可以处理阿克曼底盘车辆的运动学约束，同时在最大速度和加速度等参数约束的条件下，规划出时间最短路径和速度指令。路径　上可以表现为明显的切弯、靠近障碍物；速度指令上将会表现为速度和方向指令的快速振荡。

与 dwa 相比，在运行时，teb 可以实现获得制动最小化轨迹（时间最优目标），与障碍物分离，并遵守满足最大速度和加速度的动力学约束。

第2章 问题描述

2.1 关键性问题

Racacar 系列项目一般基于 ROS 系统，使用四轮车辆型机器人（car-like-robot），并利用相机、激光雷达等传感器，完成建图、巡航任务。这与比赛要求基本相同。

同时，比赛环境与车辆使用等因素的差异也带来了新的问题，如：订阅信息与控 制代码从实际车辆到 gazebo 的迁移；激光雷达、IMU 等虚拟仪器的使用与控制；根据小车的几何尺寸、舵机性能等信息完成参数调试等。

2.2 相关研究

本次竞赛项目的任务可描述为：在 gazebo 中，利用四轮小车，完成建图、以尽可能快的速度完成巡航与避障。它可以看作是 racecar 项目在 gazebo 中的拓展应用。

麻省理工学院最早开展了 racecar 项目，并将其作为学生的课程实践素材，目前已成为经典的 ROS 实战入门项目。随后诞生了诸多类似项目，如成本较低的 hypha-

racecar、使用普通摄像头与深度学习的 Daniel Tobias’CAR 等。这些项目大都已经开源，在任务的不同环节提供了多种可行方案。

2.3 算法实现方案

针对车辆型机器人存在最小转向半径、由前轮转向改变运动方向等特点，选择阿克曼转向几何作为车辆控制模型。导航功能的配置围绕 move_base 进行，使用 rf2o 作为车辆的里程计，并配置 amcl 以消除误差。

运行车辆，利用键盘控制结合 gmapping 建立地图，作为 move_base 进行全局规划的参照。move_base 的全局航迹控制与 teb 规划器的局部航迹控制相结合，发布 twist 类型信息，经转换脚本，成为阿克曼模型下的控制信息，指导车辆运动。最后针对 gazebo 环境下的运行情况进行参数调试。

第3章 技术方案

高质量的建立地图、对机器人位姿的精确估计、合适的路径规划与移动控制算法是实现高效导航的基础。本方案选择经典的激光雷达 gmapping 方法建立栅格化地图，以 move_base 为基础，使用 amcl 与 rf2o_odometry 提供导航时的位姿估计。

3.1 gmapping 建图

Gmapping 包订阅由 gazebo 发布的 \tf 与 \scan 话题实现激光雷达、基坐标系、里程计坐标系之间的转换与扫描点信息收集，同时发布 \map_metadata 与 \map 话题发布地图的 Meta 数据与栅格数据。

3.2 amcl

AMCL(adaptive Monte Carlo Localization，自适应蒙特卡洛定位)，是一种基于自适应蒙特卡洛算法实现对机器人在二维移动过程中的概率定位的系统，源于 MCL 算法的改进。其采用粒子滤波器来跟踪已经知道的地图中机器人位姿，对于大范围 的局部定位问题工作良好。

Amcl 结点将传入的激光扫描转换为测距框架，即在导航过程中估计基本框架（base_frame）相对于全局框架（global_frame）的变换，并发布全局框架和测距框架（odom_frame）之间的变换，如图 1 所示。

