小米杯技术报告

队伍名称	魔笛手 Pied-Piper
成　　员	罗　宇
	郑皓天
	祝培元
	魏祥睿
指导老师	方　淼

目录

1 方案概述

1.1 系统框图

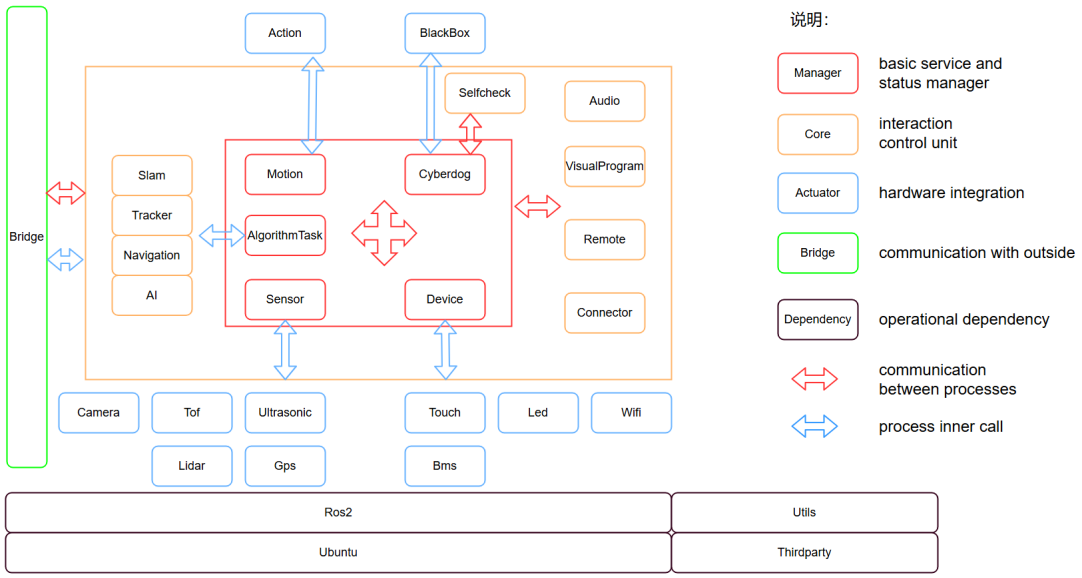

图 1: 系统框图

1.2 主要 ROS 节点

激光雷达信号节点 激光雷达信号节点是一个基于 ROS 2 的 Python 应用程序，它通过 rclpy 库实现，专注于处理来自激光雷达传感器的数据。该节点使用 Node 基类来创建 LidarNode，负责订阅/scan 话题上的 LaserScan 消息，并在接收到数据时通过 lida_callback 方法提取特定角度的测量距离。测量结果被封装为 Float32 消息类型，并发布到/lidar_distance 话题上，以便其他节点使用。此外，节点利用 QoSProfil‘确保传感器数据传输的可靠性。main 函数处理 ROS 2 的初始化和节点的生命周期，包括启动事件循环和资源的清理。整个节点的设计旨在为机器人提供实时的距离测量功能，以便于进行环境感知和避障。

RGB 相机信号节点 RGB 相机信号节点是一个 ROS 2 应用程序，它通过订阅/rgb_camera/image_raw 话题来接收图像数据。节点利用 cv_bridge 将 ROS 图像消息转换为 OpenCV 格式，以便使用 OpenCV 库 cv2 进行图像处理。ImageHandler 类负责图像的接收、转换、处理和显示。它检测图像中的绿色区域，并将检测结果通过 ROS 话题发布。节点还设置了 QoS 配置以保证消息的可靠传输，并使用定时器周期性地更新图像显示。main 函数负责节点的初始化、资源的管理和运行循环的维护。脚本最后通过条件判断确保当作为主程序运行时才启动 main 函数，同时导入了 logging 模块为后续可能的日志记录功能做准备。

