基于odom的无人机激光雷达实时三维点云建图实验

1. 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。本实验将帮助学生深入理解无人机多传感器融合、坐标变换、实时点云处理和ROS系统集成等关键技术，掌握基于里程计的三维建图方法。

2. 实验要求

- 软件要求：Windows 10及以上版本；RflySim工具链[1]。
- 硬件要求：笔记本/台式电脑1台[2]。

3.实验地址

例程目录：[安装目录]\RflySimAPIs\8.RflySimVision\2.AdvExps\e14_odomLiDAR3DPc

例程介绍：

- ./Config.json：传感器配置文件
- ./SITLRun.bat：SITL仿真环境启动脚本
- ./WinWSL.bat：WSL环境启动脚本
- ./WinWSLRunALL.bat：一键启动所有Python脚本
- ./main.py：主程序文件，负责连接仿真环境并获取传感器数据
- ./mapping.py：点云建图程序，处理里程计和点云数据
- ./odom2mavros.py：里程计转换程序，将坐标转换至Mavros坐标系
- ./view_pcd.py：点云查看程序，用于查看保存的点云数据
- ./rfly.rviz：Rviz配置文件

- ./tf_cfg.yaml： TF配置文件
- ./StartRviz.sh： 启动Rviz脚本
- ./Python38Run.bat： Python运行脚本
- ./UavCtrl.py： 无人机控制程序

4. 实验内容或步骤

4.1 步骤1、启动仿真环境

双击运行SITLRun.bat文件，等待仿真环境初始化完成。脚本将会启动 1 个 QGC 地面站、1 个 CopterSim 、1 个 RflySim3D 软件。等待CopterSim软件下侧日志栏打印出 `GPS 3D fixed & EKF initialization finished` 字样，代表初始化完成。如下图所示：