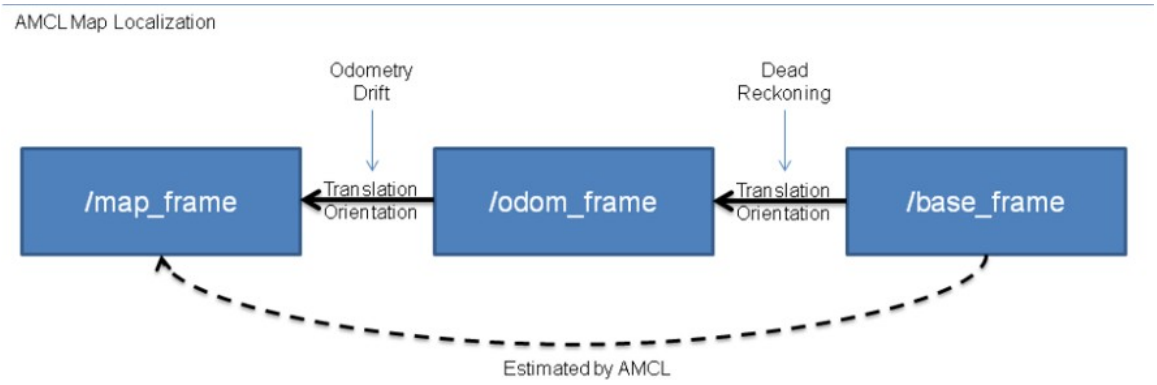

图 1: amcl 的变换作用

3.3 rf2o 激光雷达里程计

RF2O 是一种快速而精确的方法，用于从连续范围扫描中估计激光雷达的面运动，如图 2 所示。对于每个扫描点，可以根据传感器速度制定范围流约束方程，并最小化所得几何约束的鲁棒函数以获得运动估计。与传统方法相反，该方法不搜索对应关系，而是以密集的 3D 视觉测距法的方式基于扫描梯度执行密集扫描对准。最小化问题以粗到精方案解决以应对大位移，并且基于估计的协方差的平滑滤波器用 于处理无约束情景（例如走廊）中的不确定性。

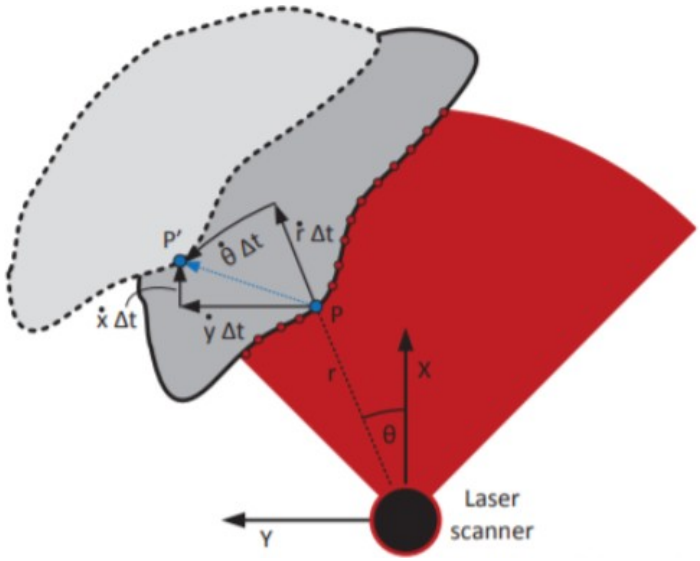

图 2: rf2o 工作原理

rf2o_laser_odometry 节点发布平面测距估计用于从车载雷达的 2D 扫描激光器的移动机器人。它首先估计激光雷达设备的里程，然后通过使用 tf 转换计算机器人基础里程。并与 amcl 结合实现上图 1 中 map—odom—base_link 坐标系之间的转换。

3.4 teb 局部规划器

3.4.1 teb 规划器概述

TEB 将机器人的一系列位姿轨迹模型抽象成带有时间信息的弹性带模型，起始点、目标点状态由用户 \ 全局规划器指定，中间插入 N 个 eletic band（橡皮筋）形状的控制点（机器人姿态），机器人的第 i 个位姿状态可以表示为 Xi = (xi, yi, βi)T ，位姿包含位置信息 xi,yi 和方向角 βi δTi 是机器人位姿 Xi, X(i + 1) 之间的过渡时间 间隔，则有如图 3 所示的位姿关系。[3]

定义事件间隔序列与位姿序列分别为:

$$\tau = \{\Delta T_i\} \quad i = 0, 1, 2 \ldots n-1, n \in \mathrm{N} \qquad (1)$$

$$Q = \{X_i\} \quad i = 0, 1, 2 \ldots n, n \in \mathrm{N} \qquad (2)$$

对上述两式合并定义状态集 B (Q, τ)

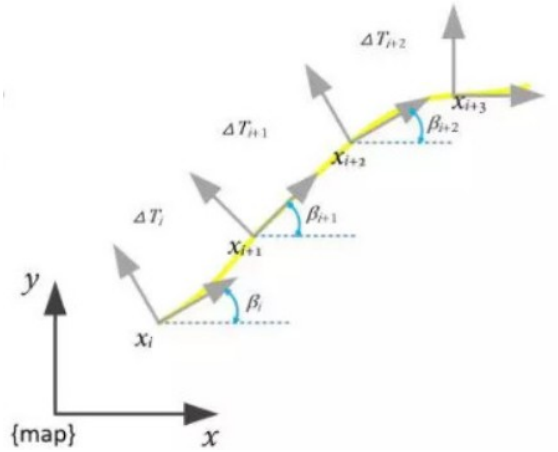

图 3: 不同时间点的位移

TEB 通过加权多目标优化求解出一系列带有时间间隔的机器人位姿序列组成最优的路径点状态集，通过调整位姿和时间间隔来优化 TEB，使用权重求和模型求解多目标优化问题[4]:

$$f(B) = \sum_k \gamma_k f_k(B) \qquad (3)$$

$$B^* = \underset{B}{\operatorname{argmin}} f(B) \qquad (4)$$

其中 B^*为最优结果，f（B）为响应约束的目标函数，γ_k 为约束的目标函数权值。

3.4.2 约束目标函数

跟随路径与避障约束 Teb 约束主要有两个目标：跟随已知的全局规划路径和避障。为保证局部路径规划一定程度上遵循全局路径规划，跟随路径施力将 eb 拉向全局路径； 为了满足机器人的避障能力，避障约束施力使其远离障碍物，以惩罚函数的形式实现约束施力，如图 4 所示。

