睿抗机器人开发者大赛
CAIM 工程创客赛道

ROS 机器人虚实挑战赛

技
术
报
告

队伍名称：ZZU-AIR（实车组别）

二零二五年

关于技术报告使用授权的说明

本人完全了解睿抗机器人开发者大赛——CAIM 工程创客赛道——ROS 机器人虚实挑战赛项关于保留、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。

日　　期：2025 年 8月 15日

目录

第 1 章 背景及意义

1.1 背景及意义

面对日益严峻的城市交通挑战，无人驾驶技术被视为核心解决方案，但其研发与落地过程充满挑战。直接在全尺寸实车上验证前沿算法，普遍存在成本高、周期长且安全风险大的核心难题。

为此，采用高集成度、等比例缩小的物理研究平台成为了一条高效、安全的技术验证路径。这种平台完整地复现了真实车辆“感知-决策-控制”的技术全链路，允许我们在真实的物理世界中：

1 **验证技术闭环：** 端到端地测试从数据采集到物理控制的完整流程，有效弥合理论与现实的鸿沟。

2 **实现快速迭代：** 在低成本、高容错的环境中大胆部署并优化前沿算法，显著加速技术的成熟周期。

3 **奠定应用基础：** 其成功的技术策略，证明了核心架构的可行性与可扩展性，为未来将技术“放大”并移植到真实车辆提供了强有力的工程依据。

本项目正是基于此理念，以 ROS 智能车为物理平台，在真实赛道环境中攻克高速动态避障等关键技术难题，旨在为无人驾驶技术的最终落地贡献一套经过充分物理验证的高效混合导航方案。

第 2 章 系统方案设计

2.1 方案选型与论证

在无人驾驶动态避障中，常见算法分为反应式方法与基于规划的非反应式方法，二者在原理、计算开销与适用环境方面差异明显。

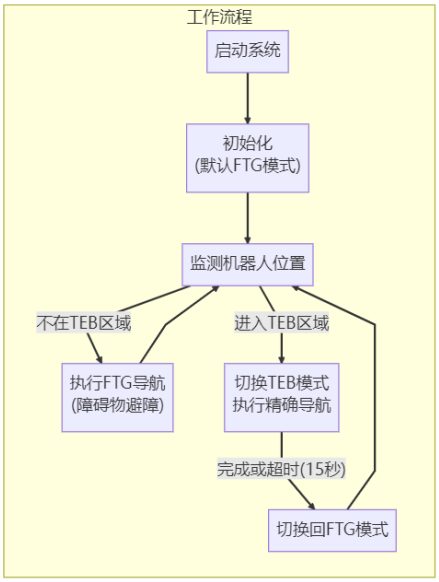
1.1 反应式方法（Reactive Methods）

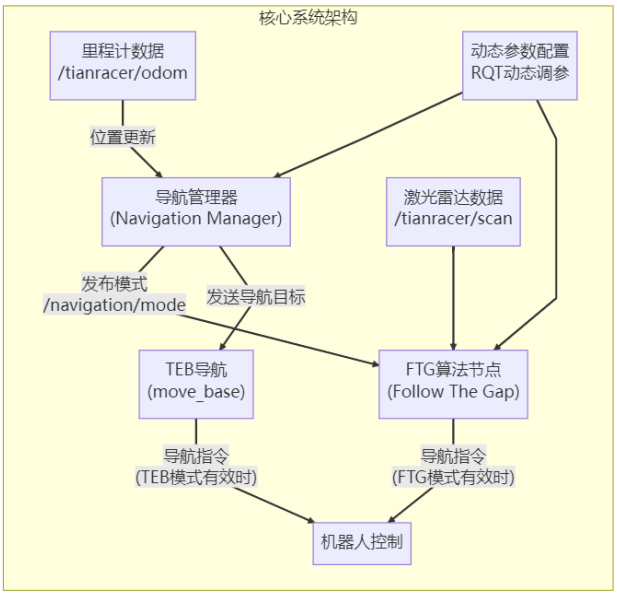
该类算法直接将传感器（如激光雷达）输入映射为控制指令，无需全局地图，响应迅速、计算轻量，适用于结构不明确或动态变化频繁的场景。典型代表包括向量场直方图（VFH）、动态窗口法（DWA）与本项目采用的 FTG 算法。FTG 通过寻找最安全的可通行间隙引导车辆前行，具备良好的灵活性，但易陷入局部最优，缺乏全局目标导向。