> **注意**：必须确认仿真环境完全初始化后，再进行后续步骤。

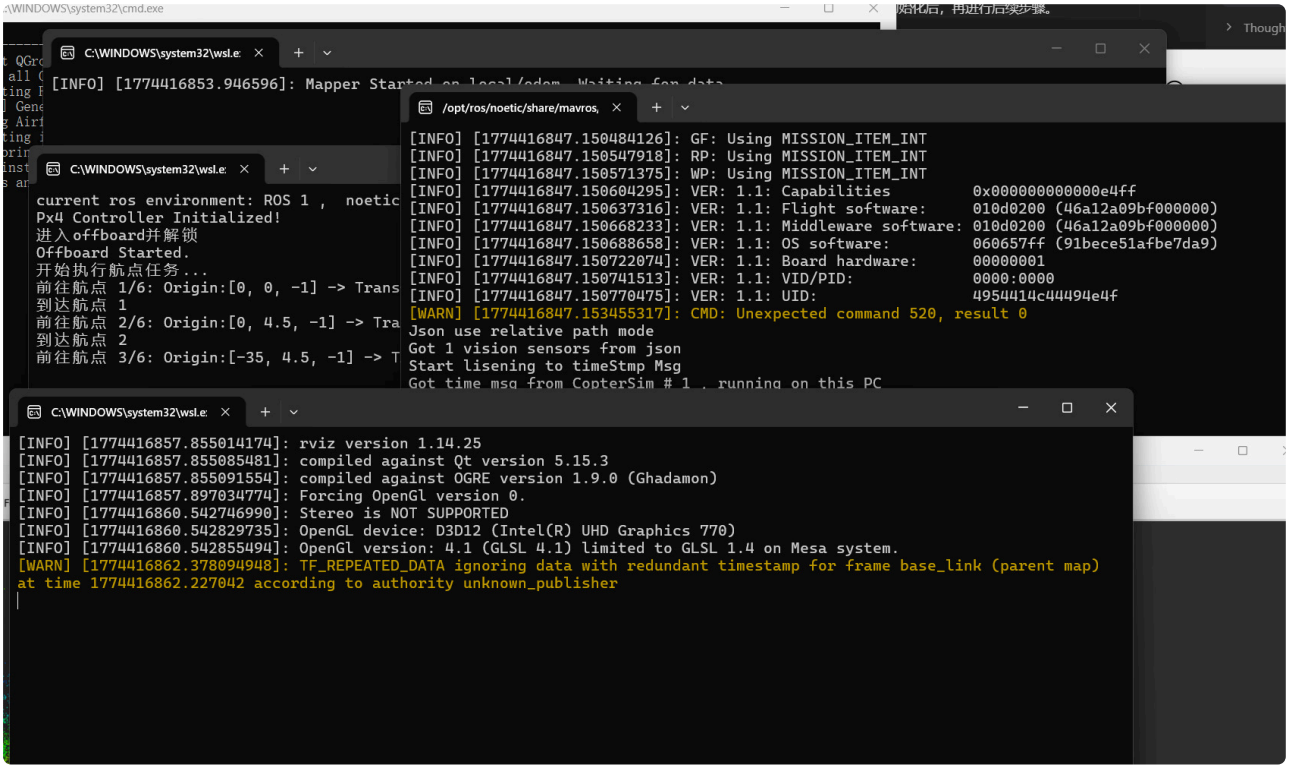
4.2 步骤2、启动实验程序

本实验提供 **两种运行方式**，任选其一即可。

方式一：一键启动（推荐）

双击 WinWSLRunALL.bat 脚本，将自动按顺序启动所有程序：

1. **main.py** — 传感器数据采集（等待15秒初始化）
2. **mapping.py** — 点云建图节点（等待5秒初始化）
3. **StartRviz.sh** — RViz 可视化界面（等待3秒初始化）
4. **UavCtrl.py** — 无人机航点飞行控制



启动后会自动打开多个WSL终端窗口，每个窗口运行一个程序。

方式二：手动分步启动

如果需要单独调试或灵活控制各模块的启动顺序，可以手动启动。先双击 WinWSL.bat 打开 **4 个WSL终端窗口**，然后按以下顺序依次执行：

终端1 — 启动传感器数据采集：

```
python3 main.py
```

等待出现 `Start Image Reciver` 提示，确认传感器数据流正常。

```
/opt/ros/noetic/share/mavros,
 * /mavros/wheel_odometry/frame_id: odom
 * /mavros/wheel_odometry/send_raw: True
 * /mavros/wheel_odometry/send_twist: False
 * /mavros/wheel_odometry/tf/child_frame_id: base_link
 * /mavros/wheel_odometry/tf/frame_id: odom
 * /mavros/wheel_odometry/tf/send: False
 * /mavros/wheel_odometry/use_rpm: False
 * /mavros/wheel_odometry/vel_error: 0.1
 * /mavros/wheel_odometry/wheel0/radius: 0.05
 * /mavros/wheel_odometry/wheel0/x: 0.0
 * /mavros/wheel_odometry/wheel0/y: -0.15
 * /mavros/wheel_odometry/wheel1/radius: 0.05
 * /mavros/wheel_odometry/wheel1/x: 0.0
 * /mavros/wheel_odometry/wheel1/y: 0.15
 * /rosdistro: noetic
 * /rosversion: 1.16.0

NODES
  /
    mavros (mavros/mavros_node)

auto-starting new master
process[master]: started with pid [854]
ROS_MASTER_URI=http://198.18.0.1:11311

setting /run_id to 3c0dc754-280c-11f1-b347-bbab6336ca59
process[rosout-1]: started with pid [882]
started core service [/rosout]
process[mavros-2]: started with pid [886]
[INFO] [1774416845.944265490]: FCU URL: udp://:20101@127.0.0.1:20100
```