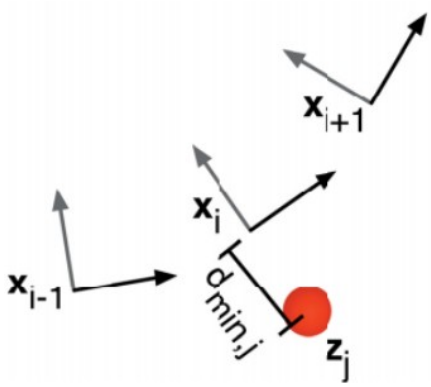

图 4: teb 避障原理

图中 z_j 为第 i 个障碍物，路径中姿态 x_i 与障碍物 z_j 的最近距离记为 $d_{(min,j)}$，有：

$$f_{\text{path}} = e_{\Gamma}\left(d_{\min,j}, r_{\text{pmax}}, \epsilon, S, n\right) \tag{5}$$

$$f_{\text{obstacle}} = e_{\Gamma}\left(-d_{\min,j}, -r_{\text{omin}}, \epsilon, S, n\right) \tag{6}$$

速度与加速度约束 机器人速度和加速度的动态约束通过类似的惩罚函数来描述几何约束。平均平移和旋转速度分别根据欧式距离和方向角的改变量计算，在两个连续的位姿 $x_i, x_{(i+1)}$ 和两个姿势之间的过渡的时间间隔 ΔT_i 。

线速度：

$$v_i \cong \frac{1}{\Delta T_i}\begin{pmatrix} x_{i+1} - x_i \\ y_{i+1} - y_i \end{pmatrix} \tag{7}$$

角速度：

$$\omega_i \cong \left(\beta_{i+1} - \beta_i\right) / \Delta T_i \tag{8}$$

线加速度：

$$a_i \cong^{2(v_{i+1} - v_i)} / \left(\Delta T_i + \Delta T_{i+1}\right) \tag{9}$$

角加速度：

$$\alpha_i \cong^{2(\omega_{i+1} - \omega_i)} / \left(\Delta T_{i+1} + \Delta T_i\right) \tag{10}$$

最快路径约束 最快时间是考虑 TEB 全局性，因为目标函数需要依赖于所有的参数。是通过最小化所有时间间隔之和的平方要求机器人尽可能快地到达规划的终点。最短时间约束函数可表示为：

$$f = \left(\sum_{i=0}^{n} \Delta T_i \right)^2, i \epsilon \mathbb{N} \tag{11}$$

3.5 参数优化

Teb 规划器提供了近百个参数接口实现对约束函数权重的调节，teb 以高于地图与控制刷新频率的速度进行着位姿的重复计算，而经过验证，对路径规划产生影响 较大的为以下几类参数：

3.5.1 机器人性能描述参数

性能描述参数如表 1 所示。应尽可能贴近车辆真实性能以获得最优性能，当所选用参数过低时会对性能发挥造成限制，当然再某些情况下可作为对路径规划的附加 控制因子；而当参数过高时，则会明显产生不正确的规划。

表 1: **性能描述参数表**

max_vel_x	最大 x 前向速度
max_vel_x_backwards	最大 x 前后速度
max_vel_theta	最大转向角速度
acc_lim_x	最大 x 加速度
acc_lim_theta	最大角加速度
min_turning_radius	车类最小转弯半径
wheelbase	驱动与转向轴距

表 2: **优化权重参数表**

weight_max_vel_x	满足最大允许平移速度的优化权重
weight_max_vel_theta	满足最大允许角速度的优化权重
weight_acc_lim_x	满足最大允许平移加速度的优化权重
weight_acc_lim_theta	满足最大允许角加速度的优化权重
weight_kinematics_turning_radius	实现最小转弯半径的优化权重
weight_viapoint	和全局路径采样点距离的权重
weight_shortest_path	驱动与转向轴距
weight_obstacle	优化权重以保持与障碍物的最小距离
weight_inflation	通胀惩罚的优化权重（应该很小）
weight_dynamic_obstacle	动态障碍权重
weight_dynamic_obstacle_inflation	动态障碍物膨胀区的权重

