1.实验名称及目的

1.1 实验名称

ORB-SLAM3 在 RflySim 仿真数据下的集成与验证（ROS1 节点）

1.2 实验目的

- 理解如何将 ORB-SLAM3 通过 ROS 节点和 RflySim 数据流（图像与 IMU）集成。
- 能够在本地构建并运行 ORB-SLAM3 ROS 节点，采集轨迹与位姿发布。

1.3 关键知识点

下面把三个常见且关键的知识点拆开说明，便于快速上手与排查问题。

A. ROS 节点与话题（image / imu）订阅与发布

- 常见消息类型：
  - 图像：`sensor_msgs/Image`（彩色通常为 RGB8，深度常用 32FC1 或 16UC1），使用 `cv_bridge` 或 `image_transport` 在节点内转换与订阅。
  - IMU：`sensor_msgs/Imu`（包含加速度、角速度与协方差矩阵）。
  - 相机位姿（可选）：`geometry_msgs/PoseStamped` 或专用轨迹文件输出。
- 话题命名与约定（建议）：
  - 单目：`/camera/image_raw` 或 `/rflysim/sensor1/img_rgb`
  - 双目：`/camera/left/image_raw` 与 `/camera/right/image_raw`
  - IMU：`/imu` 或 `/rflysim/imu`
  - 使用一致的 `header.frame_id` 与 TF 发布，保证 RViz 与 ORB-SLAM3 viewer 的坐标一致。
- 同步与缓冲：
  - 当同时使用图像与 IMU（或左右双目）时，通常需要时间同步。常用方法：`message_filters::TimeSynchronizer` 或 `message_filters::ApproximateTime`（C++/Python）。
  - 注意 IMU 频率与配置文件中的 `IMU.Frequency` 一致；IMU 通常为高频（如 200Hz），图像为低频（如 30Hz），节点需维护 IMU 缓冲并按图像时间戳回放 IMU 测量。
- 质量/性能建议：
  - 话题队列（queue_size）不宜过小（导致丢帧）也不宜过大（导致延迟）。
  - 在带宽受限的情况下使用 `image_transport` 的压缩（`compressed`）或 `theora`，但压缩会增加延迟与 CPU 负担。

示例（在 ROS 中检查数据流）：

```
# 列出话题
rostopic list
# 查看图像消息头信息
rostopic echo /camera/image_raw/header
```

B. ORB-SLAM3 配置文件（`orb_slam3/config/*.yaml`）字段与调参要点

常见配置模块及含义：

- Camera（相机内参与图像参数）
  - fx, fy, cx, cy：针孔相机的内参。
  - width, height, fps：图像尺寸与帧率（必须与仿真或摄像头一致）。
  - 畸变系数（k1,k2,p1,p2, etc.）或鱼眼参数（Kannala-Brandt）视镜头模型而定。
- ORB 提取器（特征提取）
  - nFeatures：每层最大特征数（增大可提高匹配率，但增加计算量）。
  - scaleFactor：金字塔缩放因子（通常 1.2）；较小值提高尺度连续性但增加层数计算量。
  - nLevels：金字塔层数。
  - iniThFAST / minThFAST：FAST 检测阈值，低纹理场景可降低阈值以检测更多特征。
- IMU（仅在惯性模式）
  - NoiseGyro / NoiseAcc：陀螺与加速度测量噪声（单位视配置而定），影响滤波与初值估计。
  - GyroWalk / AccWalk：随机游走噪声参数。
  - Frequency：IMU 采样频率（必须与发布频率一致，例如 200）。
  - Tbc / T_cb：相机与机体/IMU 的坐标变换（变换矩阵或四元数），若不准确会导致位姿偏差。
- 其它（Depth、Stereo、Viewer）
  - DepthMapFactor：深度图尺度（用于 RGB-D）。
  - Stereo.ThDepth：双目深度阈值。
  - Viewer 相关：用于调试显示参数（是否显示轨迹/关键帧/局部地图）。

调参建议（经验法）：

- 初始步骤：使用一个已知有效的 settings（例如 EuRoC / TUM-VI 示例），确认系统能运行；再替换为你的相机/IMU 参数并逐步调参。
- 若跟踪失败或漂移大：
  - 检查相机内参与 distortion 是否正确（相机没有正确标定会导致定位失败）。
  - 确认时间戳同步（IMU 与图像时间戳一致且单位相同）。
  - 对低纹理场景，降低 FAST 阈值并增加 nFeatures；注意计算开销。
- 惯性初始化问题：IMU 噪声参数设置过小/过大都会影响初始化收敛，优先使用仿真或硬件手册的标称噪声做初值，再根据轨迹鲁棒性微调。

配置文件位置提示：

- 仓库中通常有 `ubuntu/orb_slam3/ROS1/src/orb_slam3/config/*.yaml` 或 `orb_slam3/config/*.yaml`（视项目组织）。启动时通过 `config_path` 参数传入对应 YAML 文件。