终端2 — 启动点云建图节点：

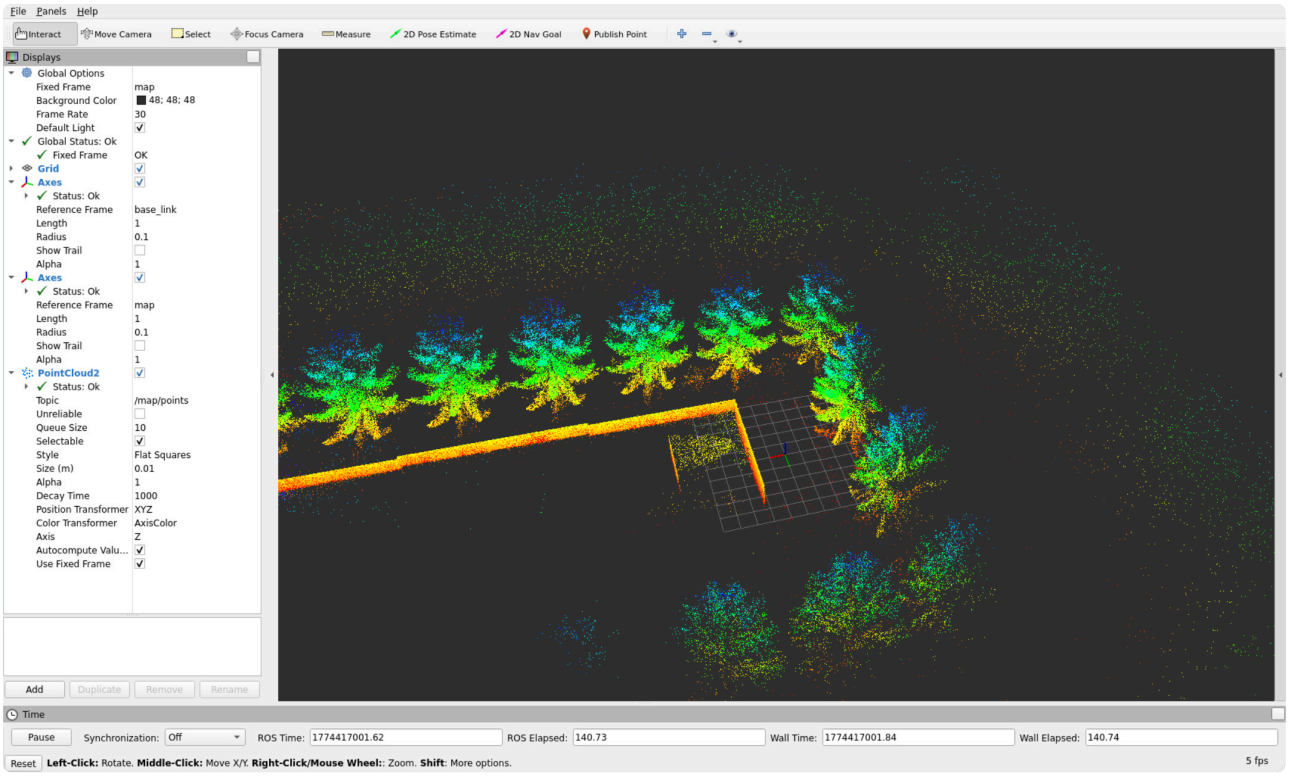
```
python3 mapping.py
```

等待出现 `Mapper Started on local/odom. Waiting for data...` 提示。

```
C:\WINDOWS\system32\wsl.e
[INFO] [1774416853.946596]: Mapper Started on local/odom. Waiting for data...
```

