1.实验名称及目的

1.1 实验名称

具身智能强化学习驱动的四旋翼加速度悬停控制（Genesis + RflySim 软件在环）

1.2 实验目的

1. 掌握从多环境仿真到策略训练的端到端强化学习流程（PPO）。
2. 理解加速度级控制与姿态/电机级 PID 间的层级关系与封装接口设计。
3. 会构建具身无人机仿真任务（状态/动作/奖励/终止条件）并注入噪声提升泛化。
4. 能够在软件在环（SIL）环境中加载并验证训练策略，实现 PX4 通信与实时性能观测。
5. 建立从高并行 GPU 训练到低并行 CPU 训练的样本等效与参数调优方法。

1.3 关键知识点与代码解析

以下知识点按模块拆分，每节包含原理说明、框架结构与核心代码引用（相对链接可跳转 `code/` 目录精简示例）。

1.3.1 强化学习任务建模（HoverEnv）

原理：将无人机悬停/点位跟踪视为马尔可夫决策过程 (MDP)。状态 s_t 包含相对位置、线速度与上一步动作；动作 $a_t \in \mathbb{R}^4$ 表示期望机体系加速度与偏航；奖励函数组合目标逼近、动作平滑、姿态约束与撞击惩罚：

$$r_t = w_{target} r_{target} + w_{smooth} r_{smooth} + w_{yaw} r_{yaw} + w_{crash} r_{crash}$$

并行环境加速采样，随机重采样目标点构成 curriculum。
结构要点：

- 初始化：构造 `gs.Scene` → 添加地面/目标/无人机 → 绑定控制器 `AttDroneController`。
- 观测构造：`[rel_pos, lin_vel, last_action]` 归一裁剪，避免梯度爆炸。
- 奖励：使用差分势函数 `||last_rel_pos||^2 - ||rel_pos||^2` 实现向目标靠近的稀疏奖励密集化。

- 终止：姿态越界、高度低于阈值或距离范围外、Episode 时间耗尽。
  核心代码片段参考真实文件 `train/env_acc.py` 与精简示例 `code/env_hover_min.py` 。

1.3.2 策略训练配置（PPO + Runner）

`train_acc.py` 中 `get_train_cfg` 定义 PPO 超参数：截断比例 `clip_param` 、自适应 KL、熵正则、学习率与多 epoch mini-batch 结构。并行环境数、每环境步数与迭代次数决定总样本量：

$$N_{samples} = N_{envs} \times N_{steps\_per\_env} \times N_{iterations}$$

多次修改 CPU 版参数以弥补并行度降低，保持统计充分性。
核心调用：

```
env = HoverEnv(...)
train_cfg = get_train_cfg(exp_name, max_iterations)
runner = OnPolicyRunner(env, train_cfg, log_dir, device=gs.device)
runner.learn(num_learning_iterations=max_iterations, init_at_random_ep_len=True)
```

1.3.3 加速度到电机的控制桥接（AttDroneController + PID）

高层策略输出期望加速度；控制器将其转换为姿态角，再通过多级 PID（角度 → 角速度 → 电机转速混控）生成四个电机 RPM。解析层级：

1. 速度/加速度闭环：由误差生成期望加速度/推力。
2. 期望加速度与当前偏航结合解算期望滚转/俯仰角 (ϕ_d, θ_d)。
3. 姿态 PID 输出期望角速度；角速度 PID 输出力矩；混控矩阵生成电机转速。
   电机混控形式（抽象）：

