

双无人机TF树构建实验

1. 实验目的

本实验旨在学习和实践ROS（Robot Operating System）中的TF（Transform）树构建方法，通过RflySim仿真平台构建双无人机系统的坐标变换关系。实验的主要目的包括：

- 理解TF树在多机器人系统中的作用和意义
- 掌握在ROS环境下构建TF树的基本方法
- 学习如何在双无人机系统中实现坐标变换和空间关系建模
- 了解多无人机系统中传感器数据的坐标系管理
- 掌握使用RflySim平台进行无人机仿真的基本流程
- 理解命名空间（namespace）在多机器人系统中的隔离作用

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\e15_TFTreeConstruction](#)

文件目录结构说明(使用代码块格式)：

```

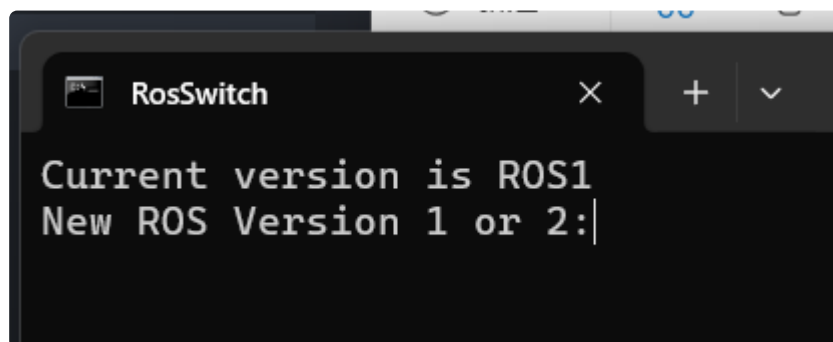
1 | └─ Build_src.sh                # 编译脚本
2 | └─ SITLRun.bat                # 启动SITL仿真环境脚本
3 | └─ WinWSL.bat                 # 启动WSL终端脚本
4 | └─ WinWSLRunBuildSrc.bat      # 编译源码并启动WSL脚本
5 | └─ WinWSLRunTempTry.bat       # 临时运行脚本
6 | └─ run_temp_try.sh            # Linux端运行脚本
7 | └─ tf_tree_ws.zip → tf_tree_ws/ # ROS工作空间目录
8 |   └─ sh/                      # 脚本目录
9 |     └─ kill_ros_pid.sh        # 杀死ROS进程脚本
10 |     └─ run_double_uav.sh      # 运行双无人机系统脚本
11 |     └─ src/                    # 源码目录
12 |       └─ faster_lio/          # Faster-LIO算法包
13 |       └─ sensor_pkg/         # 传感器功能包
14 |         └─ Config.json        # 传感器配置文件
15 |         └─ Readme.md          # 传感器包说明文档
16 |         └─ imu1.py             # 无人机1的IMU数据获取脚本
17 |         └─ imu2.py             # 无人机2的IMU数据获取脚本
18 |         └─ lidar.py            # 激光雷达数据获取脚本
19 |         └─ tf_cfg.yaml         # TF配置文件

```

4. 实验内容或步骤

4.1 首次运行编译

首先，请确认WSL子系统为ROS 1环境。可双击 `*\Desktop\RflyTools\RosSwitch.lnk` ROS切换脚本，查看显示的是否如下图所示为ROS 1，若不是请输入 `1` 来切换。



