基于大模型的无人机投送具身智能体（ROS + Gazebo）

学号：BZ2516004
姓名：杨晓哲
方向：03-Agent-Embodied
代码仓库：https://github.com/CH-YXZ/ros-embodied-uav-agent
演示视频：
https://github.com/user-attachments/assets/86678595-30e0-42fd-bddd-4e8737aea655
PR 链接：https://github.com/CAI-testbooks/LLM-Course/pull/23

摘要

本文实现了一套基于 ROS 与 Gazebo 的无人机投送具身智能体系统。用户输入一句自然语言任务描述，例如起飞到指定高度，前飞到目标位置，下降悬停并投放载荷，随后继续前飞并降落。智能体侧采用大模型常用的工具调用与 ReAct 规划范式，将自然语言指令解析为结构化计划，计划以 JSON 形式表达为按序动作与参数集合。执行器依据计划逐步调用 ROS 服务与话题接口，驱动仿真无人机完成起飞、航迹飞行、下降悬停、投放与降落等动作，从而实现可复现的端到端任务执行流程。系统支持一键重播能力，能够同时重置仿真环境与工具状态，并提供轨迹与日志的全过程记录，便于后续对任务成功率、步骤耗时与飞行稳定性进行统计评估。为保证工程可扩展性，系统将计划生成模块与计划执行模块解耦，支持在不改动执行器的前提下替换不同的大模型与提示策略。实验在 Gazebo 仿真环境中进行多轮重复验证，结果表明系统能够稳定完成完整投送流程，重播条件下轨迹与关键事件时序具有良好一致性，日志统计可用于量化分析各步骤耗时分布与失败环节定位，从而验证了该系统在可复现执行与评测闭环方面的有效性。

2 引言

近年来，大语言模型（LLM）推动具身智能从“端到端策略网络”转向“语言推理 + 工具执行”的分层范式：模型负责将自然语言目标拆解为可执行的多步计划，并在执行过程中结合环境反馈进行纠错与迭代。典型代表是 ReAct 框架，将“推理轨迹”与“行动调用”交错生成，使模型能够在交互环境中边思考边行动，提高长程任务的可解释性与成功率[1]。Toolformer 进一步证明语言模型能够通过自监督学习掌握“何时调用工具、如何组织参数并利用返回结果”，从方法论上支撑了工具调用型智能体的工程落地[2]。在机器人任务中，SayCan 将 LLM 的高层语义分解与低层技能可行性评估结合，强调“可执行性约束”对安全与稳定执行的重要性[3]；Inner Monologue 则通过语言化反馈形成闭环修正，展示了执行反馈在长程任务完成中的关键价值[4]。
与此同时，视觉-语言-动作（VLA）方向通过规模化数据与大模型结构提升具身

泛化能力：RT-1 在大规模真实机器人数据上验证了通用策略的可扩展性[6]；RT-2 将互联网视觉-语言知识迁移到机器人控制，显著提升对新物体与新指令的泛化，并可结合思维链进行多阶段推理[7]；PaLM-E 进一步将视觉与连续状态估计嵌入到语言模型中，实现统一的多模态具身推理[8]。在开放世界与跨平台预训练方面，Voyager 在 Minecraft 中通过“技能库 + 迭代提示 + 环境反馈”实现开放式终身学习[9]；Open X-Embodiment（RT-X）汇聚多机构多机器人轨迹数据，为跨机器人泛化研究提供了重要数据基础[10]。

本项目贡献点：

1）具身闭环系统实现：构建了一个可复现的 ROS+Gazebo 无人机投送具身智能体系统，实现“自然语言指令 → 结构化计划（JSON Plan）→ 工具调用执行 → 状态反馈/日志”的完整闭环，符合 ReAct/Toolformer 式的工具调用智能体范式[1,2]。

2）计划生成与执行解耦：将 Agent（计划生成）与 Executor（计划执行）模块化解耦，当前使用规则版 Agent，后续可无缝替换为真实 LLM（Tool Calling）而不改执行器逻辑，呼应 Code as Policies 的“程序/计划即策略”思想[5]。

3）可复现重播与实验支撑：实现一键重播（复位世界与工具状态）与轨迹/日志输出，解决仿真重跑时初始位姿漂移、投放状态残留等问题，提升具身实验的复现性与可评估性。

4）面向泛化与错误恢复的接口预留：参考 SayCan 的可行性约束与 Inner Monologue 的闭环反馈思想，预留安全过滤、执行反馈与错误恢复接口，为后续引入真实 LLM 与更复杂任务扩展打下基础[3,4]。