1.2 基于规划的方法（Planner-Based Methods）

此类方法结合全局地图与传感器信息，分为全局路径规划（如 A*、Dijkstra）与局部路径跟踪（如 DWA、TEB），具备精确导航能力与行为平稳性。但其依赖地图、计算开销大，对突发障碍响应较慢，存在一定局限。

本方案采用混合式分层导航框架（Hybrid Hierarchical Navigation Framework），融合 FTG 与 TEB 优点，通过导航管理器实现模式切换与智能调度，兼顾高速响应与精确控制，达成“快”与“准”的平衡。





2.2优缺点分析

方法	优点	缺点
反应式 (FTG)	1 **响应极快**：直接由传感器驱动，延迟低。 2 **计算开销小**：算法简单，适合嵌入式系统。 3 **无需全局地图**：适应性强，可在未知环境中运行。	1 **易陷于局部最优**：可能在凹形障碍物中打转。 2 **缺乏目标导向**：无法保证到达指定目标点。 3 **行为可能不够平滑。**
规划式 (TEB)	1 **目标明确**：能够精确导航至指定位姿。 2 **全局最优性**：能规划出完整的、较优的路径。 3 **行为平顺**：TEB 算法本身会优化轨迹的平滑性和时间。	1 **计算复杂度高**：需要更多 CPU 资源。 2 **依赖地图**：在无图或地图信息不准时无法工作。 3 **动态响应相对较慢**：频繁重规划可能导致卡顿。
本方案 (混合式)	1 **高效性**：在开阔区域使用高速 FTG 巡航，在关键区域使用精确 TEB 导航，**整体任务时间显著缩短**。 2 **鲁棒性**：结合了两种算法的优势，FTG 处理突发障碍，TEB 保证任务完成度，并设有**超时保护机制**防止系统卡死。 3 **高精度**：能够在需要时（如通过窄门、精确停靠）实现高精度位姿控制。 4 **灵活性**：通过动态参数服务器，可**实时调整**导航策略和区域，无需重启。	1 **系统复杂度增加**：需要设计和维护节点间的通信和状态切换逻辑。 2 **参数调试要求高**：需要精心设计模式切换的触发区域和条件，以保证切换过程的平顺和可靠。