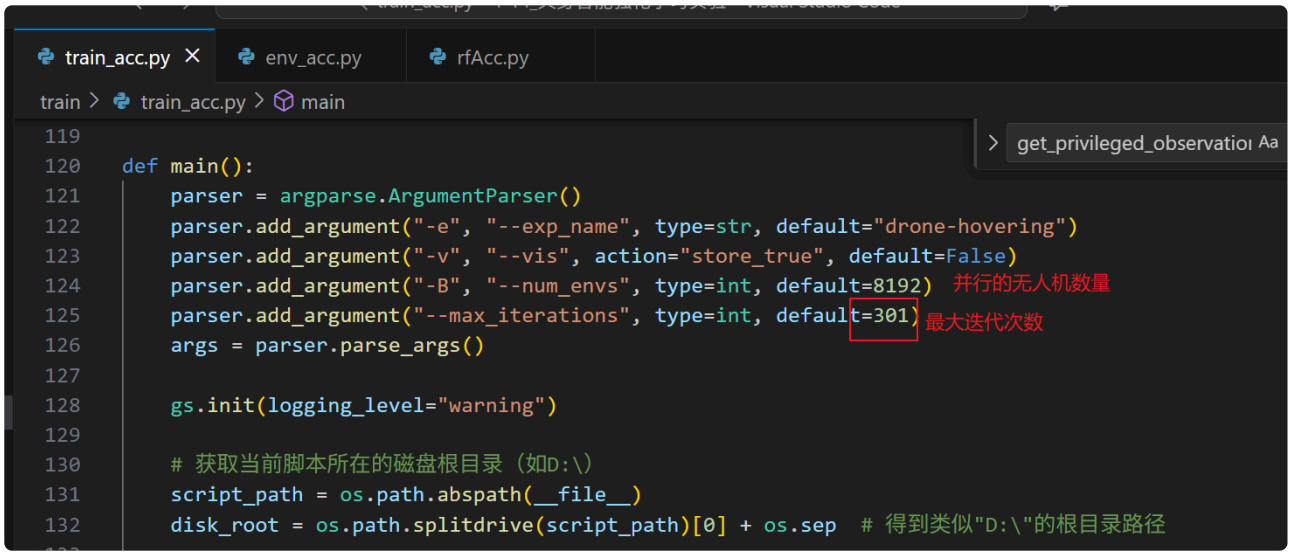
$$RPM_i = RPM_{base} \cdot (T \pm R \pm P \pm Y + 1)$$

   示例 PID 全量代码参阅 `train/att_thrust_drone.py` 。

1.3.4 域随机化与噪声注入

为提升策略泛化能力，在环境中对位置、速度、姿态加入高斯噪声（ `env_acc.py` 中的 `add_position_noise` ）。目的：缓解 sim-to-real 分布偏移，提高对传感器扰动与估计误差的鲁棒性。噪声幅度控制：位置 0.1、姿态 0.02、速度 0.1、角速度 0.01（单位见源码）。

1.3.5 采样效率与并行度映射



原 GPU 方案：8192 环境 × 100 步 × 301 迭代。CPU 降并行需提升单迭代步数与总迭代数。样本等效调参（示例）：

```
GPU: 8192 * 100 * 301 ≈ 245M samples
CPU:  512 * 400 * 1500 ≈ 307M samples (>125%)
```

保证优势函数估计方差下降速度与收敛稳定性。

1.3.6 软件在环验证（rfAcc.py）

流程：加载训练日志目录 → 读取 `cfgs.pkl` → 构建简化环境接口 `RflySimDrone` → 建立 PX4 Offboard 通信（MAVLink UDP） → 周期推理策略输出加速度 → 转换坐标系 NED→NWU → 发布给 PX4。性能指标：Step 频率（≈100Hz），稳态误差、反应时间。

2.实验效果

训练完成后策略在仿真中可稳定将无人机保持于动态采样目标点附近（平均距离 < 0.2 m），PX4 软件在环验证中可实现平稳悬停与快速小范围位置调整；终端周期输出控制频率（约 90–110 Hz），撞击与姿态越界事件显著减少，动作变化平滑，偏航角保持在安全范围内。

3.文件目录

例程根目
录：

`[RflySim安装路径]\RflySimAPIs\10.RflySimSwarm\3.CustExps\e0.CustApiExps\3.MARLExps`

序号	文件/目录	描述
2	train/att_thrust_drone.py	姿态与电机级 PID 混控封装
3	train/env_acc.py	强化学习环境定义（状态/奖励/终止）
4	train/train_acc.py	PPO 训练入口与配置保存
5	train/train_acc_cpu.py	CPU 版本入口
8	sil/rfAcc.py	软件在环验证脚本（PX4 + RflySim）
9	sil/view.py	视图/可视化辅助脚本
10	sil/RflyUdpMavlinkRealSim.bat	启动 RflySim UDP/MAVLink 仿真批处理
11	sil/Python38Run.bat	Windows 下 Python 环境启动批处理
12	sil/WinWSL2.bat	WSL2 辅助启动批处理