终端3 — 启动 RViz 可视化：

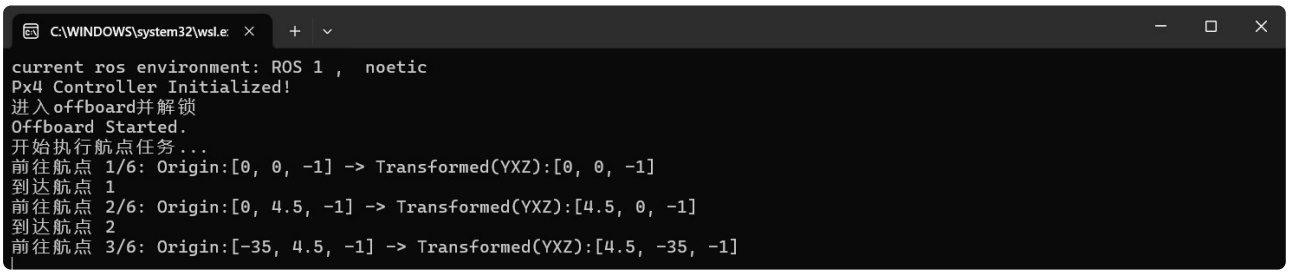
```
bash StartRviz.sh
```



终端4 — 启动无人机飞行控制：

```
python3 UavCtrl.py
```

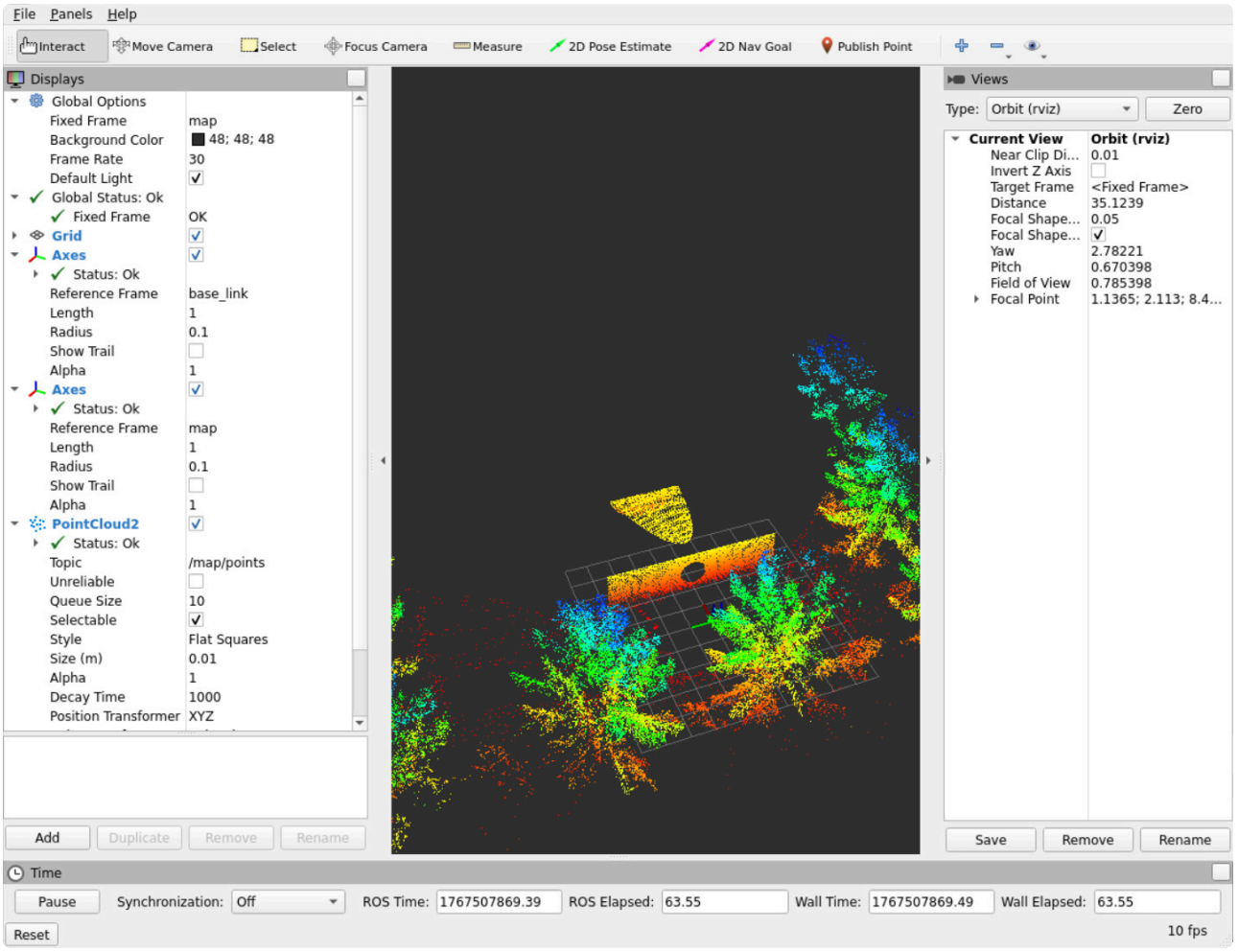
执行后无人机将进入 Offboard 模式并自动按预设航点飞行。



> **注意**：手动启动时，请确保严格按照上述顺序执行。`mapping.py` 必须在 `main.py` 之后启动（需要接收传感器数据），`UavCtrl.py` 应最后启动（确保建图节点已就绪）。