C. 使用 VisionCaptureApi / `sensor_pkg` 将图像与 IMU 发送到 ROS

主要流程（`example.py` 的典型步骤）：

1. 加载配置：脚本会以当前脚本目录为基准尝试加载 `Config.json`（如果找不到会尝试当前工作目录）。配置中 `VisionSensors` 数组定义了每个相机的 SeqID、分辨率、频率与 `SendProtocol`（回传 IP 与端口）。
2. 启用 ROS 转发：设置 `VisionCaptureApi.isEnableRosTrans = True`，使视觉采集库在本地把接收到的数据发布为 ROS 话题。
3. 请求仿真发送数据：调用 `vis.sendReqToUE4()` 与 `vis.sendImuReqCopterSim()`，并通过 `vis.startImgCap()` 启动图像接收循环。

关键字段说明（`Config.json`）示例：

```
{
	"VisionSensors": [
		{
			"SeqID": 0,
			"TypeID": 3,
			"TargetCopter": 1,
			"DataWidth": 640,
			"DataHeight": 480,
			"DataCheckFreq": 30,
			"SendProtocol": [1,127,0,0,1,9998,0,0]
		}
	]
}
```

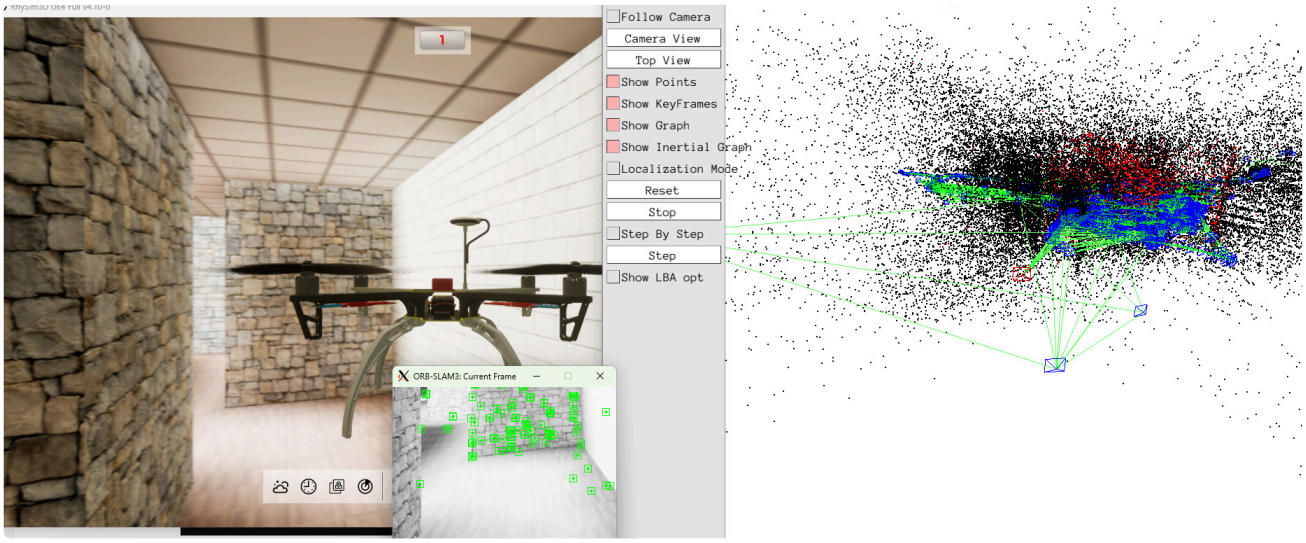
- `SendProtocol` 中通常包含回传协议标识、目标 IP（127.0.0.1 表示本机）与端口（如 9998），需要与仿真端的发送端配置对应。

运行与排查要点：

- 在 Windows 上运行 `example.py`（或在 WSL + ROS 环境中），确保防火墙允许所用端口，并且仿真（CopterSim）已经启动并能响应请求。
- 检查 ROS 话题：运行 `rostopic list` 与 `rostopic echo <topic>`（或用 `rqt_image_view` / `image_view` 查看图像）。
- 若出现丢帧或延迟：确认网络延迟、端口冲突与脚本中的 `DataCheckFreq` 与 `IMU.Frequency` 配置一致。

2.实验效果

- 成功接收 RflySim 中的图像与 IMU 数据，并在 ROS 中启动 ORB-SLAM3 节点。
- ORB-SLAM3 在 RViz/Viewer 中可视化关键帧与位姿轨迹。