3.5.2 优化权重

Teb 配置中提供了 10 余个不同的优化参数，如表 2 所示。此类参数将影响在路径规划过程中对其他参数的参考程度，具体反应为将多大程度上接近设定的参数。

3.5.3 障碍代价

提供了多个参数以调节障碍物在代价地图上的膨胀范围与权重。

值得指出的是,障碍代价参数应配合 footprint 模型使用,当使用 Line 模型时配合 min_obstacle_dist 参数使用可以获得能较好包裹机器人的膨胀空间，而当使用能符合车辆实际外形的 polygon 模型时则考虑不使用 min_obstacle_dist 以实现最佳路线。

3.5.4 全局规划参考

Teb 为了实现跟随路径约束 teb 使用 f_{path} 函数将规划路径拉向全局规划，并提供了多个参数为实现接近程度的调节，例如全局规划取点数目、间隔、规划权重等。此类参数的使用应结合车辆期望速度调节，若速度过快而跟随约束过大，则路径规划时对障碍的考虑即路径更新可能无法满足避障需求。

值得一提的是在 teb 中还提供了可以影响 teb 对车辆控制精度的参数 dt_ref，该参数本意为最优相邻树上相邻姿态 x 之间的最优距离，在实际使用时反映为对车辆控制精度的限制。

3.6 运动控制

3.6.1 move_base 控制原理

控制过程中使用 move_base 结点进行决策控制提供 action 通信机制，当通过 rviz 划定全局导航目标点后，将调用 Astar 全局路径规划器生成全局路径，发布全局路径给局部路径进行路径规划，并下发 cmd 控制指令给机器人底盘。

3.6.2 twist-ackermann 消息转换

Move_base 提供的 base_local_planner 实现的导航规划与移动控制不适应与 Ackermann 模型，故而选用适配的 teb_local_planner。而 teb 默认提供的是包含平移和角速度 v 和 omega 的 geometry_msgs\Twist 消息

而不是 gazebo 仿真中要求的 ackermann_msgs\AckermannDriveStamped 消息，故而我们使用一个简单的脚本来监听 twist 消息并将其转化为 ackermann 消息。（详见附录 A）

第4章 方案实现

机器人在 ROS 系统下，以 move_base 为核心，完成巡航、避障任务。具体实现方案分析如下：

4.1 传感器信息

机器人使用了激光雷达与 IMU 来获取外部信息。在 gazebo 中，这些虚拟仪器分别发送扫描、位姿信息到 scan、imu_data 话题，然后再被 move_base、amcl、rf2o_odometry 等节点订阅，进行相关计算。

4.2 建立地图

选用了较为成熟的 Gmapping 作为激光雷达建图方案。Gmapping 是目前应用最广的 2D 激光 SLAM 方案，ROS 下 Gmapping 包集成了 Rao-Blackwellized 粒子滤波算法，为开发者隐去了复杂的内部实现。完成 Gmapping 基础参数配置后，通过获取激光雷达扫描信息和 gazebo 提供的精准里程计，即可获得基于概率的二维栅格地图，供 amcl 及 move_base 使用。

4.3 move_base 相关配置

move_base 是机器人完成路径规划与巡航避障的关键部分。其配置所需信息如图 5 所示。每部分详述如下：

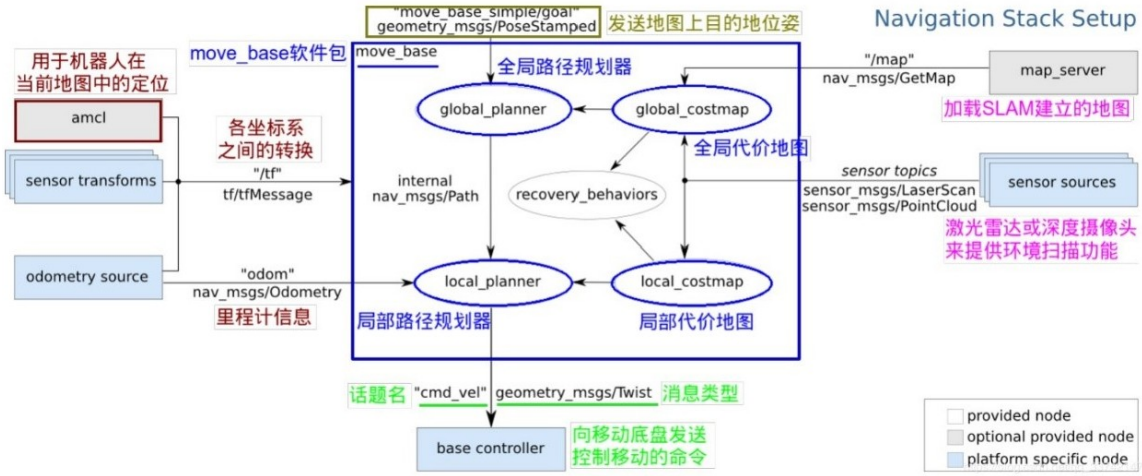

图 5: move_base 基本结构

4.3.1 里程计信息

机器人选用 rf2o 激光雷达里程计，并用 amcl 根据已知地图进行定位，在里程计附近使用范围使用粒子滤波进行扫描数据与地图的匹配，尽可能消除里程计的累积误差。