4.3 步骤3、Rviz 查看建图效果

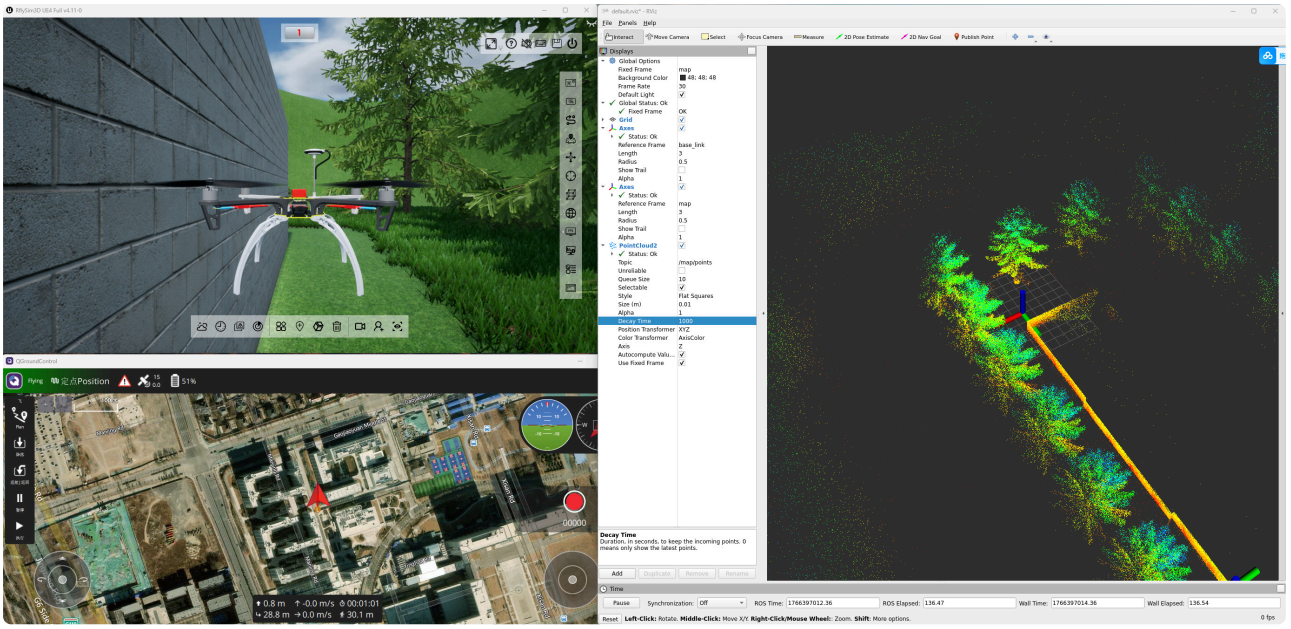
RViz 启动后将自动加载 rfly.rviz 配置文件，显示实时点云地图。



> **手动配置 RViz**：也可以直接输入 `rviz` 打开后手动配置。在 RViz 界面点击左下角 "Add" 键，依次添加 "Axes"、"Axes"、"PointCloud2"。前两个 Axes 的 "Reference Frame" 分别选择 "base_link" 和 "map"，"PointCloud2" 下的 "Topic" 选择 "/map/points"，再把 "Decay Time" 设置为 1000，"Color Transformer" 设置为 AxisColor。