3.文件目录

例程目录：[安装目录]\RflySimAPIs\8.RflySimVision\3.CustExps\e6_ORB-SLAM3

序号	路径	说明
1	`./ubuntu/orb_slam3/build_ORBSLAM.sh`	顶层构建脚本（用于 ubuntu），自动构建 Pangolin、ORB_SLAM3 与 ROS 工作区并复制运行所需的库。
2	`./ubuntu/orb_slam3/WinWSL.bat`	在 Windows 上快速进入本实验的 WSL 发行版（例如 `RflySim-20.04`），用于启动 WSL 环境。
3	`./ubuntu/orb_slam3/ORB_SLAM3/`	ORB-SLAM3 官方源码目录，包含 Examples、Thirdparty、CMake 配置与构建脚本。
4	`./ubuntu/orb_slam3/Pangolin-master/`	Pangolin 可视化库源码及构建文件，用于 ORB-SLAM3 的可视化与 viewer。
5	`./ubuntu/orb_slam3/config/`	运行所需的 YAML 模板目录（mono/stereo/rgbd/imu 等）。
6	`./ubuntu/orb_slam3/config/rflysim_mono.yaml`	单目模式配置（当前为空，需填充相机内参、ORB 与 viewer 参数）。
7	`./ubuntu/orb_slam3/config/rflysim_mono_imu.yaml`	单目+IMU 模式配置（当前为空，包含 IMU 噪声与坐标变换参数）。
8	`./ubuntu/orb_slam3/config/rflysim_rgbd.yaml`	RGB-D 模式配置（当前为空，包含深度尺度与 ORB 参数）。
9	`./ubuntu/orb_slam3/config/rflysim_stereo.yaml`	双目模式配置（当前为空，包含左右相机矩阵与 baseline/bf）。
10	`./ubuntu/orb_slam3/config/rflysim_stereo_imu.yaml`	双目+IMU 模式配置（当前为空）。
11	`./ubuntu/orb_slam3/ROS1/`	ROS1 工作空间（包含 `src/`，用于构建 orb_slam3 ROS 包与示例）。
12	`./ubuntu/orb_slam3/ROS1/src/sensor_pkg/`	仿真数据桥接包（VisionCapture 示例，负责将仿真图像/IMU 推送到 ROS 话题）。
13	`./ubuntu/orb_slam3/ROS1/src/sensor_pkg/example.py`	使用 `VisionCaptureApi` 启动图像與 IMU 数据转发到 ROS 的演示脚本（示例）。
14	`./ubuntu/orb_slam3/ROS1/src/sensor_pkg/Config.json`	仿真传感器配置（VisionSensors 数组：端口、分辨率、频率等）。
15	`./ubuntu/orb_slam3/ROS1/src/sensor_pkg/readbag.py`	简易示例：将 ROS 图像话题保存到磁盘的脚本。
16	`./ubuntu/orb_slam3/ROS1/src/sensor_pkg/rflysim.rviz`	RViz 配置文件（包含预设 Displays 与视图布局）。
17	`./ubuntu/orb_slam3/ROS1/src/sensor_pkg/tf_cfg.yaml`	TF / frame_id 配置示例（sensors_frame_id、

序号	路径	说明
		imu_frame_id、imu_topic_name 等)。
18	`./ubuntu/orb_slam3/ROS1/src/orb_slam3/`	ORB-SLAM3 的 ROS 包目录（包含 `src/`、`config/`、`launch/`、`vol/` 等子目录)。
19	`./ubuntu/orb_slam3/ROS1/src/orb_slam3/src/*.cpp`	ORB-SLAM3 的 ROS 节点实现（mono, mono_imu, rgbd, stereo, stereo_imu)。
20	`./ubuntu/orb_slam3/ROS1/src/orb_slam3/config/*.yaml`	ROS 包使用的 settings 文件（相机内参、IMU 噪声、ORB 参数等)。
21	`./ubuntu/orb_slam3/ROS1/src/orb_slam3/launch/*.launch`	启动文件，传入 `vocabulary_path` 与 `config_path`，便于 roslaunch 启动不同模式。
22	`./ubuntu/orb_slam3/ROS1/src/orb_slam3/vol/ORBvoc.txt`	ORB 词袋文件（必须存在，供回环检测与 place recognition 使用)。
23	`./ubuntu/orb_slam3/ROS1/src/orb_slam3/CMakeLists.txt`	ROS 包的 CMakeLists（定义可执行目标、依赖与链接)。
24	`./windows/ORB_SLMA3.bat`	Windows 软件在环仿真启动脚本（用于启动 CopterSim/UE4/QGroundControl、PX4 SITL 等仿真流程)。
25	`./windows/Python38Run.bat`	快速打开已配置 Python环境的命令窗口并设置 PATH。
26	`./windows/server_ue4.py`	与 UE4/CopterSim 与 MAV 控制交互的 Python 示例脚本（多机控制、路径规划示例)。