附：`train_acc.py` 与 `train_acc_cpu.py` 的主要区别

以下说明针对 `train/` 目录下的两份训练入口脚本，帮助选择合适脚本并了解它们的默认参数差异：

- 用途区别：
  - `train_acc.py`：面向通用（默认以 GPU 高并行为目标）的训练入口，适合在带有 CUDA 的服务器或工作站上运行以充分利用大规模并行环境（示例中默认并行环境数较大）。
  - `train_acc_cpu.py`：面向 CPU 或低算力环境的训练入口，含有针对单机 CPU 运行的参数预置（例如更小的并行数、更长的每环境步数或调整的超参数），便于在无 GPU 的工作站或笔记本上实验。
- 参数与实现差异（常见项）：
  - `num_envs`（并行环境数）：GPU 版本通常设置为数千以获得高吞吐；CPU 版本设置为几十到几百以匹配 CPU 内存与线程能力。
  - 采样策略：CPU 版本常通过增加 `num_steps_per_env` 或总迭代次数来保证样本总量与 GPU 训练可对比，从而尽量获得类似训练效果。
  - `gs.init` 后端：GPU 版会默认使用 `gs.init()` 的 GPU 后端（若可用），而 CPU 版会显式设置 `backend=gs.cpu`，避免自动选择不兼容的 CUDA 后端。
  - 日志与保存路径：两者一般共享相同的 `log_dir` 机制，但 CPU 版本常建议使用更小、易于调试的目录结构并开启更频繁的检查点保存以便断点续训。
- 何时使用：

    - 有可用 GPU（NVIDIA + 驱动、CUDA）：优先使用 `train_acc.py`，并根据 GPU 显存与 CPU 核心数调整 `-B/--num_envs`。
    - 无 GPU 或仅本地开发调试：使用 `train_acc_cpu.py` 或在 `train_acc.py` 中传入参数使其在 CPU 后端运行（例如 `--num_envs 512 --max_iterations 1500` 并在代码中 `gs.init(backend=gs.cpu)`）。
- 兼容性提示：
  若需要在同一机器上切换运行模式，推荐直接使用 `train_acc.py` 并通过命令行参数与环境变量控制后端和并行度，或维护两个轻微不同的脚本（已在仓库中保留 `train_acc_cpu.py` 以便参考历史参数与设置）。

4.运行环境

硬件（按需）：PC（>=16GB RAM）、可选 PX4 飞控或虚拟飞控、网络支持 UDP 通信。若扩展真实传感器，可接入 IMU、激光雷达或深度相机。
软件：

- RflySim 工具链（对应版本：含 UE4 场景与 PX4 仿真接口）。
- Python ⩾ 3.12（训练） / 可选 3.8 兼容生态；虚拟环境隔离推荐。
- genesis-world 0.3.4。
- rsl-rl-lib 2.2.4（替代 rsl_rl）。
- PyTorch（CPU 或 GPU，GPU 优先但示例兼容 CPU）。
- 依赖库：numpy (1.26.x 或 2.x 取决于子环境)、trimesh、pyvista、mujoco（若使用更复杂场景）、PX4MavCtrlV4、UE4CtrlAPI。