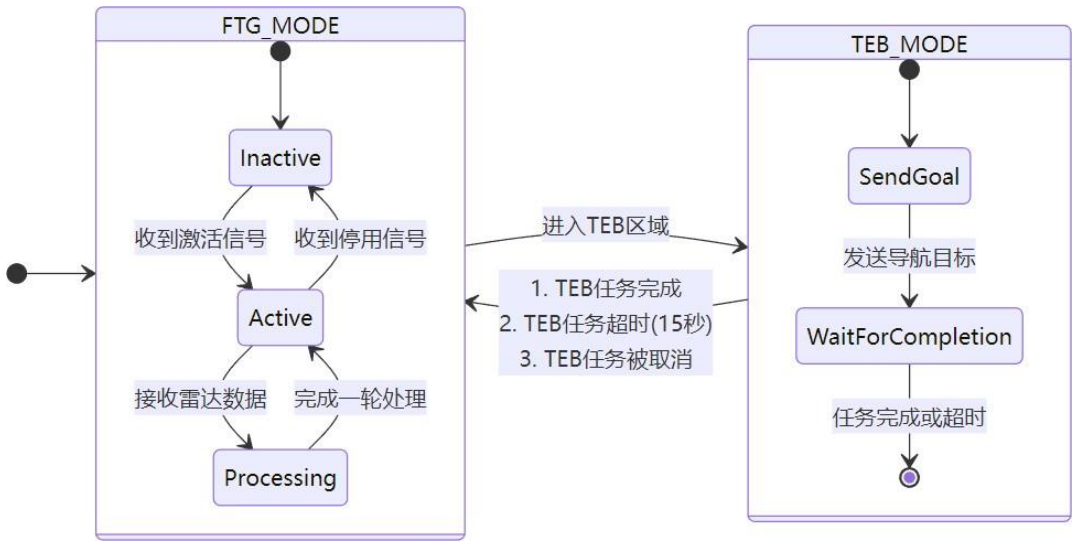
第 3 章 方案实现

本方案构建两个核心节点——**navigation_manager（导航管理器）**和 **ftg_node（FTG 算法节点）**

2.1决策层：导航管理器 (navigation_manager.py)

- **状态机**：它内部维护一个核心状态 current_mode，可以在 FTG_MODE 和 TEB_MODE 之间切换。
- **地理围栏（Geo-fencing）**：通过 ROS 的 dynamic_reconfigure 服务，动态定义多个“TEB 区域”。每个区域包含中心点坐标、半径、以及目标位姿。
- **模式切换逻辑**：
  - 该节点订阅机器人的里程计信息(/Tianracer/odom)。
  - 在 FTG_MODE 下，它持续检测机器人是否进入了任何一个预定义的 TEB 区域。
  - 一旦进入， 它立即将模式切换为 TEB_MODE，并向 move_base 发送该区域预设的精确导航目标。
  - 当 move_base 完成任务，或者**导航任务超过 15 秒**，管理器会强制取消 move_base 任务， 并将模式切换回 FTG_MODE。
- **全局广播**：管理器通过 latching 方式发布/navigation/mode 话题。

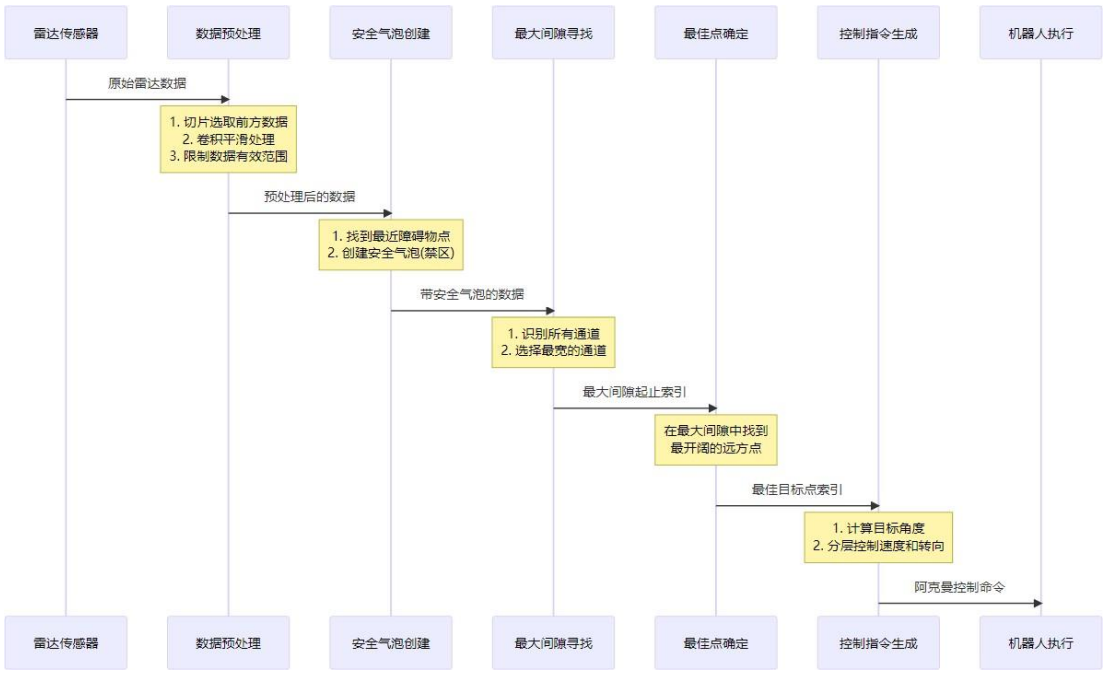
决策层实现的导航模式的切换逻辑示意图：



2.2执行层：**FTG 算法节点 (ftg_node.py) 与 move_base**

- **FTG 算法节点：**
  - 订阅激光雷达数据(/Tianracer/scan)和导航模式话题(/navigation/mode)。
  - **只有在接收到模式为 FTG_MODE 时**，它才会激活并执行完整的 FTG 算法流程，并最终发布 AckermannDriveStamped 控制指令。
  - 实现一套**分层速度控制策略**，能根据转向角的缓急自动调整车速。