4.运行环境

4.1 软件要求

Windows 10及以上版本；RflySim工具链（安装时需选择WSL2)；

> ①：WSL2安装方法请见：`\8.RflySimVision\1.BasicExps\5.LLMUavComp`

4.2 硬件要求

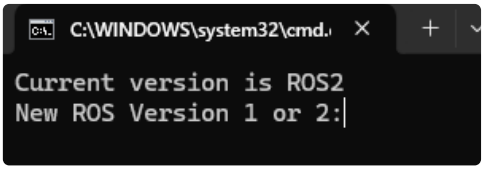
笔记本/台式电脑① 1台；

> ①：推荐配置请见：https://rflysim.com/

5.实验步骤

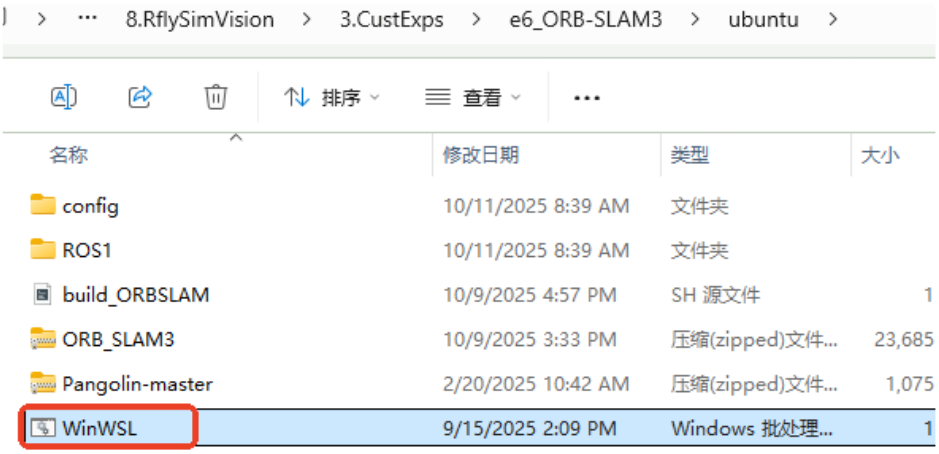
步骤 0 — 配置ROS环境

选择WSL2安装完平台后，在`\Desktop\RflyTools`中找到双击运行“RosSwitch.bat”，切换为ROS1版本

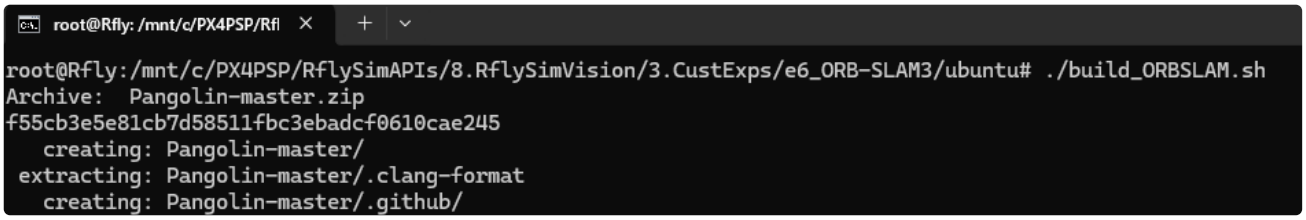


步骤 1 — 在 WSL 上构建 ORB-SLAM3 与依赖

1. 进入本例程的`ubuntu/orb_slam3`目录，双击WinWSL.bat打开一个终端，输入如下命令运行build_ORBSLAM.sh脚本




```
chmod +x *.sh
./build_ORBSLAM.sh
```



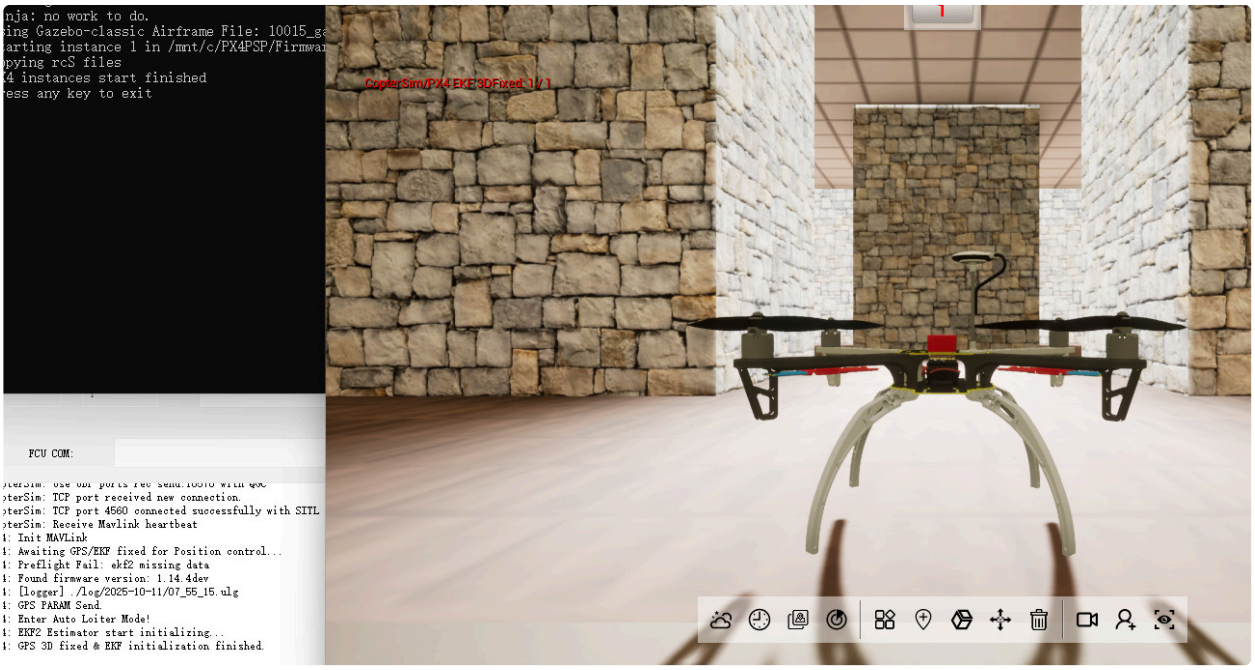
说明：脚本会解压 Pangolin、构建 Pangolin、解压 ORB_SLAM3 并执行其`build.sh`，最后进入`ROS1`目录运行`catkin_make`并拷贝库文件。