4.3.2 TF 配置

tf 维持时间缓冲的树结构中的坐标系之间的关系，并允许用户在任何时间点、任何两个坐标系之间进行点或向量的变换。

本次比赛中主要关注的 map、dom、base_link 这三个坐标系间的转换关系。其中：

map:一般为固定坐标系（fixed frame），与机器人所在的世界坐标系一致
base_link:机器人本体坐标系，与机器人中心重合
odom:里程计坐标系，表征机器人运动时里程计相对 map 坐标系产生的误差

base_link 与 odom 两个坐标系间使用 rf2o 里程计连接，map 与 odom 之间则通过 amcl(adaptive Monte Carlo Localization，自适应蒙特卡洛定位）连接。amcl 会根据已知地图进行定位，在里程计附近使用范围使用粒子滤波进行扫描数据与地图匹配，修正里程计的累积偏移。

4.3.3 teb 局部规划器

针对阿克曼模型的特点，以及竞速比赛的需求，选择 teb 作为局部规划器。teb 局部规划器是 2D 导航栈的 base_local_planner 的插件，能够针对运行时的不同情况，对全局规划器生成的初始轨迹进行实时优化，来为机器人提供路径指引。

它通过在车辆附近空间内散布大量的姿态，然后通过搜索树搜索最优方案，可以有效地获得最优轨迹。用户还可以为优化问题提供权重，以便在目标冲突的情况　下指定行为。比赛中，将 teb_local_planner 插件与里程计、局部代价地图、全局路径等信息连接，来实现局部路径规划。

4.4　控制指令转换

move_base 规划出行驶路线后，将控制信息发布至 cmd_vel 话题。这一 twist 类型消息无法为采用阿克曼模型的机器人所用，故需要经指令转换。

指令转换基于阿克曼转向的数学原理。阿克曼原理由德国车辆工程师 Lankensperger 于 1817 年提出的，指在车辆转弯时每个车轮绕同一中心转动，从而保证轮胎与地面之间无滑动摩擦而处于摩擦力最小的纯滚动状态。可简化为如图 6 所示模型[4]：

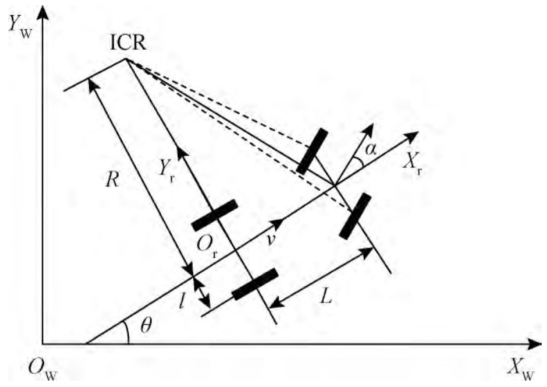

图 6: 简化 Ackerman 模型

第5章 测试分析

团队围绕比赛任务进行了大量的测试工作。其中，teb 局部规划器的参数繁多，调试较为复杂，下取几个特殊参数及其对路径规划的影响进行分析。

1） Dt_ref：最优树上相邻姿态的最优距离

此项参数的变化可以明显观察到控制精度的改变。

Dr_ref 改变最优距离，影响 teb 规划过程中最优树上的位姿状态，改变对机器人预期位置的设定，影响位姿与速度控制精度，如图 7、8。应该注意的是，此参数应当配合 dt_hysteresis 使用，当相邻姿态距离和 dt_ref 的差超过正负 dt_hysteresis 时，规划器将改变这一距离，通常将其设置为 dt_ref 的 10%。

