1. 实验名称及目的

1.1 实验名称

基于平台odom的无人机激光雷达实时三维点云建图实验

1.2 实验目的

本实验旨在利用 RflySim 仿真平台提供的无人机传感器数据（LiDAR 与 里程计），在 ROS 环境下实现点云数据的坐标系转换与拼接。通过构建 TF 变换树和齐次变换矩阵，将机身坐标系下的局部点云实时映射到世界坐标系中，从而在 Rviz 中构建并显示连贯的 3D 实体地图。

1.3 关键知识点

1）多坐标系转换系统 (Coordinate Systems)

无人机系统中存在三个核心坐标系，本实验重点解决它们之间的转换与对齐：

1. **仿真坐标系 (NED/Left-Hand):** RflySim/UE4 内部使用的NED。
2. **ROS 世界坐标系 (ENU/Right-Hand):** Rviz 与 ROS 标准的ENU。
3. **机身坐标系 (Body Frame):** 随无人机运动的坐标系，前方为 X，左方为 Y，上方为 Z。

实验通过读取 `/rflysim/uav1/local/odom` 话题（通常已转换为 ENU 或需简单映射），获取无人机在世界系下的位姿，从而消除坐标系定义差异带来的“重影”或“镜像”问题。

2）齐次变换矩阵与 TF 树 (Transformation Matrix & TF Tree)

为了将雷达扫描到的局部点 P_{body} 转换到全局地图点 P_{map}，我们需要构建 4×4 的齐次变换矩阵 T：

$$P_{map} = T_{map \leftarrow body} \cdot P_{body}$$

其中 T 由旋转矩阵 R (来自四元数) 和平移向量 t (来自里程计位置) 组成：

$$T = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ 0 & 1 \end{bmatrix}$$

ROS 的 **TF (Transform)** 系统负责维护 `map` -> `base_link` -> `lidar` 的变换树，确保数据在不同时间戳下的空间一致性。

1.4 关键代码

本实验包含三个核心脚本，实现了从数据获取到地图构建的全流程。

1) 接口使用 (`main.py`)

```
req = ReqCopterSim.ReqCopterSim() \# 获取局域网内所有CopterSim程序的电脑IP列表
TargetIP = req.getSimIpID(StartCopterID) \#自动获取CopterSim的StartCopterID号程序所在电脑的IP，作为目标IP。
这里获取CopterSim所在仿真电脑的IP。
vis = VisionCaptureApi.VisionCaptureApi(TargetIP) \#创建一个视觉传感器实例，这个实例对应的ip号为TargetIP
req.sendReSimIP(CopterID) \# 请求mavlink数据到本电脑
```

2) 建图 (`mapping.py`)

这是本实验的核心算法文件，相比于旧版 `mapping.py`，进行了以下重大改进：

- **时间同步 (`ApproximateTimeSynchronizer`)**：

  ```
  ts = ApproximateTimeSynchronizer([odom_sub, cloud_sub], queue_size=10,
  slop=0.1)
  ts.registerCallback(sync_callback)
  ```

  强制将同一时刻的里程计 (Odom) 和雷达点云 (Lidar) 配对，防止因时间戳不对齐导致的地图"漂移"。

- **向量化计算 (Vectorization)**：

  ```
  # 构造变换矩阵 T (4x4)
  transform_matrix = np.eye(4)
  transform_matrix[:3, :3] = rotation_matrix[:3, :3]
  transform_matrix[:3, 3] = [tx, ty, tz]

  # 批量矩阵乘法 (N x 4)
  xyz_map_homo = np.dot(transform_matrix, point_lidar.T).T
  ```