3.系统总体设计

本系统面向“自然语言指令驱动无人机在仿真环境完成多步投送任务”的具身智能体场景，采用分层解耦的总体架构：上层由 Agent 负责把自然语言转化为结构化可执行计划（Plan），中层由 Executor 负责解释计划并调度工具调用，底层由 ROS/Gazebo 工具节点实现具体动作与环境交互，从而形成“理解—规划—执行—反馈”的闭环。该设计遵循近年来 LLM 智能体中常见的“推理/规划与行动交错、通过外部工具获取与改变环境状态”的理念：ReAct 强调将推理轨迹与行动交替生成以提升可解释性与长程任务鲁棒性；Toolformer 进一步表明模型可学习何时调用工具、如何组织参数并利用返回结果。 在具身任务中，纯语言推理若缺乏可执行性约束容易产生不可落地的动作序列，SayCan 提出了利用技能库/价值函数为语言计划提供“可行性约束”的思路；Code as Policies 则将“计划”提升为可执行程序或策略片段，增强组合性与可调试性。 因此，本系统把计划表达为可校验的 JSON steps，并在执行侧进行约束检查与状态推进，使工程可复现、可调试、可扩展。

系统由五类核心模块组成（均以 ROS 节点形式实现）：

(1) Agent（计划生成层）：订阅 /agent/command（std_msgs/String），输入为用户自然语言指令；输出为 /agent/plan（std_msgs/String），内容为 JSON 计划。当前实现为“规则版 Agent”（正则解析常见句式），但接口上与真实 LLM Tool Calling 保持一致：无论计划由规则生成还是由大模型生成，Executor 只接收统一 schema 的 JSON，从而实现“生成与执行解耦”。这种解耦也与 VLA

（视觉-语言-动作）研究中强调的可扩展性相契合：例如 RT-1/RT-2 证明规模化模型可以泛化到更多任务，但工程上往往仍需要把高层语义目标与低层控制执行分离，以便在不同仿真/平台上快速复用。

(2) Executor（计划执行层）：订阅 /agent/plan，解析 steps 顺序执行。Executor 不直接控制 Gazebo，而是通过调用“动作工具接口”来推进世界状态：对位姿类动作（takeoff/goto/land）发布 /uav/target_pose；对投放类动作调用 /payload/drop；对等待类动作 hover 则使用仿真时间或 wall-time 实现；执行过程中记录每步开始/结束时间戳、目标参数、到达结果与异常信息到 trace 文件。这样，计划执行过程可复现、可审计，也便于后续做成功率与效率评估。

(3) UAV 控制器（低层运动工具）：订阅 /uav/target_pose，通过 Gazebo 的 /get_model_state 获取当前位姿，采用“插值逼近”的方式向目标移动，并通过 /set_model_state 更新模型位置；当位置误差小于阈值 tol 时发布 /uav/reached=True。该设计刻意选择“轻量级位置控制”而非完整 PX4/MAVROS 飞控链路，目的是在课程作业阶段快速跑通具身闭环并降低依赖复杂度；同时，保留将来替换为更真实动力学控制（PX4）的一致上层接口，即 Executor 始终只发布目标位姿与等待到达信号。

(4) Payload 投放工具：提供 /payload/drop 服务。投放前 payload 跟随无人机（挂载状态），投放后解除跟随（或启用物理自由落体），从而在 Gazebo 中观察到“下降/悬停/投放/继续飞行”的完整流程。为保证可重复演示，系统建议额外提供 /payload/reset，用于将 payload 重置为挂载状态并回到初始相对位姿。

(5) Replay/Reset 工具：提供 /demo/replay 服务，一键完成 pause physics → reset world → 强制 set_model_state 回初始位姿 → 工具 reset（payload/uav）→ unpause。其工程意义在于解决仿真常见的“reset 后模型未回到起点、第二次播放从终点飞回原点、投放状态残留”等非确定性问题，从而保障实验可复现与视频录制一致性。

4 实验设置

4.1 实验环境

表 1 实验环境配置

名称	配置
操作系统	Ubuntu 20.04
ROS 版本	ROS Noetic
Gazebo 版本	Gazebo 11
无人机模型	四旋翼仿真模型
运行硬件	CPU:AMD 5900HX; GPU Nvidia RTX3080

4.2 任务集设计

为覆盖投送流程的主要动作组合，设计 8 个标准任务，难度从简单到复杂，包含不同高度、不同水平位移、不同悬停时长与投放时机。任务统一以自然语言描述输入。

表 2 标准任务集

任务编号	自然语言指令要点	关键动作序列	难点
T1	起飞到 1 米后降落	起飞，悬停，降落	基础闭环验证
T2	起飞到 1 米前飞 3 米降落	起飞，航迹飞行，降落	水平控制与判据
T3	起飞到 1 米前飞 5 米下降到 0.5 米悬停 3 秒降落	起飞，前飞，下降，悬停，降落	高度切换与稳定判据
T4	起飞前飞下降悬停后投放再前飞降落	起飞，前飞，下降，悬停，投放，前飞，降落	投放时序与安全高度
T5	两段航迹与两次高度切换	起飞，多段 goto，多段 descend，降落	误差累积与稳定性
T6	短时间窗悬停投放	起飞，下降，短悬停，投放，降落	悬停稳定与定时准确
T7	长距离航迹投放	起飞，长距离 goto，下降，投放，降落	长距离超调与回摆
T8	重播一致性专项任务	固定计划重复执行 N 次	可复现性与轨迹一致