核心算法：Follow The Gap(FTG) 示意



2.3 核心代码实现：混合模式切换与安全保障

```python
    def switch_to_teb_mode(self, zone_info):
        """
        切换到 TEB 模式，并启动一个 15 秒的超时计时器。

        Args:
            zone_info (dict): 区域信息字典
        """
        # 更新内部状态
        self.current_mode = "TEB_MODE"
        self.active_teb_zone_name = zone_info["name"]

        # 公告新状态
        self.publish_mode(self.current_mode)

        # 准备并发送导航目标给 move_base
        goal = MoveBaseGoal()
        goal.target_pose.header.frame_id = "map"
        goal.target_pose.header.stamp = rospy.Time.now()
```

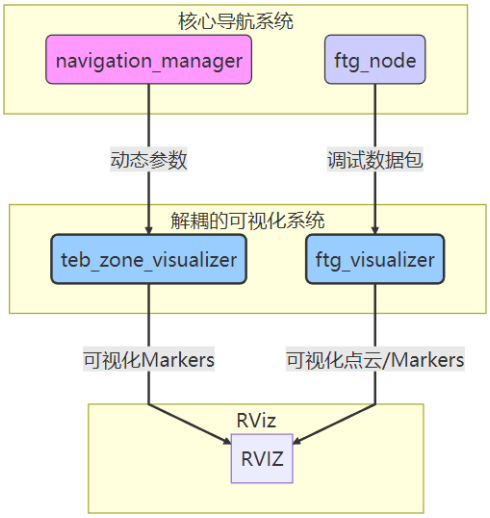
```
        # 设置目标位置
        goal.target_pose.pose.position.x = zone_info["goal"]["x"]
        goal.target_pose.pose.position.y = zone_info["goal"]["y"]
        goal.target_pose.pose.position.z = 0.0

        # 转换 yaw 角度为四元数
        quat = quaternion_from_euler(0, 0, zone_info["goal"]["yaw"])
        goal.target_pose.pose.orientation = Quaternion(*quat)

        # 发送目标，并指定任务完成后的回调函数
        rospy.loginfo(f"发送 TEB 导航目标到 ({zone_info['goal']['x']:.2
f}, "
                      f"{zone_info['goal']['y']:.2f}, "
                      f"{math.degrees(zone_info['goal']['yaw']):.1f}°)
")

        self.move_base_client.send_goal(goal, done_cb=self.on_teb_goa
l_done)

        # 【【【核心新增：启动超时计时器】】】
        # 创建一个 15 秒后触发一次的 Timer
        # oneshot=True 表示它只触发一次，然后自动停止
        rospy.loginfo(" 启 动 15 秒 导 航 超 时 计 时 器 ...")
        self.teb_timeout_timer = rospy.Timer(rospy.Duration(15.0),
                                             self.on_teb_timeout,
                                             oneshot=True)
```