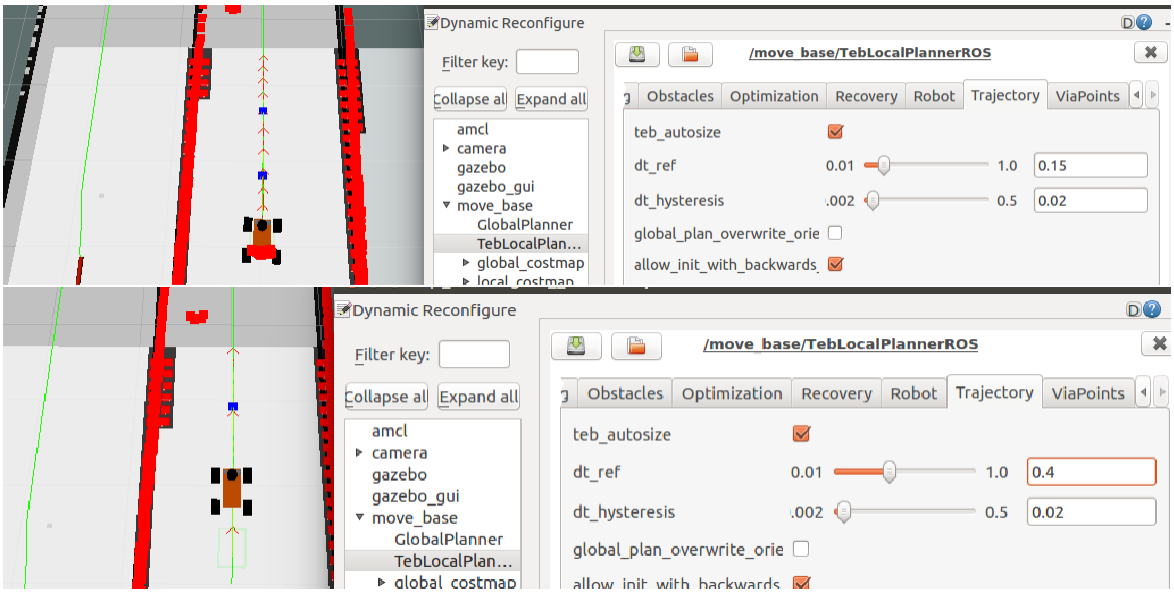

图 7: 不同 dt_ref 下的控制精度

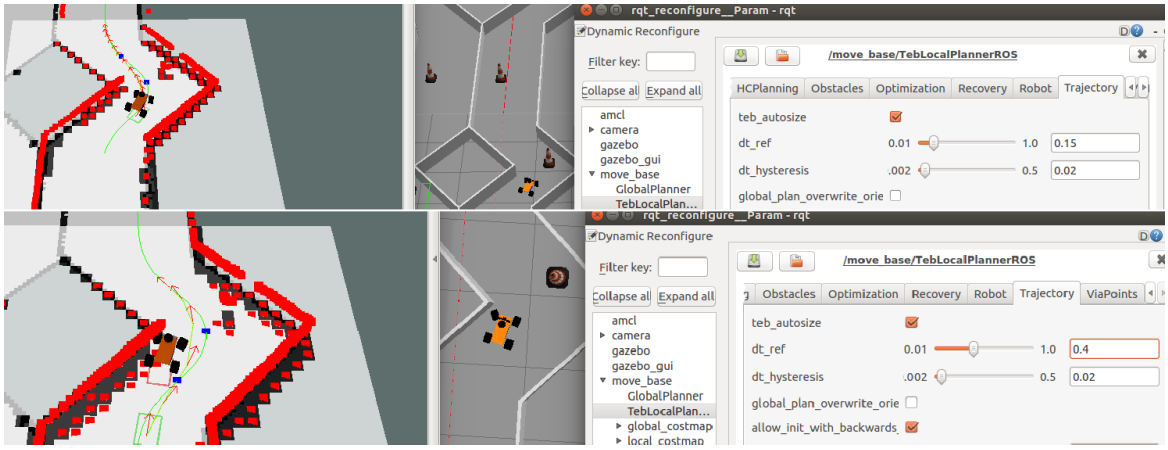

图 8: 面对同一障碍时的路径规划

2) **global_plan_viapoint_sep 与 feasibility_check_no_poses：从全局规划路径中提取的路点的相互距离与向前采样点个数**