4.4 步骤4、控制无人机飞行建图

无人机启动后将自动按照预设航点环绕建筑物飞行，同时在 RViz 中可以看到实时建立的三维点云地图：



也可以通过 QGC 地面站手动控制无人机飞行，自由探索建图区域：

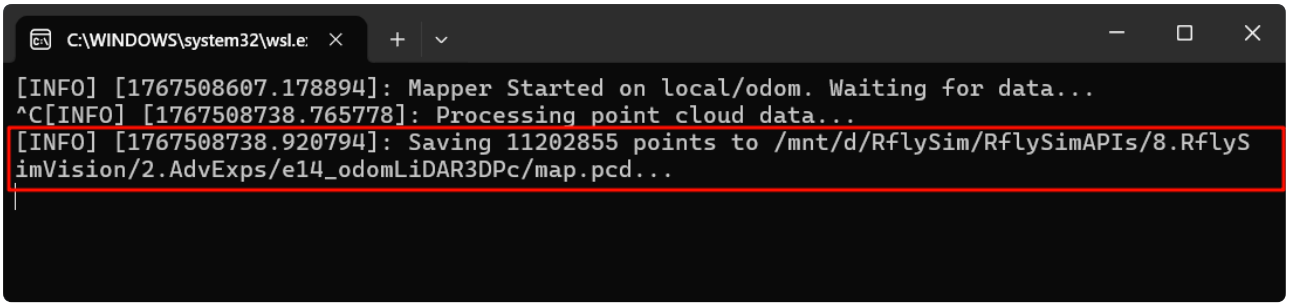
```
C:\WINDOWS\system32\wsl.e
current ros environment: ROS 1 ,  noetic
Px4 Controller Initialized!
进入offboard并解锁
Offboard Started.
开始执行航点任务...
前往航点 1/6: Origin:[0, 0, -1] -> Transformed(YXZ):[0, 0, -1]
到达航点 1
前往航点 2/6: Origin:[0, 4.5, -1] -> Transformed(YXZ):[4.5, 0, -1]
到达航点 2
前往航点 3/6: Origin:[-35, 4.5, -1] -> Transformed(YXZ):[4.5, -35, -1]
```


```
C:\WINDOWS\system32\wsl.e
current ros environment: ROS 1 ,  noetic
Px4 Controller Initialized!
进入offboard并解锁
Offboard Started.
开始执行航点任务...
前往航点 1/6: Origin:[0, 0, -1] -> Transformed(YXZ):[0, 0, -1]
到达航点 1
前往航点 2/6: Origin:[0, 4.5, -1] -> Transformed(YXZ):[4.5, 0, -1]
到达航点 2
前往航点 3/6: Origin:[-35, 4.5, -1] -> Transformed(YXZ):[4.5, -35, -1]
```