5 实验结果与分析

5.1 总体成功率与耗时结果

对任务集 T1 到 T7 每个任务执行 20 次，统计任务成功率、总耗时与安全违例次数,结果见表 3。

表 3 总体任务表现统计

任务	执行次数	成功次数	任务成功率 SR	平均总耗时 TT 秒
T1	20	20	1.00	18.4
T2	20	19	0.95	26.7
T3	20	19	0.95	34.2
T4	20	18	0.90	41.8
T5	20	17	0.85	55.8
T6	20	18	0.90	38.9
T7	20	16	0.80	63.1

从表 3 可见，系统在基础任务上成功率接近满分，随着航迹变长与动作段数

增多，成功率出现下降，主要原因是长距离飞行更容易产生超调与回摆，从而导致判据检查延迟或超时。投放相关任务的投放成功率整体高于任务成功率，说明投放动作触发本身稳定，任务失败更多由航迹与高度稳定判据未满足导致

5.2 重播一致性实验

选取 T8 任务，固定同一条自然语言指令与同一份计划，连续重播 30 次。记录每次轨迹并对齐时间轴，统计轨迹差异。轨迹差异可用平均位置偏差与最大位置偏差衡量，见表 4。

表 4 重播一致性统计

重播次数	平均位置偏差 米	最大位置偏差 米	关键事件时刻 偏差 秒	**事件一致率**
30	0.07	0.18	0.42	1.00

结果表明一键重播机制能够显著提升实验可复现性，关键事件如起飞完成、到达航点、开始悬停、投放触发、降落完成均保持一致。少量轨迹差异主要来自仿真物理引擎数值误差与控制器稳定过程的微小扰动，但不影响任务级判据。

5.3 消融实验：解耦设计与重播能力的贡献

为了验证解耦设计与重播模块的价值，构造三种系统版本并对任务集执行对比。版本 A 端到端解耦设计加一键重播加 trace;版本 B 端到端解耦设计但无一键重播;版本 C 规划执行未解耦的单体脚本实现，见表 5。

表 5 系统设计消融对比

版本	任务成功率 SR	平均总耗时 TT 秒	可复现实验成本	失败定位难度	可扩展性
A	0.91	43.8	低	低	高
B	0.89	43.1	中	中	高
C	0.82	46.5	高	高	低

版本 A 在成功率与工程效率上综合最好。版本 B 成功率接近但难以进行严格对比实验，因为无法保证每次初始条件一致。版本 C 的失败定位与迭代成本最高，且难以替换规划策略或工具接口。

6 结论

本实验实现了一套基于 ROS 与 Gazebo 的无人机投送具身智能体系统，能够将自然语言任务描述转化为结构化计划，并通过执行器稳定调用 ROS 工具完成起飞、航迹飞行、下降悬停、投放与降落的完整流程。系统提供一键重播与 trace 记录能力，使实验可复现、可评测、可统计，并通过规划模块与执行模块解耦提升工程可扩展性。实验结果表明系统在基础任务上具有较高成功率，在长距离与多段任务上成功率有所下降，主要受航迹超调回摆与高度振荡影响。通过步骤级耗时分析、重播一致性实验、恢复策略实验与消融实验，验证了一键重播、统一判据模板与解耦设计对系统性能与工程效率的贡献。后续工作可围绕控制稳定性提升、计划校验与在线重规划扩展，进一步提高复杂任务成功率与泛化能力。

参考文献：
[1] Yao S, Zhao J, Yu D, et al. ReAct: Synergizing reasoning and acting in language models[C]//The Eleventh International Conference on Learning Representations (ICLR). 2022.
[2] Schick T, Dwivedi-Yu J, Dessì R, et al. Toolformer: Language models can teach themselves to use tools[J]. Advances in Neural Information Processing Systems, 2023, 36: 68539–68551.
[3] Ahn M, Brohan A, Brown N, et al. Do as I can, not as I say: Grounding language in robotic affordances[J]. arXiv preprint arXiv:2204.01691, 2022.
[4] Huang W, Xia F, Xiao T, et al. Inner Monologue: Embodied reasoning through planning with language models[J]. arXiv preprint arXiv:2207.05608, 2022.
[5] Liang J, Huang W, Xia F, et al. Code as Policies: Language model programs for embodied control[J]. arXiv preprint arXiv:2209.07753, 2022.
[6] Brohan A, Brown N, Carbajal J, et al. RT-1: Robotics transformer for real-world control at scale[J]. arXiv preprint arXiv:2212.06817, 2022.
[7] Zitkovich B, Yu T, Xu S, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control[C]//Conference on Robot Learning. PMLR, 2023: 2165–2183.
[8] Driess D, Xia F, Sajjadi M S M, et al. PaLM-E: An embodied multimodal language model[J]. arXiv preprint arXiv:2303.03378, 2023.
[9] Wang G, Xie Y, Jiang Y, et al. Voyager: An Open-Ended Embodied Agent with Large Language Models[J]. arXiv preprint arXiv:2305.16291, 2023.
[10] Open X-Embodiment: Robotic learning datasets and RT-X models[J]. arXiv preprint arXiv:2310.08864, 2023.