这两个参数结合使用将影响在局部路径规划过程中路径追踪约束函数的取样精度，如图 9。

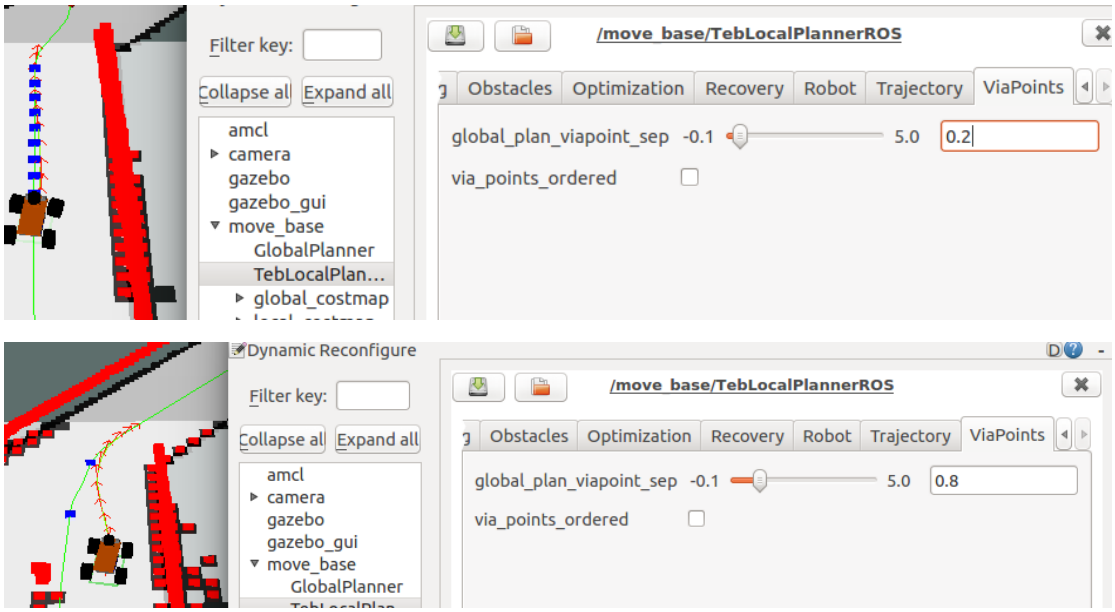

图 9:参数改变对取样精度的影响

3) **3. max_global_plan_lookahead_dist: 最大向前看距离**

此参数限制了在沿 global_path 考虑避障时的最大距离。

应随车辆最大速度的增大而增大，不应超过激光雷达等传感器的可靠测量范围，不应超过局部耗费地图的大小。

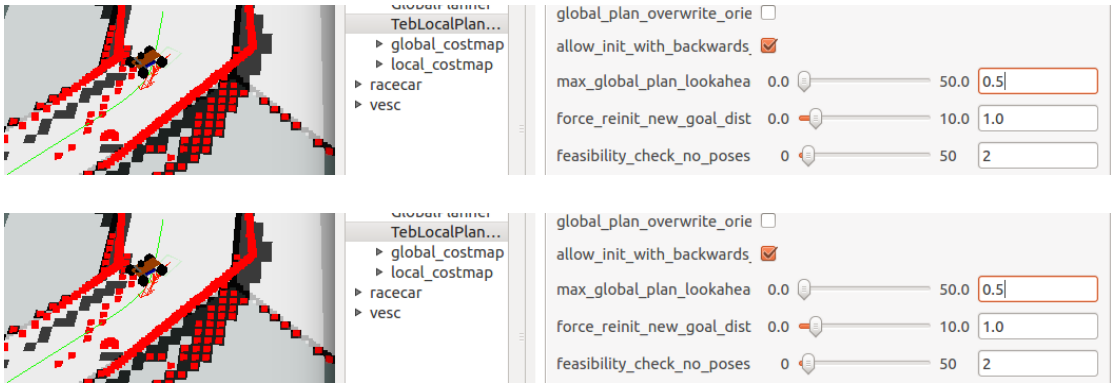

图 10:参数对规划情况的影响

从图 10 可以明显观察到向前规划距离减小并严重影响位姿规划。

第6章 作品总结

报名本届无人驾驶竞速车虚拟仿真赛以来，我们小组的成员团结协作，继续学习 ros 系统知识、配置开发环境、查阅相关项目的资料与技术文献，搭建项目工程、运行并调试优化参数，现已能够初步完成比赛任务。

在本技术报告中，我们主要介绍了 ROS_Gazebo 环境下实现 Ackermann 模型仿真的技术路线、TF 转换树的搭建、Slam_gmapping 建立栅格化地图的实现与导航栈配置、适配 Ackermann 模型的 teb 局部路径规划器原理及其优化。在机器人操作系统（ROS）下，使用 Lidar 和 IMU 姿态传感器获取的智能车周围环境信息与车体的姿态信息进行处理，通过 SLAM_gmapping 算法构建环境地图并依托该结点实现建图时的坐标系转换，使用 AMCL 算法结合 rf2o 里程计进行导航时位姿估计定位，借助 move_base 功能包+teb_local_planner，实现导航路径规划和移动机器人控制。

就作品整体而言，展现了队员们对主观能动性的充分发挥，将现有知识储备充分利用，接下来我们将继续完成这一工作并尝试针对 Teb_local_planner 繁琐的参数调试过程构造一种更简便的方法。

参考文献

[1] 常皓 and 杨巍. 基于全向移动模型的 gmapping 算法. 计量与测试技术, 43(10):1 - 4, 2016.

[2] Christoph Rsmann, Frank Hoffmann, and Torsten Bertram. Integrated online trajectory planning and optimization in distinctive topologies. Robotics Autonomous Systems, 88:142 - 153, 2017.

[3] Christoph Roesmann, Wendelin Feiten, Thomas Woesch, Frank Hoffmann, and Torsten Bertram. Trajectory modification considering dynamic constraints of au- tonomous robots. In Robotics Proceedings of ROBOTIK 2012 7th German Con- ference on, 2012.

