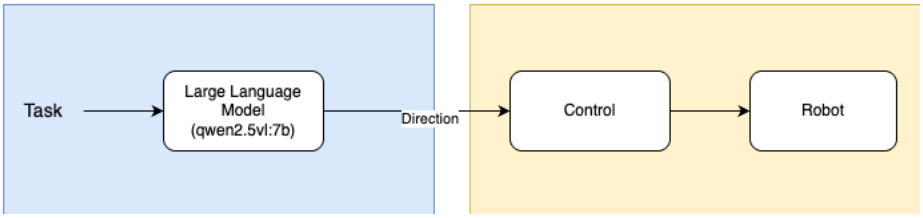
# 美团第三届
# 低空经济智能飞行管理挑战赛 性能赛
# 样例设计

2025.08.01

# 1. 总体设计




LLM 模块在接收到任务指令（以自然语言的形式）后，会首先理解该语言，接着解析出目标物体的方向（基于当前地面小车所处的位置），然后控制地面小车往指定的方向移动。

# 2. 主要功能

a) 接收自然语言指令并解析成运动方向
b) 控制地面小车按指定方向运动（旋转+前进）
c) 通过摄像头实时检测 ArUco 标记
d) 当到达 ArUco 标记附近时停止运动并发送完成状态

# 3. 类结构分析

## DirectionVelocityController.init(self)

a) **功能**: 初始化方向速度控制器，设置所有必要的参数和 ROS 接口
b) **主要初始化内容**:
   **ROS 节点**: 创建名为 'direction_velocity_controller' 的节点
c) **发布**:
   i. /magv/omni_drive_controller/cmd_vel (Twist): 发送速度命令
   ii. /status (Int32): 发送任务完成状态
d) **订阅**:
   i. /instruction (String): 接收语言指令
   ii. /magv/camera/image_raw (Image): 接收原始图像
   iii. /magv/odometry/gt (Odometry): 接收机器人位置信息
e) **关键参数配置**:
   i. linear_speed = 0.5 m/s: 线速度
   ii. angular_speed = 0.5 rad/s: 角速度
   iii. arrival_threshold = 2.0 m: 到达判断阈值
   iv. rate = 10 Hz: 控制循环频率
f) **ArUco 检测配置**:
   i. 使用 DICT_4X4_50 字典
   ii. ArUco 标记大小: 1 米
   iii. 摄像头内参根据 URDF 文件计算
   iv. 相机外参: 位置偏移 [0.5, -0.04, 0.57]，俯仰角 0.314 弧度
g) **方向映射**: 建立 8 个方向到角度的映射关系

# 4. 回调函数详细分析

a) instruction_callback(self, msg)
   i. **功能**: 处理接收到的语言指令
   ii. **参数**:msg (std_msgs/String): 包含语言指令的 ROS 消息
   iii. **处理流程**:
      1. 提取并清理指令文本
      2. 验证指令是否为空
      3. 记录接收到的指令
      4. 调用 process_language_instruction() 处理指令

b) image_callback(self, msg)
   i. **功能**: 处理摄像头图像，检测 ArUco 标记
   ii. **参数**:msg (sensor_msgs/CompressedImage): 原始图像消息
   iii. **处理流程**:
      1. **早期退出**: 如果已检测到 ArUco 标记，直接返回
      2. **图像转换**: 使用 cv_bridge 将 ROS 图像转换为 OpenCV 格式
      3. **ArUco 检测**: 使用 OpenCV ArUco 模块检测标记
      4. **姿态估计**: 计算检测到的标记的 3D 位置和姿态
      5. **位置计算**: 调用 calculate_aruco_position() 计算标记在世界坐标系中的位置
      6. **异常处理**: 捕获并记录图像处理过程中的所有异常
c) odom_callback(self, msg)
   i. **功能**: 处理机器人里程计信息，更新位置并检查是否到达目标
   ii. **参数**:msg (nav_msgs/Odometry): 里程计消息
   iii. **处理流程**:
      1. **位置更新**: 提取机器人在世界坐标系中的位置 (x, y, z)
      2. **姿态更新**: 将四元数转换为 yaw 角度
      3. **到达检测**:
         检查是否已检测到 ArUco 标记
         计算机器人到标记的距离
         如果距离小于阈值且未标记为已到达，触发到达处理

## 5. 核心控制函数 publish_velocity_command(self, direction)

a) **功能**: 根据方向执行完整的运动序列
b) **参数**:direction (str): 目标方向字符串
c) **执行流程**:
   i. 方向验证
      验证方向有效性
      检查方向是否在映射表中
   ii. 运动执行
      情况 1: 方向为 "front"
         直接前进，无需旋转
         持续前进直到 self.arrive = True
      情况 2: 其他方向
         第一阶段 - 旋转:
            i. 计算旋转时间: rotation_time = |target_angle| / angular_speed
            ii. 发送旋转命令直到达到预计时间
            iii. 短暂停止 0.5 秒确保稳定
         第二阶段 - 前进:
            i. 发送前进命令
            ii. 持续前进直到 self.arrive = True
         终止处理
            i. 发送停止命令
            ii. 记录运动完成日志

## 6. 指令解析函数 process_language_instruction(self, instruction)

a) **功能**：处理自然语言指令的完整流程
b) **参数**:instruction (str)：自然语言指令
c) **处理步骤**:
   i. **指令解析**：调用 parse_goal_direction() 解析指令
   ii. **结果验证**：使用 validate_direction() 验证解析结果
   iii. **运动执行**：调用 publish_velocity_command() 执行运动
   iv. **异常处理**：捕获并记录处理过程中的所有异常

## 7. 整体效果

任务：

```
{
        "scene_id": "Level01",
        "reference_length": 11,
        "timeout":100.0,
        "start_pose": [-4.5, 5.5, 0.0],
        "goal": [6.5, 5.5],
        "instruction": "move forward and stop at the tree"
}
```

任务拆解：

Direction: "front" -> 0 度

任务数据回放视频：https://s3plus.sankuai.com/udss-sim-data/result_25/download/demo.mp4