5.实验步骤

> 提示：若使用 CPU 低并行，需增加迭代与步数补偿样本量。

步骤 1：完成高样本训练

双击 `Python38Run.bat` 启动 Python 环境，执行：

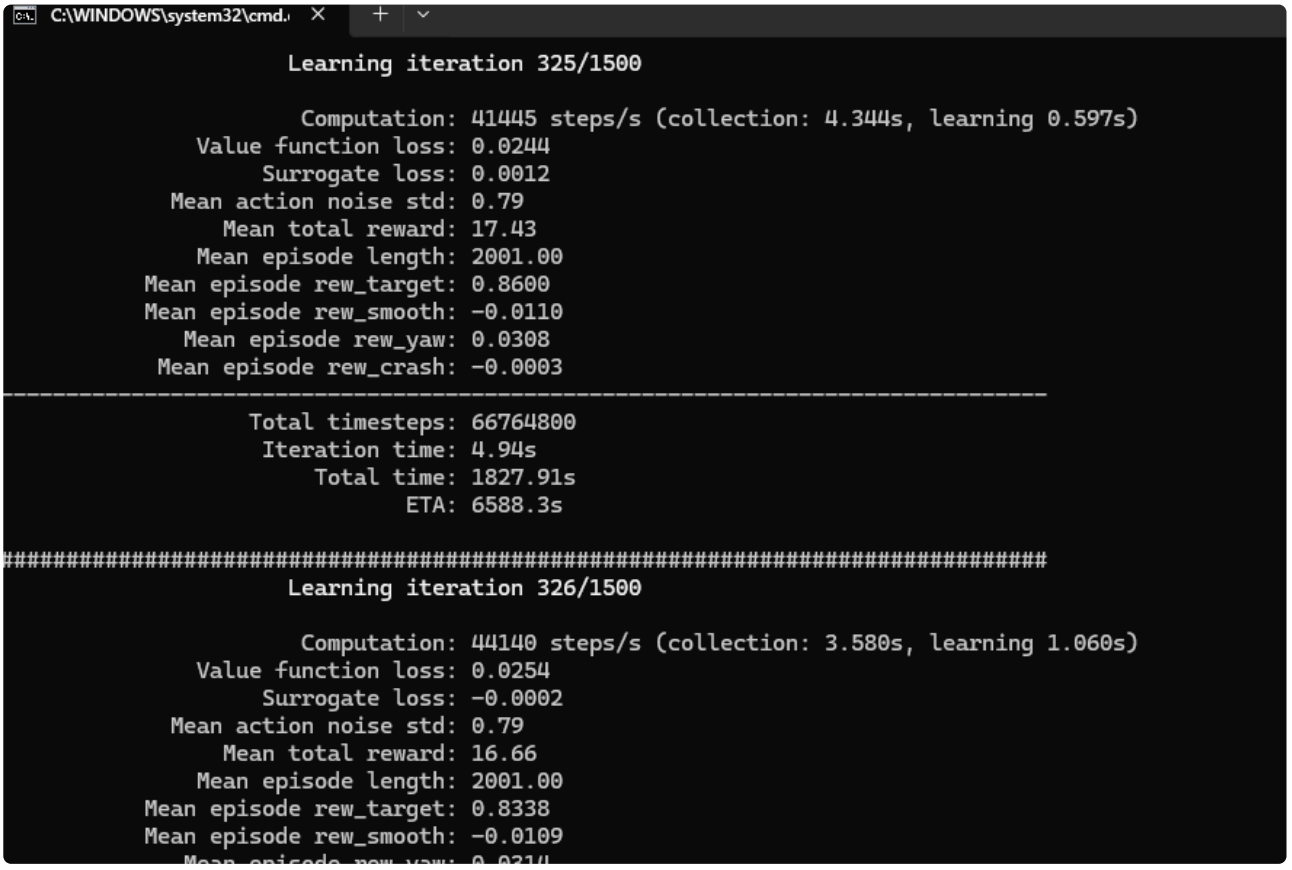
```
python train/train_acc_cpu.py
```

> [!TIP]

若已经安装了WSL2-GPU，参见例程`1.RflySimIntro\2.AdvExps\e11.WSL2_GPUAccConfig`，可以双击`WinWSL2.bat`执行如下脚本运行本例程的GPU版本