4.5 步骤5、保存与查看结果

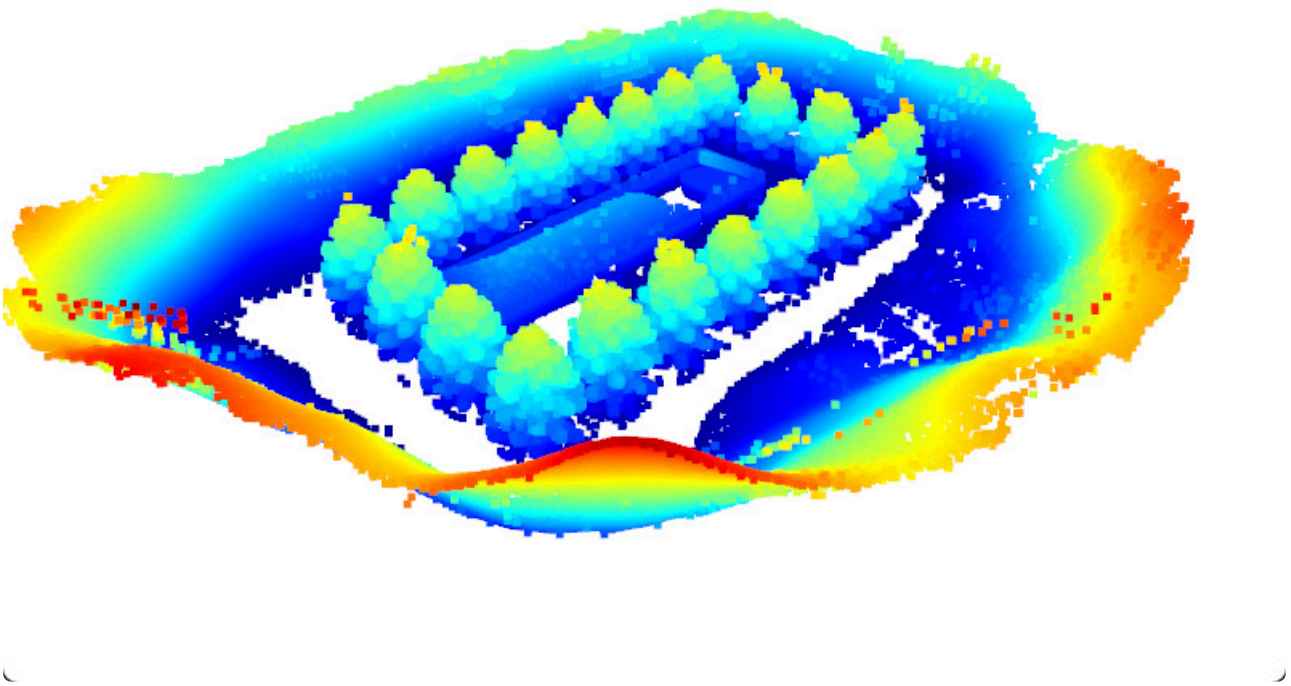
等待飞行完成后，在运行 `mapping.py` 的终端窗口中，按下 `Ctrl+C` 键停止建图。程序会将累积的点云数据自动保存为 `./map.pcd` 文件。

```
C:\WINDOWS\system32\wsl.e
[INFO] [1767507697.367440]: Mapper Started on local/odom. Waiting for data...
```



最后，双击 Python38Run.bat 脚本，输入 `python view_pcd.py` 命令，即可使用 Open3D 查看保存的点云数据。如下所示：

> **注意**：点云数据较大，请耐心等待读取。



5. 关键知识点

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。

关键知识点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。

关键知识点3：实验整体思路和框架

本实验通过三个核心程序实现无人机激光雷达实时三维点云建图：

- main.py：负责连接RflySim仿真环境，获取无人机传感器数据
- mapping.py：核心建图算法，处理里程计和点云数据，实现坐标转换和点云拼接
- odom2mavros.py：处理坐标系转换，将里程计数据适配到Mavros系统

6.参考资料

1. RflySim官方文档
2. ROS官方文档
3. PCL(Point Cloud Library)官方文档
4. TF坐标变换系统详解

7.常见问题

Q1：运行程序后，Rviz中无法显示点云数据。

A1：首先检查是否正确启动了SITLRun.bat和WinWSL.bat，确认仿真环境正常运行。其次，检查WSL终端中Python脚本是否正确执行，确保`python3 main.py`和`python3 mapping.py`命令运行无错误。最后，在Rviz中检查是否正确添加了PointCloud2话题，并且话题名称为`/map/points`。

Q2：Rviz中点云显示异常，出现重影或漂移现象。

A2：这通常是由于里程计数据与点云数据时间同步不准确导致的。检查mapping.py中的时间同步参数`slop`，可适当调整该值。另外，确保无人机运动速度不要太快，避免数据采集频率跟不上导致的数据不一致。

Q3：程序运行时报错，提示无法连接到仿真环境。

A3：检查SITLRun.bat是否已成功启动，确认CopterSim软件日志中出现`GPS 3D fixed & EKF initialization finished`提示。另外，确认main.py中的TargetIP设置正确，如果是本机运行则IP应为127.0.0.1，如果是远程访问则需要设置为正确的局域网IP地址。

---

1. https://rflysim.com/ ↩
2. 推荐配置请见： https://rflysim.com/doc/zh/HowToInstall.pdf ↩