  利用 NumPy 直接对点云数组进行变换，解决了 Python 处理点云卡顿的问题。

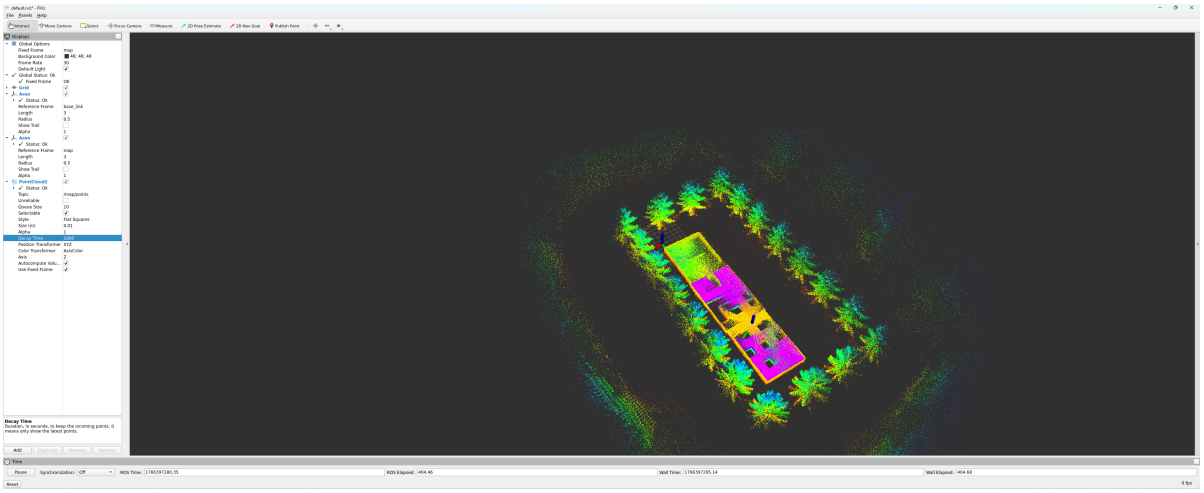
- **TF 广播**：
  实时发布 `odom1` (或 `map`) 到 `base_link` 的 TF 变换，使得 Rviz 可以正确显示无人机的运动姿态。

---

2. 实验效果

3.1 实体地图构建

在 Rviz 中，通过将 `PointCloud2` 的 **Decay Time** 设置为较大值（如 1000），可以实现"视觉暂留"效果。随着无人机飞行，雷达扫描过的区域被保留在屏幕上，形成完整的 3D 栅格化地图结构。



3. 文件目录

文件名称	说明
`SITLRun.bat`	仿真平台运行脚本
`WinWSL.bat`	WSL1/Ubuntu 20.04环境程序运行脚本
`main.py`	仿真接口启动脚本，负责连接 CopterSim 并开启数据
`mapping.py`	建图脚本
`odom2mavros.py`	里程计转换脚本，用于适配 MAVROS 控制接口
`Readme.md`	实验说明文档