双击运行 `WinWSLRunBuildSrc.bat` 脚本将复制 `tf_tree_ws.zip` 压缩包复制到WSL系统的根目录下，并自动开始编译，编译成功后。

```

C:\WINDOWS\system32\cmd. x + v
from /opt/ros/noetic/include/ros/serialized_message.h:33,
from /opt/ros/noetic/include/ros/serialization.h:36,
from /opt/ros/noetic/include/livox_ros_driver2/CustomMsg.h:14,
from /root/tf_tree_ws/src/faster_lio/include/Laser_mapping.h:4,
from /root/tf_tree_ws/src/faster_lio/app/run_mapping_online.cc:8:
/usr/include/boost/bind.hpp:36:1: note: '#pragma message: The practice of declaring the Bind placeholders (_1, _2, ...)
in the global namespace is deprecated. Please use <boost/bind/bind.hpp> + using namespace boost::placeholders, or define
BOOST_BIND_GLOBAL_PLACEHOLDERS to retain the current behavior.'
36 | BOOST_PRAGMA_MESSAGE(
    | ^^^^^^^^^^^^^^^^^^^^^
In file included from /usr/include/boost/smart_ptr/detail/sp_thread_sleep.hpp:22,
from /usr/include/boost/smart_ptr/detail/yield_k.hpp:23,
from /usr/include/boost/smart_ptr/detail/spinlock_gcc_atomic.hpp:14,
from /usr/include/boost/smart_ptr/detail/spinlock.hpp:42,
from /usr/include/boost/smart_ptr/detail/spinlock_pool.hpp:25,
from /usr/include/boost/smart_ptr/shared_ptr.hpp:29,
from /usr/include/boost/shared_ptr.hpp:17,
from /opt/ros/noetic/include/rosbag/chunked_file.h:42,
from /opt/ros/noetic/include/rosbag/bag.h:41,
from /root/tf_tree_ws/src/faster_lio/app/run_mapping_offline.cc:6:
/usr/include/boost/bind.hpp:36:1: note: '#pragma message: The practice of declaring the Bind placeholders (_1, _2, ...)
in the global namespace is deprecated. Please use <boost/bind/bind.hpp> + using namespace boost::placeholders, or define
BOOST_BIND_GLOBAL_PLACEHOLDERS to retain the current behavior.'
36 | BOOST_PRAGMA_MESSAGE(
    | ^^^^^^^^^^^^^^^^^^^^^
[ 93%] Linking CXX executable /root/tf_tree_ws/devel/lib/faster_lio/run_mapping_online
[100%] Linking CXX executable /root/tf_tree_ws/devel/lib/faster_lio/run_mapping_offline
[100%] Built target run_mapping_online
[100%] Built target run_mapping_offline
请按任意键继续. . .