多话题节点 MultiTopicNode(多话题节点) 为 ROS 2 的一个节点，它具备以下核心功能：在初始化阶段，节点配置了与 LCM 协议的通信，加载了配置文件 cyberdog.toml，并设置了必要的初始变量和标志位，同时创建了多个订阅者以接收激光雷达和 RGB 数据。节点通过回调函数实时更新传感器数据，并通过一系列逻辑处理函数 update_num_based_on_sensors1 至 update_num_based_on_sensors5，根据传感器数据和预设条件更新状态，决定执行的动作序列。在 run 函数的持续循环中，节点检查传感器数据更新标志位，并调用相应的处理函数和 update_and_publish 函数，后者从配置文件中获取控制参数，打包成 robot_control_cmd_lcmt 消息，并通过 LCM 发布到指定通道，以控制机器人的动作。此外，节点还定义了多个动作序列函数，例如 left_jump 和 right_jump，通过循环和延时来模拟机器人的物理动作。最后，main 函数负责节点的初始化、事件循环的启动和程序的退出，确保节点能够响应传感器输入，执行复杂动作，并在结束时正确关闭。

1.3 通信框架

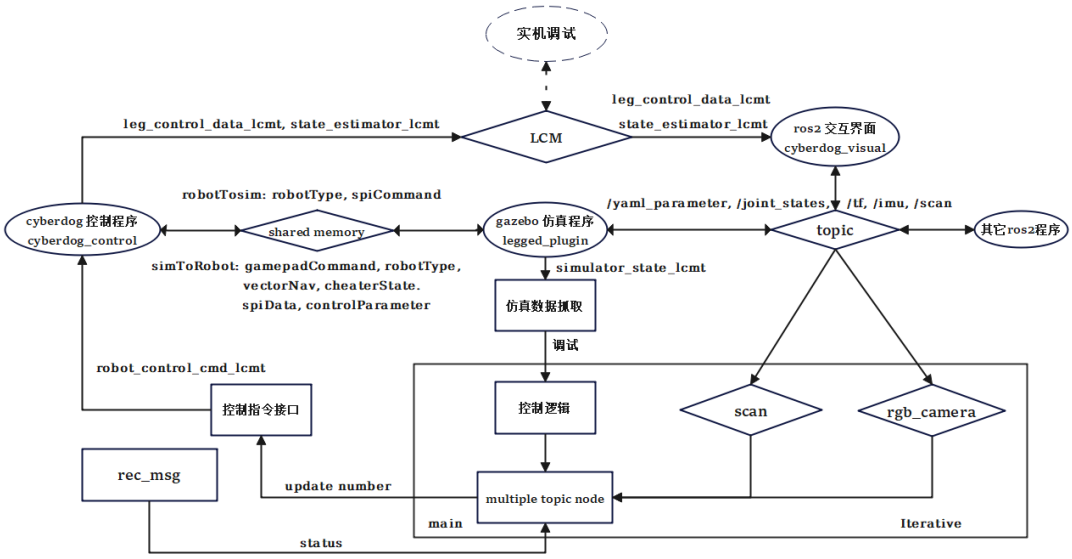

图 2: 通信框架图

1.4 ROS Topic 和 LCM 通信接口介绍

1.4.1 ROS Topic

我们创建了 3 个 ROS Topic：

- **lidar_distance_90**

  订阅名为 lidar_distance_90 的主题，消息类型为 Float32，回调函数为 self.lidar_callback，队列大小为 10。这个主题用于接收正方向上的激光雷达 (LIDAR) 距离数据。

- **lidar_distance_0**

  订阅名为 lidar_distance_0 的主题，消息类型为 Float32，回调函数为 self.lidar_callback_0，队列大小为 10。这个主题可能用于接收机器狗正右方向上的激光雷达 (LIDAR) 距离数据。

- **color_position**s

  订阅名为 color_positions 的主题，消息类型为 String，回调函数为 self.rgb_callback，队列大小为 10。这个主题用于接收颜色定位信息。

1.4.2 LCM

关于数据的发送，我们使用了 robot_control_cmd_lcmt 的通信接口用来实现对机器狗基础动作的调用，调用了 robot_control_response 的通信接口来获取机器人运动过程中所反馈的数据用于机器人的状态调整，同时用 simulator_lcmt 的接口来捕获一个上帝视角的数据用来观察机器人状态并调整相应的步态与控制.