[4] 郑凯林, 韩宝玲, and 王新达. 基于改进 teb 算法的阿克曼机器人运动规划系统. 科学技术与工程, 20(10):3997 - 4003, 2020. 11-4688/T

附录

A 代码示例

Listing 1: **twist2acker.py**

```
<?xml version="1.0"?>
<launch>
  <arg name="world_name" default="racetrack" />
  <arg name="gui" default="true" />
  <arg name="run_camera" default="false"/>

  <arg name="x_pos" default="0.0"/>
  <arg name="y_pos" default="0.0"/>
  <arg name="z_pos" default="0.0"/>

  <include  file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(find racecar_gazebo2)/worlds/$(arg world_name).world"/>
    <arg name="gui" value="$(arg gui)"/>
  </include>

  <!-- urdf xml robot description loaded on the Parameter Server, converting the xacro into a
      proper urdf
      file-->
  <param name="robot_description" command="$(find xacro)/xacro --inorder '$(find
      racecar_description2)/urdf/racecar.xacro'" />

  <!-- push robot_description to factory and spawn robot in gazebo -->
  <node name="racecar_spawn" pkg="gazebo_ros"
      type="spawn_model" output="screen" args="-urdf -param robot_description -model racecar -x
      $(arg x_pos) -y $(arg y_pos) -z $(arg z_pos)" />

  <!-- ros_control racecar launch  file -->
  <include  file="$(find racecar_control)/launch/racecar_control.launch" ns="/"/>

  <!-- Spawn the MUXs -->
  <arg name="racecar_version" default="racecar-v2" />
  <include  file="$(find racecar)/launch/mux.launch" ns="vesc" />

  <!-- Publish "better odom" topic that is normally generated by the particle  filter
  <node name="better_odom" pkg="topic_tools"  type="relay"
          args="/vesc/odom /pf/pose/odom" />-->

<!-- Localization -->
    <!-- AMCL-->
    <include  file="$(find racecar_gazebo2)/launch/includes/amcl.launch">
    </include>


```

```
<!-- ODOMETRY -->
    <!--rf2o_Laser_Odometry
    <include  file="$(find racecar_gazebo2)/launch/includes/rf2o.launch.xml" />-->



<!--rf2o_Laser_Odometry-->
    <include  file="$(find racecar_gazebo2)/launch/rf2o_laser_odometry.launch" />


<!-- robot_localization
    <node pkg="robot_localization"
        type="ekf_localization_node" name="ekf_se" clear_params="true">
        <rosparam command="load"  file="$(find racecar_gazebo2)/config/ekf_params.yaml" />
    </node>-->



 <!--Launch the simulation joystick control-->
 <rosparam command="load"  file="$(find racecar_gazebo2)/config/keyboard_teleop.yaml" />
 <node pkg="racecar_gazebo2"  type="keyboard_teleop.py" name="keyboard_teleop" />



</launch>
```

Listing 2: **racecar.launch**

```
#!/usr/bin/env python

import rospy, math
from geometry_msgs.msg import Twist
from ackermann_msgs.msg import AckermannDriveStamped

def convert_trans_rot_vel_to_steering_angle(v, omega, wheelbase):
  if omega == 0 or v == 0:
    return 0

  radius = v / omega
  return math.atan(wheelbase / radius)


def cmd_callback(data):
  global wheelbase
  global ackermann_cmd_topic
  global frame_id
  global pub

  v = data.linear.x
```

```
  steering = convert_trans_rot_vel_to_steering_angle(v, data.angular.z, wheelbase)

  msg = AckermannDriveStamped()
  msg.header.stamp = rospy.Time.now()
  msg.header.frame_id = frame_id
  msg.drive.steering_angle = steering
  msg.drive.speed = v

  pub.publish(msg)


if __name__ == '__main__':
  try:

    rospy.init_node('cmd_vel_to_ackermann_drive')

    twist_cmd_topic = rospy.get_param('~twist_cmd_topic', '/cmd_vel')
    ackermann_cmd_topic = rospy.get_param('~ackermann_cmd_topic',
        '/racecar/ackermann_cmd_mux/output')#/ackermann_cmd
    wheelbase = rospy.get_param('~wheelbase', 1.0)
    frame_id = rospy.get_param('~frame_id', 'odom')

    rospy.Subscriber(twist_cmd_topic, Twist, cmd_callback, queue_size=1)
    pub = rospy.Publisher(ackermann_cmd_topic, AckermannDriveStamped, queue_size=1)

    rospy.loginfo("Node 'cmd_vel_to_ackermann_drive' started.\nListening to %s, publishing to
        %s. Frame id: %s, wheelbase: %f", "/cmd_vel", ackermann_cmd_topic, frame_id, wheelbase)

    rospy.spin()

  except rospy.ROSInterruptException:
    pass
```