场地搭建测试：

2.4创新性实现：解耦的调试与可视化架构

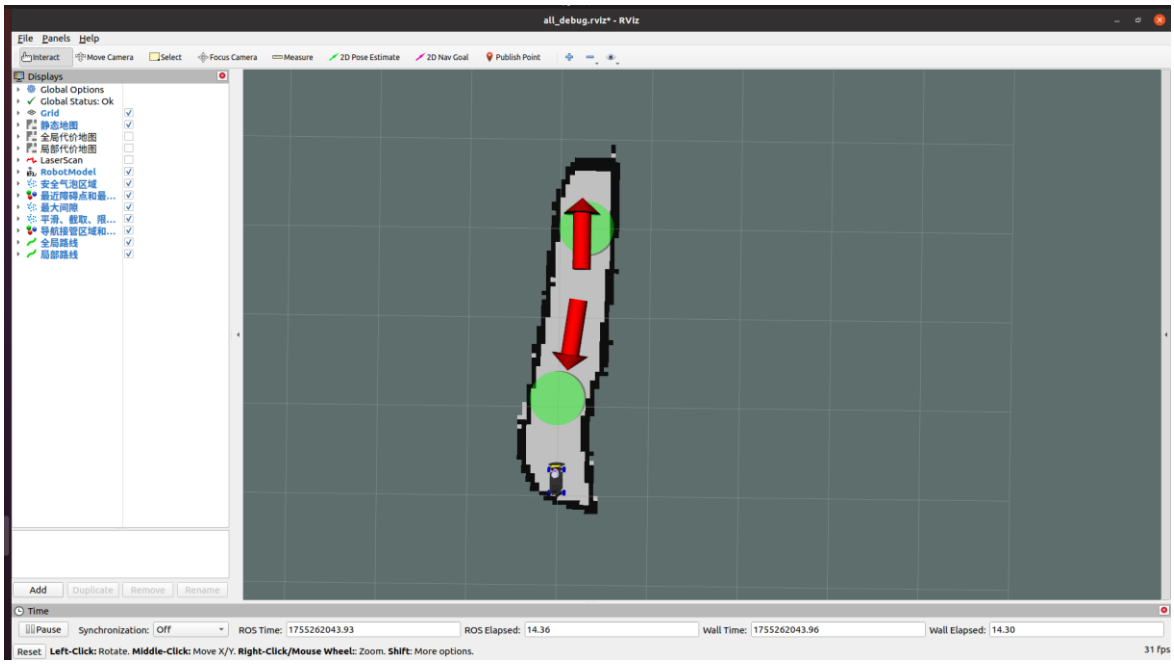
除了核心的导航逻辑，本方案在**系统可维护性和开发效率**方面也进行了深度优化，实现了一套完全解耦的调试与可视化架构。