2 创新点/亮点概述

2.1 RGB 智能量化与决策支持技术

我们研发的 RGB 智能量化与决策支持技术 (Color Intelligent Quantization and Decision Support Technology, CIQDST) 是为了解决 opencv 中 set() 函数识别准确度不高的问题，我们自研的萨维塔算法通过将图像转换到 HSV 颜色空间，定义颜色范围，进行区域划分，以及使用掩膜识别特定颜色，实现了对图像中颜色分布的精确量化和智能分析。它通过权重调整和条件判断，将颜色分布转化为可操作的信息，并通过 ROS 2 发布机制，为机器人或其他系统提供了决策支持。这种创新的方法不仅提高了颜色识别的准确性，还增强了系统的交互性和适应性。

2.2 反向攀登技术

论文 [马 16] 基于 ZMP 理论分析, 提出了一种新的四足机器人质心调整方法, 通过调节四足机器人前后腿小腿腿长的方法来调节质心位置. 这种调节腿长的方法比其他方法控制简单, 可以同时实现两个方面的调整: 一方面是调整机器人质心前移, 使质心在地面的投影点落在支撑多边形内部; 另一方面是调整机器人机体姿态角度, 防止机器人向后翻倒. 鉴于调整质心在本次比赛中过于麻烦，我们团队化繁为简，打破传统观念，简采取让机器人倒着上坡的创新思维，既保证了质心前移，同时也调整了机体姿态角度，达到了防止机器人向后翻滚的目的，实现了一石二鸟。

2.3 支撑平面估计

在本次比赛中，我们采用了支撑平面估计技术，对机器狗的落足点和身体姿态进行了优化。通过学习相关理论基础，并分析/home/cyberdog_sim/src/cyberdog_locomotion/control/src/convex_mpc 路径下的 convex_mpc_loco_gaits.cpp 文件,我们对公版代码中的 USE_TERRAIN_DETECTER 模块有了初步的掌握。在此基础上，我们对代码参数进行了调整，提升了机器狗在斜坡等特殊地形上的运动性能，使得 cyberdog 2 在赛道上的行走更加稳健。在接下的比赛中，我们将继续研究支撑平面估计方法，努力让 cyberdog 2 的步态更加稳定。

2.4 步态和 PID 调参

在本次比赛中，我们对机器狗的步态周期进行了探索和优化。通过调整运动周期、前后足的步长偏移以及支撑相位的比例，我们对 cyberdog 2 的对角步态和平行步态进行了多种尝试。我们根据不同地形的特点，将这些步态策略性地应用于赛道的不同赛段，使得机器狗能够以更加高效和适应性强的方式完成比赛。

同样，我们也从 PID 控制的角度对机器狗进行了调整。通过精调比例系数 K_p 和微分系数 K_d 的大小及其比例，我们降低了机器狗在攀爬楼梯时的震荡现象，并成功引导它登上了我们自主构建的仿真楼梯。然而，由于赛道中楼梯的高度和宽度具有一定特殊性，我们的步态策略在应对比赛中的楼梯时仍面临一些挑战。我们计划在准备决赛的过程中，进一步优化我们的步态算法，以克服这些难题，让 cyberdog2 在各种复杂环境中都有稳定的表现。

3 关键技术

3.1 环境感知

3.1.1 地形障碍识别算法介绍

由于 openCV 中自带的 set() 函数不能较为准确的识别 rgb 值并作出正确的分类，我们利用已有的知识设计了简单的萨维塔算法，功能是通过 RGB 相机收集图像数据，并分析图像中的颜色分布。以下是对算法中颜色赋值和处理部分的详细解释：

1. **颜色范围定义**：算法首先定义了一个颜色范围的字典 color_ranges，其中包含了不同颜色的 HSV 阈值以及每种颜色对应的一个数值。这些数值将用于后续的颜色分布计算。