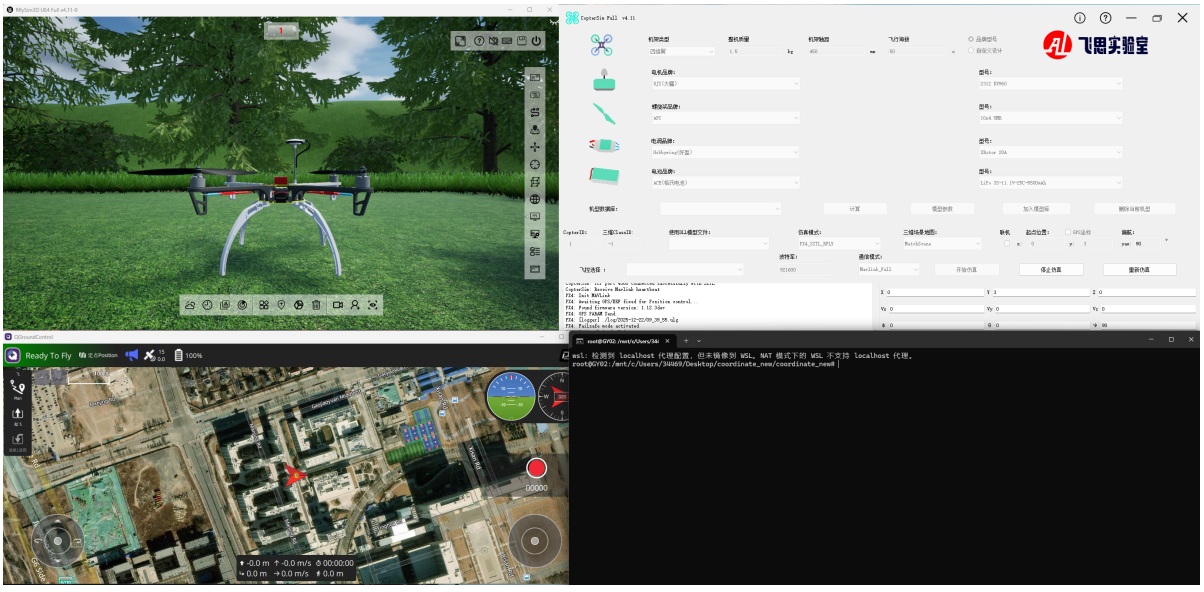
4. 运行环境

项目	要求
操作系统	Ubuntu 20.04 (Noetic) 或 Windows WSL + ROS Noetic
仿真软件	RflySim工具链

5. 实验步骤

Step 1: 启动仿真环境

双击运行coordinate_new文件夹下的SITLRun.bat，启动Rflysim仿真平台，将会启动1个QGC地面站，1个CopterSim软件且其软件下侧日志栏必须打印出GPS 3D fixed & EKF initialization finished字样代表初始化完成，并且有1个RflySim3D软件有1架无人机；再点击WinWSL.bat，将会启动一个WSL终端，如下图所示：

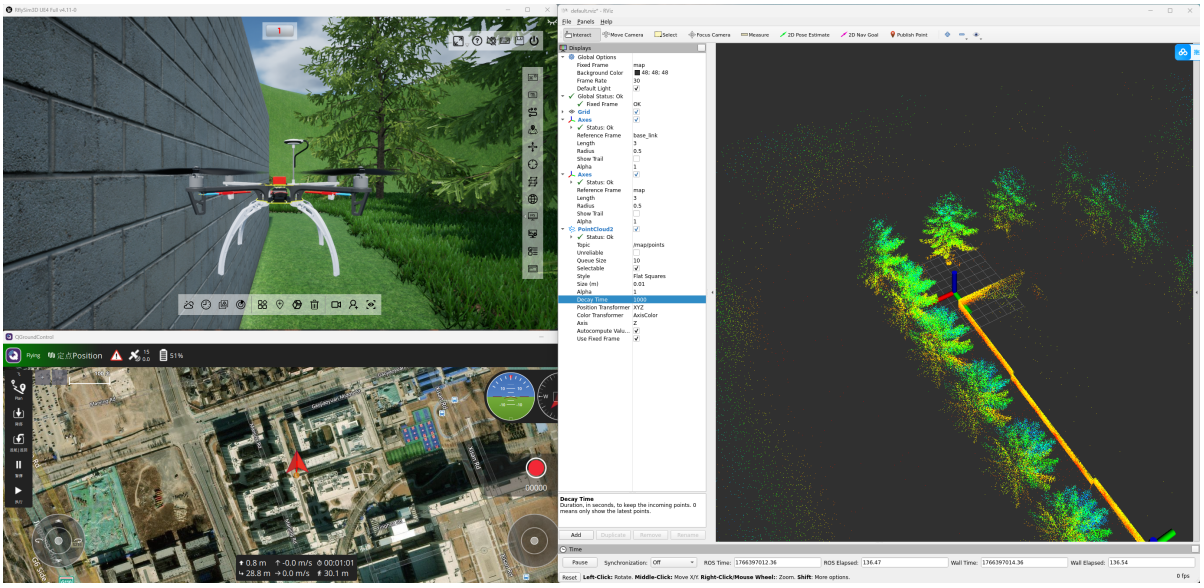


Step 2: 运行程序

开两个终端，分别依次输入python3 main.py 和 python3 mapping.py

Step 3: 观察效果

再开一个终端，输入rviz启动rviz。进入rviz界面后点击左下角“add”键，依次添加“Axis”，“Axis”和“PointCloud2”，前两个Axis的“Topic”分别选择“base_link”和“map”，“PointCloud2”下的“Topic”选择“/map/points"，再把”Decay Time"设置为1000。最后在QGC中起飞无人机并控制无人机绕场景旋转可以在rviz中看到实时建立的三维点云地图，如下图所示：



6.参考资料

无

7.常见问题

无