架构图：

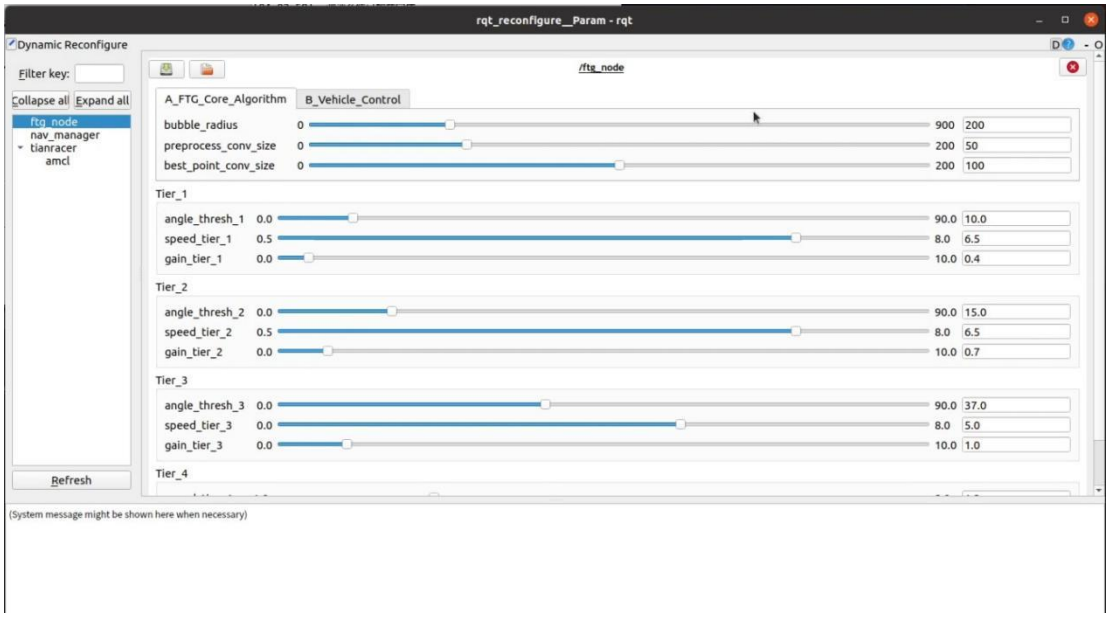


效果示意图：

1 **RVIZ** 可视化



2 rqt_reconfigure 动态调参



第 4 章 系统测试

2.4.1 数据分析

2.4.1.1 FTG 巡航模式性能调优

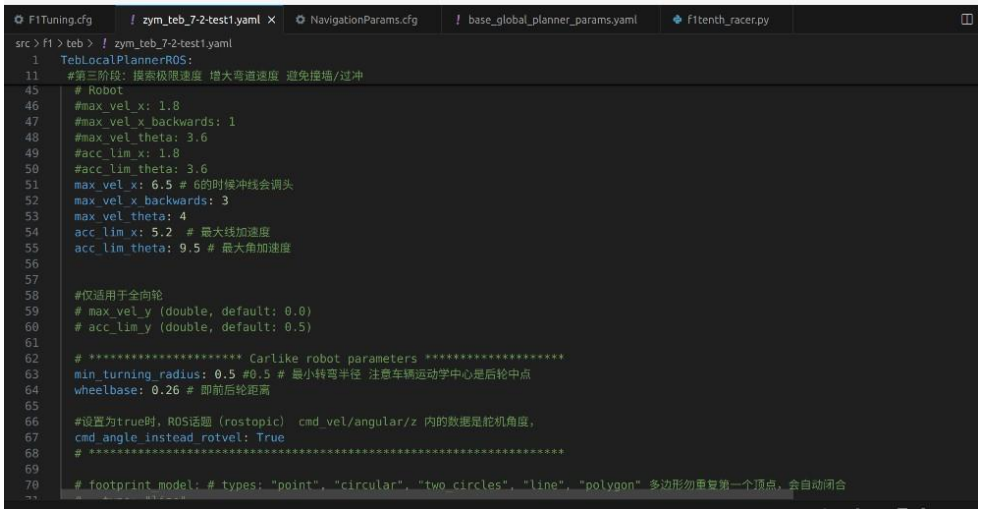
1 **核心算法参数调优：**

- **安全与效率的权衡：**bubble_radius（安全气泡半径）逐步增大至 **100**，使得车辆在高速行驶时能有效忽略远处的非关键障碍。
- **平顺性优化：**调整 preprocess_conv_size 和 best_point_conv_size，采用 **40** 和 **90** 的组合

2 **分层速度控制策略优化：**

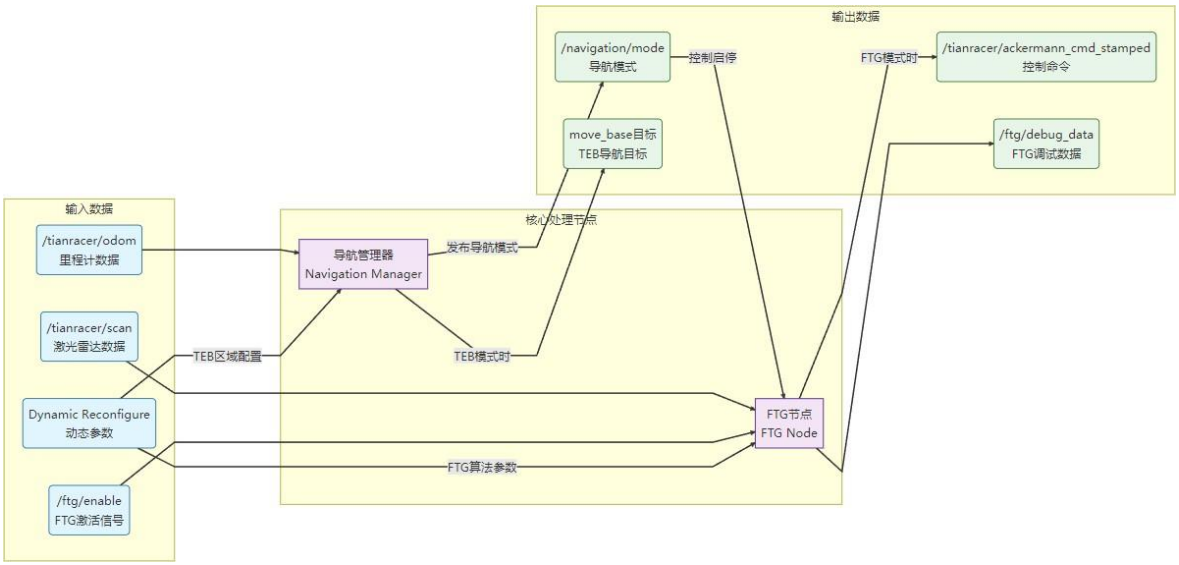
层级	角度范围（度）	速度（m/s）	转向增益	调优目标与结果分析
Tier 1 (高速直行)	0-10.0	2.5	0.4	在大直道上最大化车辆速度至 2.5m/s
Tier 2 (中速缓行)	10.0-15.0	2.5	0.6	针对高速缓弯速度提升 2.5 m/s 并适量增益
Tier 3 (低速急转)	15.0-37.0	1.5	0.8	对于急弯速度设定为 1.5m/s 并采用 0.8 的增益
Tier 4 (发夹弯道)	37.0-90.0	0.5	1.0	针对发夹弯或紧急避障等极端情况