中间无报错，编译成功如下：

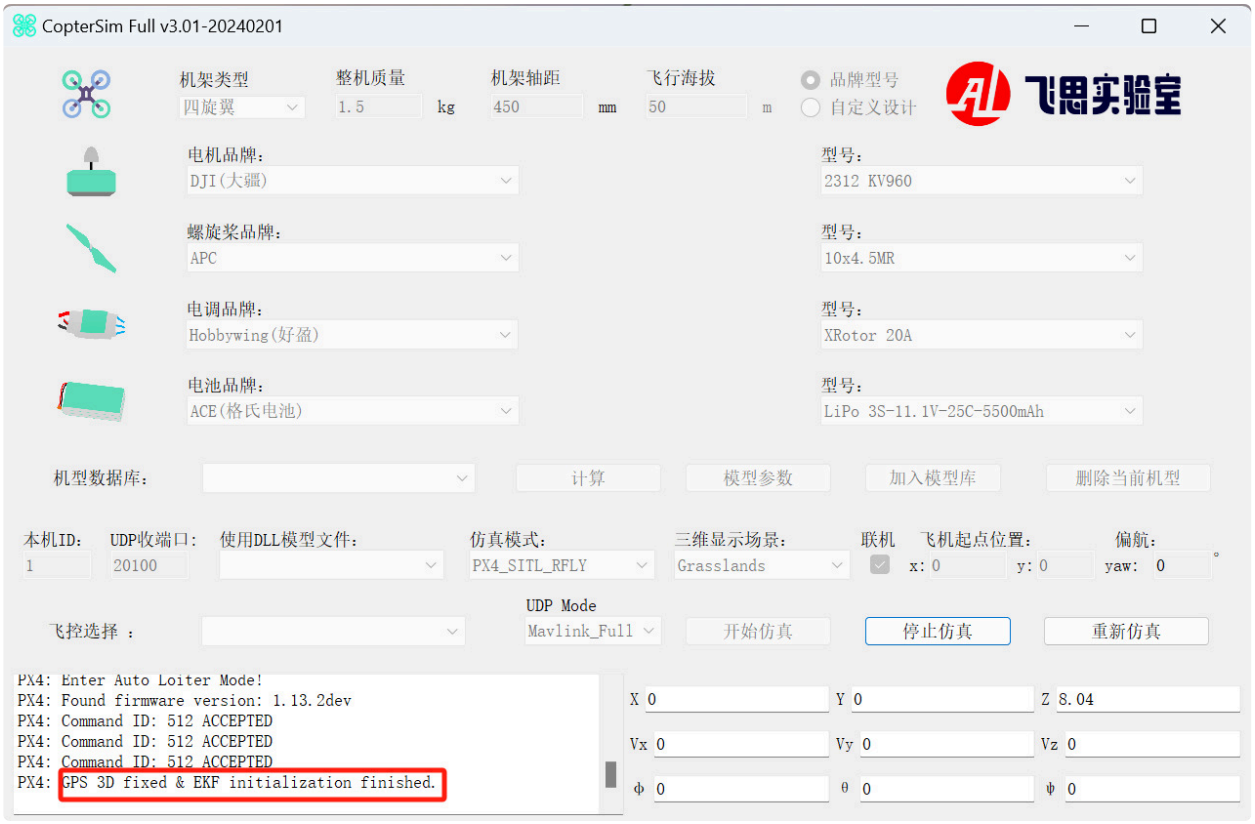
```
/mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3/ubuntu/ROS1/src/orb_slam3/../
ameraModels/KannalaBrandt8.h:102:21: warning: ‘ORB_SLAM3::KannalaBrandt8::precision’ will be in
r]
  102 |         const float precision;
      |                     ^~~~~~~~~
/mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3/ubuntu/ROS1/src/orb_slam3/../
ameraModels/KannalaBrandt8.h:88:26: warning:   ‘std::vector<int> ORB_SLAM3::KannalaBrandt8::mvL
   88 |         std::vector<int> mvLappingArea;
      |                          ^~~~~~~~~~~~~~
/mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3/ubuntu/ROS1/src/orb_slam3/../
ameraModels/KannalaBrandt8.h:59:9: warning:   when initialized here [-Wreorder]
   59 |         KannalaBrandt8(KannalaBrandt8* pKannala) : GeometricCamera(pKannala->mvParamete
precision), mvLappingArea(2,0) ,tvr(nullptr) {
      |         ^~~~~~~~~~~~~~~
[ 80%] Linking CXX executable /mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3
b_slam3/stereo_imu_node
[ 80%] Linking CXX executable /mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3
b_slam3/mono_imu_node
[ 80%] Linking CXX executable /mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3
b_slam3/mono_node
[ 90%] Linking CXX executable /mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3
b_slam3/rgbd_node
[100%] Linking CXX executable /mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3
b_slam3/stereo_node
[100%] Built target mono_node
[100%] Built target stereo_imu_node
[100%] Built target mono_imu_node
[100%] Built target rgbd_node
[100%] Built target stereo_node
root@Rfly:/mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3/ubuntu#
```