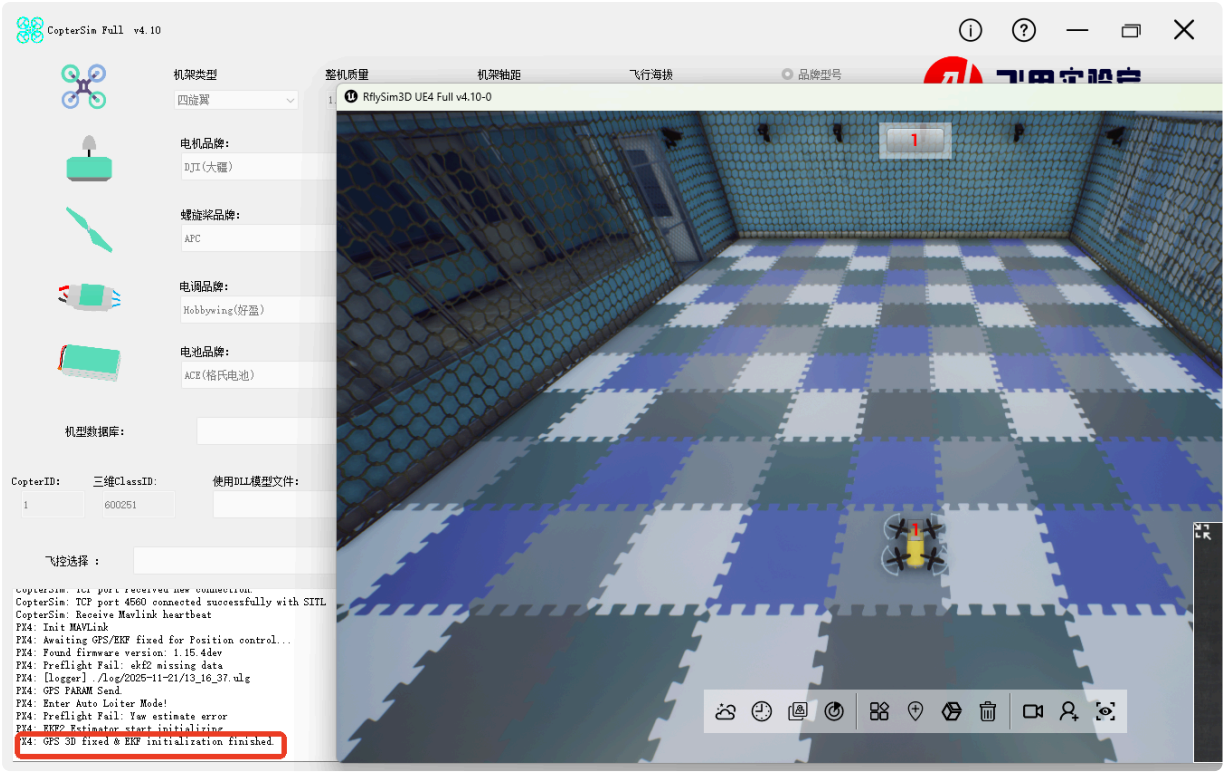
```
python3 train/train_acc.py
```

检查点：`[当前train_acc_cpu.py脚本所在磁盘]/tmp/logs_drone-hovering/model_*.pt` 生成；平均奖励随迭代上升稳定。



步骤 2：软件在环仿真验证训练效果

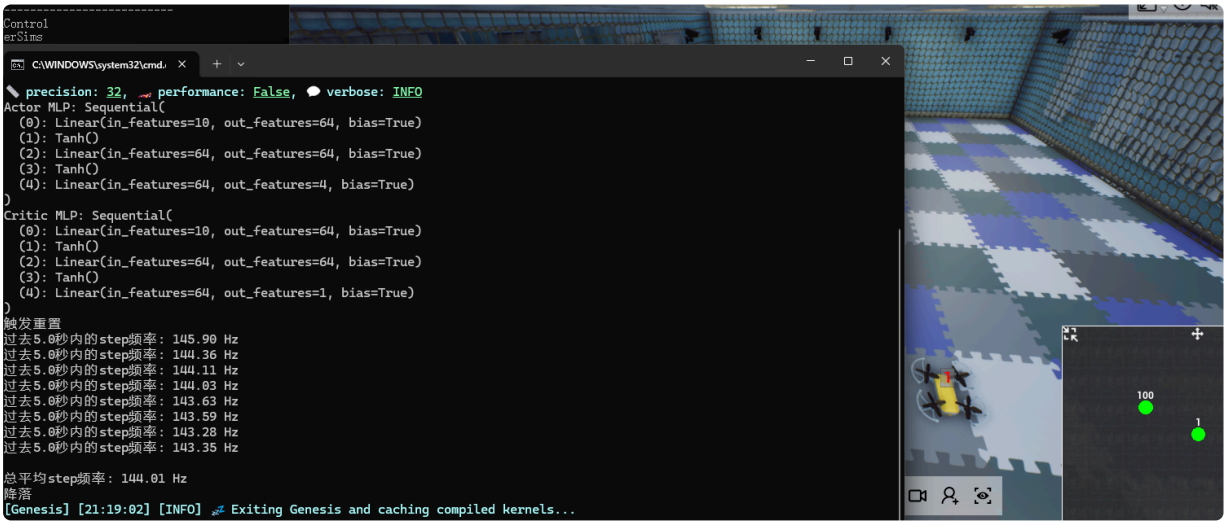
1. 运行 `sil/RflyUdpMavlinkRealSim.bat` 启动软件在环仿真，等待初始化完成。



2. 双击 `Python38Run.bat` 启动 Python 环境，执行：

```
python sil/rfAcc.py
```

3. 观察终端 Step 频率与无人机在虚拟场景中的无人机是否能及时跟上气球的位置。



7.常见问题

- 问题1：提示 “Please uninstall 'rsl_rl'...”
  - 解答：卸载旧包：`pip uninstall rsl_rl rsl-rl -y`，安装 `rsl-rl-lib==2.2.4`。
- 问题2：训练报 distutils 断言错误。

    - 解答：升级 setuptools 或设置环境变量：`SETUPTOOLS_USE_DISTUTILS=stdlib`。
- 问题3：SIL 运行报 UDP 端口占用 (Address already in use)。
    - 解答：确认无残留 PX4/RflySim 进程，使用 `netstat -ano` 释放端口或更换端口号。
- 问题4：策略不收敛或奖励停滞。
    - 解答：减小学习率（如 3e-4→1e-4），增加 `num_steps_per_env` 或检查奖励缩放与观测归一化。
- 问题5：动作抖动明显。
    - 解答：提高 `smooth` 惩罚权重绝对值或在 PID 层增加微分滤波。

6.参考资料

- Sutton & Barto. Reinforcement Learning: An Introduction (2nd Ed.).
- PPO 原论文：Schulman et al. “Proximal Policy Optimization Algorithms”. https://arxiv.org/abs/1707.06347
- PX4 官方文档：https://docs.px4.io/
- MAVLink 协议规范：https://mavlink.io/en/
- Genesis 框架文档（项目主页或官方 README）。