2.4.1.2 TEB 精准对接模式参数调优



经过上述双重调优，系统整体性能得到质的飞跃。FTG 模式提供了极致的速度，而 TEB 模式则保证了关键节点的可靠通过。例如，在调优后，车辆通过狭窄 Z 形弯的时间从平均 12 秒减少到 5 秒，直角转弯成功率从 70%提升至 95%以上，最终实现了 69 秒的优异圈速成绩。

理解数据流图：



2.4.2 创新点

本项目最大的创新点在于**通过混合导航架构，从系统层面实现了定点巡航任务的整体时间优化**，而非仅仅局限于对单个算法的参数调优。

1 任务分段，各司其职：
我们将一个长距离的导航任务解构为“**高速巡航阶段**”和“**精准对接阶段**”。
1.1在占总路程 90%以上的开阔路段，我们启用轻量级、高响应的 **FTG 算法**。该算法不进行复杂的路径规划，直接以高速度（例如 2.5 m/s）沿着最开阔的方向前进，极大地提升了通行效率。
1.2仅在接近目标点、需要精确操作的“最后一公里”(即我们定义的 TEB 区域)，系统才无缝切换到 move_base，利用其强大的规划能力完成精准停靠。

2 容错恢复，避免时间黑洞：
传统的导航系统在规划失败或被卡住时，可能会陷入长时间的恢复尝试，甚至永久卡死，成为“时间黑洞”。我们设计的 **15 秒超时保护机制**。一旦超时，系统会果断放弃当前困难任务，切换回灵活的 FTG 模式继续探索，保证了整体任务流程的推进。

3 先进的工程化设计：解耦的调试与可视化架构：
我们为系统设计了独立的、可插拔的可视化模块，利用 dynamic_reconfigure 客户端等高级 ROS 工具，实现了对核心逻辑的零侵入式监控。

第 5 章 总结

本项目成功设计并实现了一套基于 ROS 的、高效、鲁棒的无人驾驶混合导航系统。我们没有拘泥于单一的算法范式，而是创新性地提出并构建了一个**分层控制框架**，将反应式避障算法（FTG）与基于规划的导航算法（TEB）进行了深度融合。该系统的核心是一个智能的**导航管理器**，它能够根据车辆的实时位置，在高速的 FTG 巡航模式和高精度的 TEB 停靠模式之间进行无缝、自动的切换。我们通过 dynamic_reconfigure 实现了导航策略的在线动态配置，极大地提升了系统的灵活性和适应性。同时，为 TEB 任务设计的超时保护机制，有效避免了系统在复杂环境下陷入僵局。

测试结果表明，该混合导航方案相比于传统的单一导航方法，在完成复杂的定点巡航任务时，能够**显著缩短总耗时**。