步骤 2 — 在 Windows 上启动软件在环仿真及控制程序

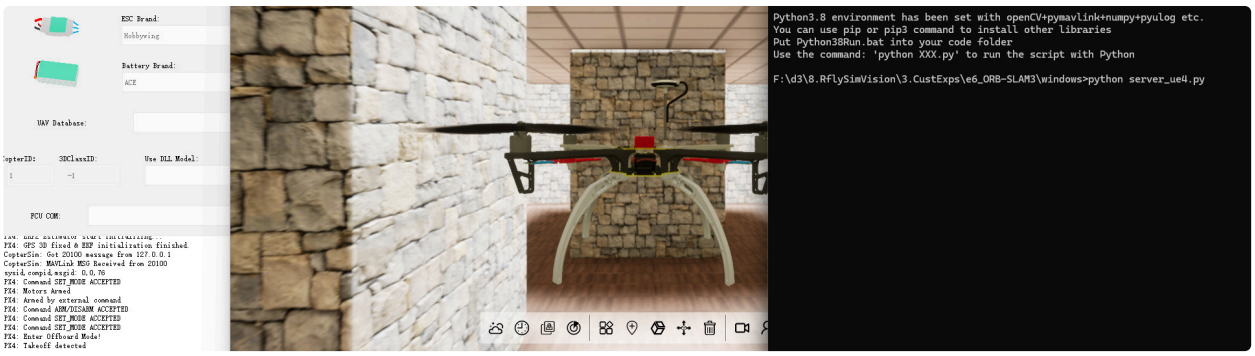
1. 进入本例程的Windows目录， 双击ORB_SLMA3.bat启动软件在环



2. 检查 CopterSim / RflySim3D / QGroundControl 是否已启动，确保所有仿真组件正常运行，无人机初始化完成（CopterSim软件下侧日志栏显示`GPS 3D fixed & EKF initialization finished`）



3. 双击Python38Run.bat，在终端输入`python server_ue4.py`运行外部控制示例程序，此时无人机开始沿预设轨迹飞行



步骤 3 — 传感器数据转发

在ubuntu/orb_slam3目录双击`WinWSL.bat`进入WSL环境，然后输入`roscore`命令

```
roscore http://198.18.0.1:11311/
root@Rfly:/mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3/ubuntu# roscore
... logging to /root/.ros/log/4dba7b16-a678-11f0-a41f-ff91fd3cf12e/roslaunch-Rfly-2673.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://198.18.0.1:45174/
ros_comm version 1.16.0


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.16.0

NODES

auto-starting new master
process[master]: started with pid [2705]
ROS_MASTER_URI=http://198.18.0.1:11311/

setting /run_id to 4dba7b16-a678-11f0-a41f-ff91fd3cf12e
process[rosout-1]: started with pid [2737]
started core service [/rosout]
```

再次双击`WinWSL.bat`进入新的WSL终端运行`example.py`，示例：

```
python3 ROS1/src/sensor_pkg/example.py
```

该脚本会：加载`Config.json`、向仿真请求数据、启动图像接收循环并将数据以 ROS 话题转发（如果`VisionCaptureApi.isEnableRosTrans=True`）。

```
root@Rfly:/mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3/ubuntu# python3 ROS1/src/sensor_pkg/example
py
current ros environment noetic
current ros environment noetic
True True
使用默认的全局坐标系下的frame_id:map
Json use absolute path mode
jsonPath= /mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3/ubuntu/ROS1/src/sensor_pkg/Config.json
Got 2 vision sensors from json
Start lisening to timeStmp Msg
Got time msg from CopterSim # 1 , running on this PC
Got CopterSim time Data for img
Got start time for SeqID # 0
Got start time for SeqID # 1
Start Image Reciver
Start lisening to IMU Msg
Got CopterSim IMU Msg!
```

步骤 4 — 在 WSL 上启动 ORB-SLAM3 ROS 节点

1. 再次双击`WinWSL.bat`进入新的WSL终端，source ROS 工作区：

```
source ROS1/devel/setup.bash
```

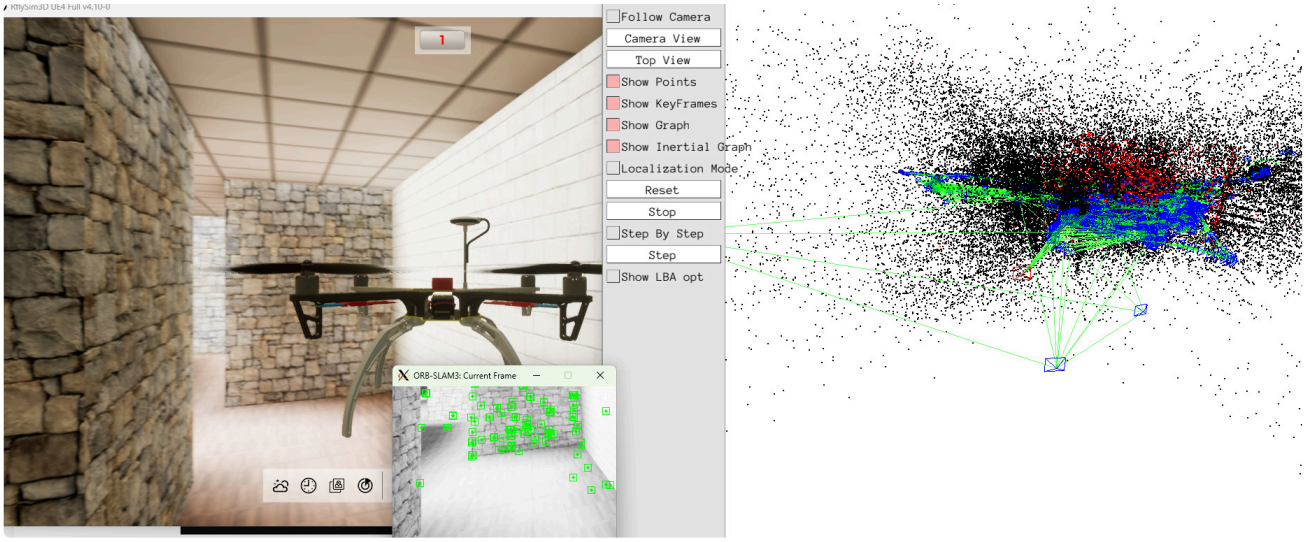
```
root@Rfly:/mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3/ubuntu# source ROS1/devel/setup.bash
root@Rfly:/mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3/ubuntu#
```