2. **图像转换**：通过回调函数 image_callback 接收到的图像被转换为 HSV 颜色空间，这是由于 HSV 颜色空间更适合颜色的识别和分割。

3. **区域划分**：图像被水平划分为三个区域：左侧、中间、右侧。这种划分允许算法分别评估每个区域的颜色分布。

4. **颜色检测与累加**：对于每个区域，算法遍历 color_ranges 中定义的颜色，使用 cv2.inRange 函数创建掩膜来识别属于特定颜色范围的像素。如果一个区域中存在某个颜色，它的对应数值会被累加到该区域的总和中。

5. **权重计算**：每个区域的累加结果会乘以一个预定义的权重值 (left_weight、middle_weight、right_weight)，这些权重反映了不同区域在最终决策中的重要性。

6. **结果综合**：最后，算法将加权后的三个区域的颜色值总和相加，得到一个综合结果 res。

7. **条件判断与赋值**：根据 res 的值，算法执行一系列的条件判断，将 res 映射到一个特定的值或范围内。例如，如果 res 等于某个预定义的数值，算法将 mm 赋值为一个特定的数字，代表了一个特定的颜色分布情况。对于特定的 res 值，算法还会进一步分析图像中的绿色分布，并根据绿色在左右两侧的比例，将 mm 赋值为 7、8 或 wrong。

8. **颜色信息发布**：如果计算出的 mm 值与上一次不同，算法将这个值转换为字符串，并通过 ROS 2 的发布者发布到/color_positions 话题上。这样，其他节点可以订阅这个话题来获取颜色分布的信息。

9. **更新与显示**：一旦颜色信息发布，last_order 变量将更新为当前的 mm 值，以跟踪颜色分布的变化。同时，图像通过 display_image 函数显示出来，供用户或系统其他部分进行视觉检查。

整个算法的关键在于通过颜色的数值化表示和权重计算，将图像的颜色分布转换为可量化和可发布的信息，从而为机器人提供颜色识别和决策支持。

3.1.2 传感器的使用

雷达 我们创建了 lidar_pkg 软件包来处理雷达收集到的数据。在 lidar_publisher.py 文件中，我们定义了 LidarNode 类，代表雷达节点，通过这个节点来订阅和接收雷达的消息。具体来说，我们在 LidarNode 类中创建了一个订阅者 lidar_sub，用于订阅/scan 话题上的 LaserScan 消息，并将 lidar_callback 函数作为订阅消息的回调函数。同时，我们创建了一个发布者 distance_pub，用于发布 Float32 消息到/lidar_distance 话题上。通过这些步骤，我们成功实现了雷达数据的接收和处理，并且能够通过 ranges[ ] 数组中的数据为机器狗提供前方测距的功能。

RGB 相机 我们创建了 get_RGB 软件包来处理 RGB 相机收集到的图像数据。在 get_rgb.py 文件中，我们定义了 ImageHandler 类，代表图像处理节点，通过这个节点来订阅和接收相机的消息。具体来说，我们在 ImageHandler 类中创建了一个订阅者，用于订阅/rgb_camera/image_raw 话题上的 'Image'消息，并将 image_callback 函数作为订阅消息的回调函数。同时，我们创建了一个发布者，用于发布 String 消息到/color 话题上。通过这些步骤，我们成功实现了图像数据的接收和处理，并且能够通过分析图像中的 RGB 值来为机器狗提供颜色识别和分布分析的功能。这使得机器狗能够根据颜色信息做出相应的决策，例如导航或物体识别。

3.2 运动控制

初始化 程序开始时，创建了一个 ROS2 节点和一个 LCM 通信实例，加载了配置文件，并初始化了控制变量。

动作序列 定义了一系列的动作执行函数，如 left_stop、right_stop、circle_stop、double_left、right_jump 和 left_jump 等，每个函数都通过循环和延时来控制动作的执行。

控制指令更新 update_and_publish 函数根据配置文件中的参数更新机器人控制指令，并将其通过 LCM 发布到指定话题。

主循环 run 方法包含一个主循环，连续调用 execute_sequence 方法来执行预定义的动作。

多线程 使用 Python 的 threading 模块创建一个控制线程，使得控制逻辑可以在后台运行，而主线程则处理 ROS2 的事件循环。

配置文件解析 使用 toml 模块解析名为 cyberdog.toml 的配置文件，该文件包含了机器人动作所需的详细参数，如模式、步态 ID、期望速度、加速度等。

LCM 通信 LCM 在运动控制中主要有两个作用：第一，发布控制指令，使多个单位协同完成运动任务；第二，抓取实时仿真数据，为运动控制提供信息。

日志记录 使用 ROS 2 的日志系统记录程序运行时的信息，便于监控和调试。

程序入口　main 函数作为程序的入口点，初始化 ROS2，创建节点实例，启动控制线程，并运行事件循环。

程序结束　设置 running 标志为 False 以结束循环，等待控制线程结束后销毁节点并关闭 ROS2。

3.2.1　小米运控公版代码的修改

1. 打开了 USE_TERRAIN_DETECTER 用于支撑平面估计自动调整 rpy 和落足点，身体趋向平行于地面使其稳定的在斜坡上行走.

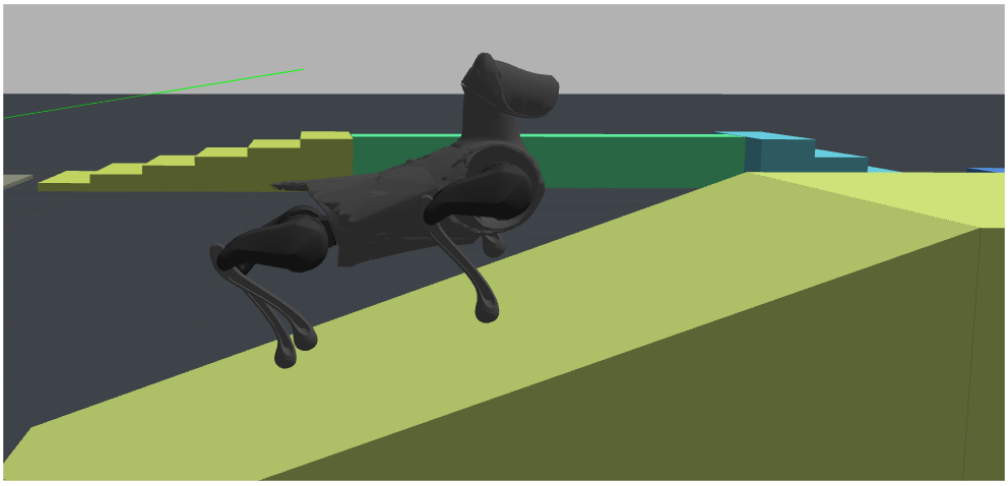

图 3: 开启 USE_TERRAIN_DETECTER 后，cyberdog2 在斜坡上稳定行走

2. 结合台阶宽度与高度修改所使用步态的周期，调整支撑相的占比，增强机器狗在台阶上的稳定性。
3. 由于在斜坡台阶等地形对机身的稳定性扰动较大，我们修改了 convex_mpc_loco_gaits 下相应的步态对应落足点在 z 坐标的控制力度，使其相适应于比赛的地形。