2. 选择运行模式并使用 roslaunch（示例：单目、单目+IMU、RGB-D、双目、双目+IMU）：

```
# 单目
roslaunch orb_slam3 mono.launch

# 单目+IMU
roslaunch orb_slam3 mono_imu.launch

# RGB-D
roslaunch orb_slam3 rgbd.launch

# 双目
roslaunch orb_slam3 stereo.launch

# 双目+IMU
roslaunch orb_slam3 stereo_imu.launch
```

以单目为例，效果如下

```
root@Rfly:/mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3/ubuntu/orb_slam3# source ROS1/devel/setup.ba
sh
root@Rfly:/mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3/ubuntu/orb_slam3# roslaunch orb_slam3 mono.l
aunch
... logging to /root/.ros/log/47a88a50-a67e-11f0-a1a2-43a058bd8c4b/roslaunch-Rfly-3381.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://198.18.0.1:44174/

SUMMARY
========

PARAMETERS
 * /mono/config_path: /mnt/c/PX4PSP/Rfl...
 * /mono/img_topic: /rflysim/sensor0/...
 * /mono/is_viewer: True
 * /mono/vocabulary_path: /mnt/c/PX4PSP/Rfl...
 * /rosdistro: noetic
 * /rosversion: 1.16.0

NODES
  /
    mono (orb_slam3/mono_node)

ROS_MASTER_URI=http://198.18.0.1:11311

process[mono-1]: started with pid [3417]
[WARN] [1760172303.573169953]: Usage: roslaunch rgb_slam3 mono.launch

ORB-SLAM3 Copyright (C) 2017-2020 Carlos Campos, Richard Elvira, Juan J. Gómez, José M.M. Montiel and Juan D. Tardós, Un
iversity of Zaragoza.
ORB-SLAM2 Copyright (C) 2014-2016 Raúl Mur-Artal, José M.M. Montiel and Juan D. Tardós, University of Zaragoza.
This program comes with ABSOLUTELY NO WARRANTY;
This is free software, and you are welcome to redistribute it
under certain conditions. See LICENSE.txt.

Input sensor was set to: Monocular
Loading settings from /mnt/c/PX4PSP/RflySimAPIs/8.RflySimVision/3.CustExps/e6_ORB-SLAM3/ubuntu/orb_slam3/ROS1/src/orb_sl
am3/config/rflysim_mono.yaml
Camera1.k3 optional parameter does not exist...
        -Loaded camera 1
```



6. 常见问题与解决

Q1: 双目/RGB-D 模式在"Starting the Viewer"后卡住（必现）

症状:

- 单目模式正常运行
- 双目/RGB-D 模式显示 "Starting the Viewer" 后无反应
- Map Viewer 窗口打开但无数据更新

根本原因:
话题名称不匹配导致消息同步器永远等不到匹配的图像对。

解决方案:

1. **快速诊断** - 在 WSL 终端运行：

```
cd ubuntu/orb_slam3
chmod +x check_topics.sh
./check_topics.sh
```

2. **检查关键话题是否存在**：

```
rostopic list | grep rflysim
# 应该看到:
# /rflysim/sensor0/img_rgb
# /rflysim/sensor1/img_rgb
# /rflysim/imu
```