经过以上调整，我们在官方现有步态的基础上优化了机器狗在楼梯赛段的表现，虽然还未能成功通过赛道，但已经能登上我们自己搭建的仿真楼梯：

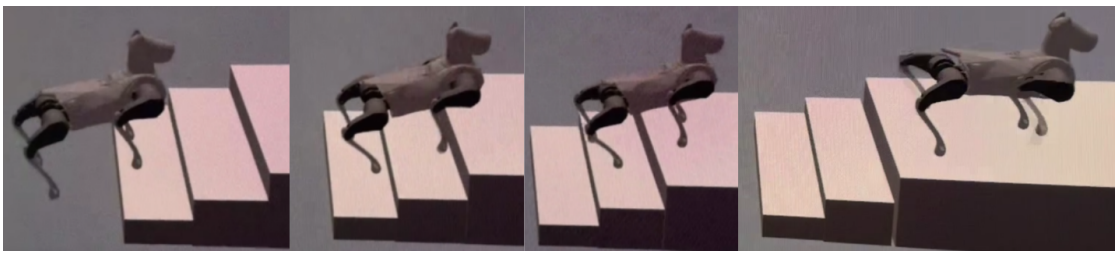

图 4: cyberdog2 成功登上我们自主搭建的仿真楼梯

3.2.2 强化学习

强化学习作为近年来机器人控制领域的新方法，展现出了极好的效果和适应性。我们团队也计划在本次比赛中，通过强化学习训练狗的运控算法。我们主要参照了论文 [RHRH22] 和 github 仓库 [leg]，并初步将 cyberdog2 部署到 Isaac 仿真环境中进行训练。但由于时间限制，我们训练的模型还存在一定的问题，需要进一步调整、优化，我们会在复赛中进一步展示这一部分的内容。

以下是我们总结的强化学习部署要点：

1. 从/home/cyberdog_sim/src/cyberdog_simulator/cyberdog_robot/cyberdog_description /xacro 路径下的 robot.xacro 文件中提取出 cyberdog2 的 xacro 文件，并将其转化为 urdf 文件, 以此作为 cyberdog2 的仿真模型在 Isaac 仿真环境中进行训练。
2. 在 legged_gym/legged_gym/envs/目录下建立一个 cyberdog2 文件夹，然后在 cyberdog2 文件夹中建立一个 cyberdog2_config.py 的配置文件
3. 修改 task_registry 文件，添加 cyberdog2 相关的代码。
4. 修改/legged_gym_master/legged_gym/envs/base/legged_robot.py 文件中 compute_reward 函数，以此来调整机械狗在训练过程中获得的奖励

3.3 特殊地形处理

3.3.1 台阶/高台/斜坡

斜坡 选择相应的爬坡的动作模式。开启地形检测。调整 step_height 来适应斜坡的坡度。根据斜坡的倾斜程度，调整了 rpy_des 中的俯仰角 (Pitch)。调整 pos_des 中的 Z 轴值，让机器人的身体升高。降低 vel_des 中的速度值，以防止过快爬坡导致失去平衡。

台阶 通过地形估计判断出机器狗是处于上楼梯还是下楼梯，将机器人的落足点优化进行处理，选用合适的运动模式，调整摆动腿和支撑腿的相位，在遇到台阶的震动扰动时，将 WBC 的 pd 减小，实现盲走楼梯。

3.3.2 崎岖地形，沙石等

针对这一情况，我们主要做出以下 4 处调整：设置适合的动作模式; 调整 step_height 来达到最佳通过效果; 调整 pos_des 中的 Z 轴值，让机器人的身体升高; 降低 vel_des 中的速度值，以防止过快导致失去平衡。

3.3.3 独木桥，转角，石板

选择常规步态通过即可。

4 未来规划

1. 对于赛道的楼梯部分，高楼梯的部分还缺乏稳定性，目前仅能实现较低的楼梯攀爬，还需加强对传统运控算法的进一步学习，加入视觉功能去检测台阶边缘防打滑，去开发实现更稳定的底层控制。
2. 高台赛道很难通过修改俯仰角与抬腿高度完整通过，缺少对跳跃动作的研究，未来应进一步的学习与调整，实现高台的稳定通过。
3. 针对决赛比赛规则，程序中加入机器狗的自我状态调整，防跌落和闭环控制中的异常处理，提高在决赛赛场上的通过率，拿到更高的分数。
4. 增添闭环控制的信息获取渠道实现多传感器融合，帮助机器狗实现更精准的判断。
5. 使用定位与地图构建技术，帮助机器狗实现自主导航和路径规划同时提高多场景适应性。
6. 使用 AI 大模型技术做高层规划器帮助机器狗实现自主学习与决策，使用强化学习技术做底层运动控制的开发。

参考文献

[leg] legged_gym. `https://github.com/leggedrobotics/legged_gym`.

[RHRH22] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[马 16] 马宗利，张培强，吕荣基，王建明. 四足机器人坡面行走稳定性分析. 东北大学学报, 2016.