3. **验证话题有数据**：

```
rostopic hz /rflysim/sensor0/img_rgb
# 应该显示 ~30 Hz
```

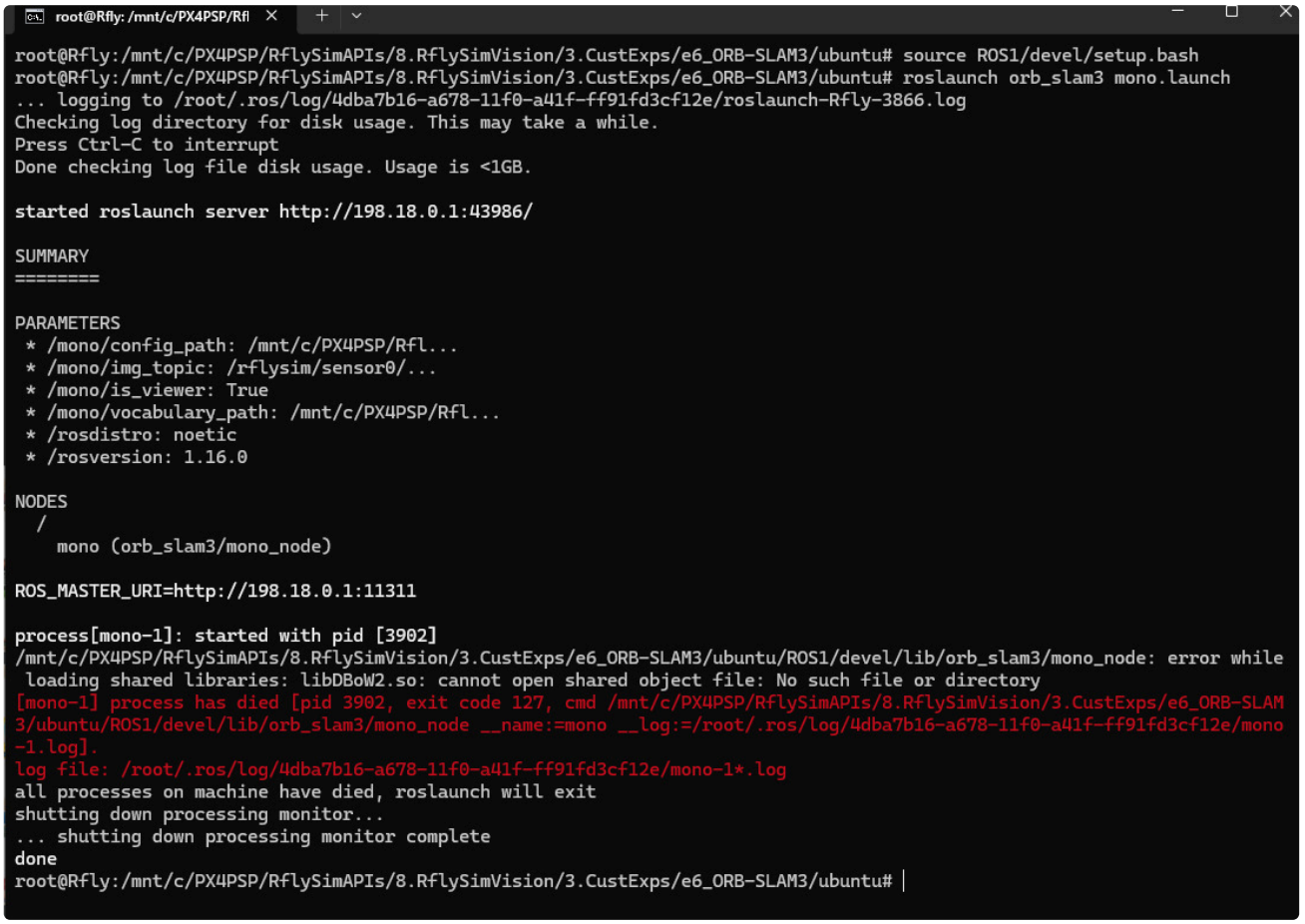
4. **已修复的配置**：
   - 所有 launch 文件已从 `img_gray` 改为 `img_rgb`
   - 需要重新编译并运行（如果修改了 Viewer.cc）

测试修复：

```
# 使用测试脚本
chmod +x test_stereo.sh
./test_stereo.sh
```

详细说明见：`FIX_STEREO_RGBD_BLOCKING.md`

Q2: 运行ros节点时提示找不到库文件



必须保持这个路径才能成功运行：\ubuntu\orb_slam3

Q3: 启动时提示找不到 `ORBvoc.txt` 或 vocabulary 加载失败

- 原因：`vocabulary_path` 配置不正确或文件缺失
- 解决：检查 `orb_slam3/vol/ORBvoc.txt` 是否存在，launch 中引用为 `$(find orb_slam3)/vol/ORBvoc.txt`。如果路径不对，请修改 launch 或把文件放到正确位置。

Q4: `example.py` 报错找不到 `Config.json`

- 原因：脚本运行时的工作目录与脚本所在目录不一致

- 解决：推荐以脚本路径运行或使用 `python ubuntu/orb_slam3/ROS1/src/sensor_pkg/example.py`，脚本已实现基于 `__file__` 的相对加载并尝试当前工作目录作为备用。

Q5: IMU 与图像时间不同步导致轨迹异常

- 原因：传感器时间戳或缓冲处理不当
- 解决：确保 IMU 的频率与配置文件匹配（如 200Hz）；检查 `sensor_pkg` 中数据发送端和 ROS topic 的时间戳是否一致。

Q6: RGB-D 模式话题不存在

- 原因：`Config.json` 只配置了 sensor0 和 sensor1
- 解决：需要在 `Config.json` 中添加 sensor2(depth) 和 sensor3(RGB)，参考 `FIX_STEREO_RGBD_BLOCKING.md` 中的配置示例

6.参考资料

1. ORB-SLAM3 官方仓库与论文
2. ROS 官方文档（ros.org）
3. RflySim 项目文档与示例