```

4.2 仿真初始化

双击 `SITLRun.bat` 脚本将会启动2个 CopterSim、1个 RflySim3D 软件，等待每个 CopterSim软件启动完成，如下图所示：



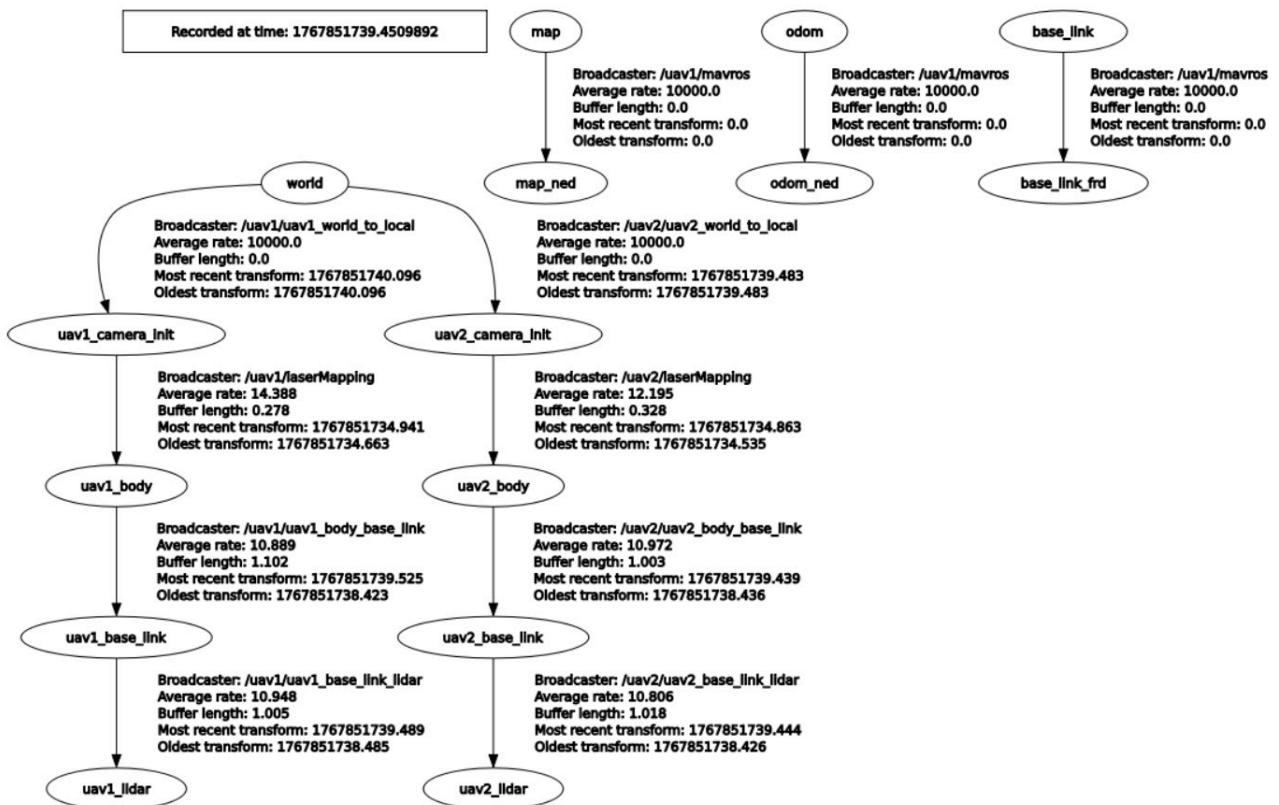


4.3 运行程序

双击运行 `WinWSLRunTempTry.bat` 脚本，将自动启动所有程序。

```
C:\WINDOWS\system32\wsl.e. x + v
trap: SIGINT: bad trap
启动 激光雷达 接口
启动 IMU 接口
mavros
[UAV 1] 启动 Faster-LIO...
[UAV 2] 启动 Faster-LIO...
启动 rviz...
```

等待几秒可以看到弹出了rqt图形窗口，其中展示了当前的tf_tree；

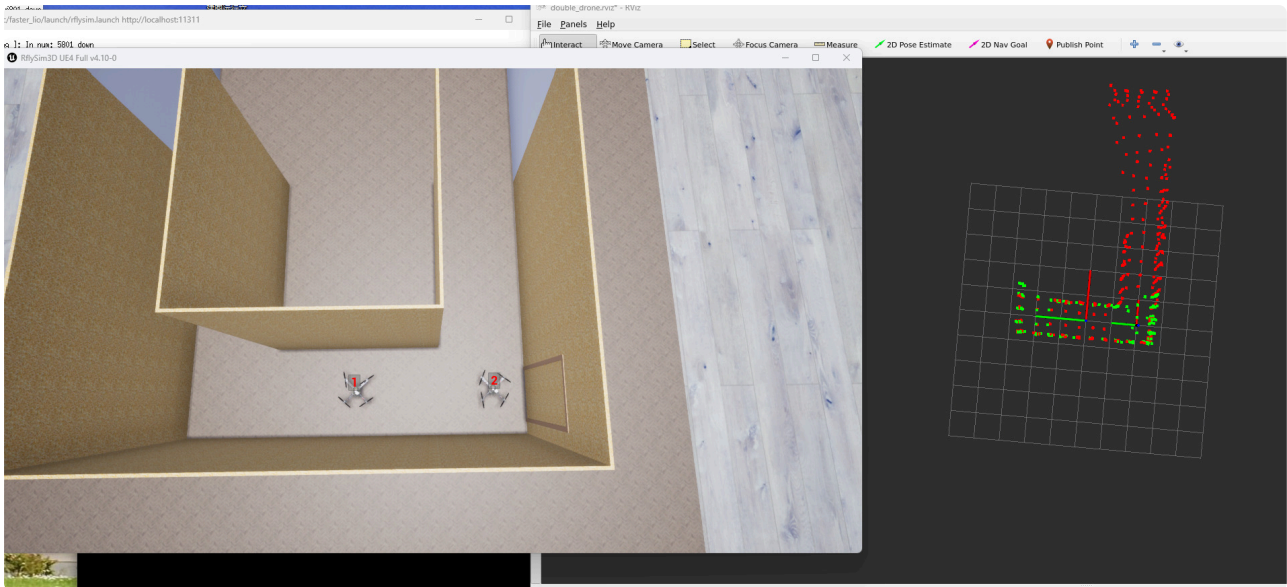


然后，再双击 `WinWSL.bat` 打开一个终端输入 `rostopic list`。可以从打印信息中看到启动了两个mavros，分别在uav1和uav2两个命名空间下。如下图所示：

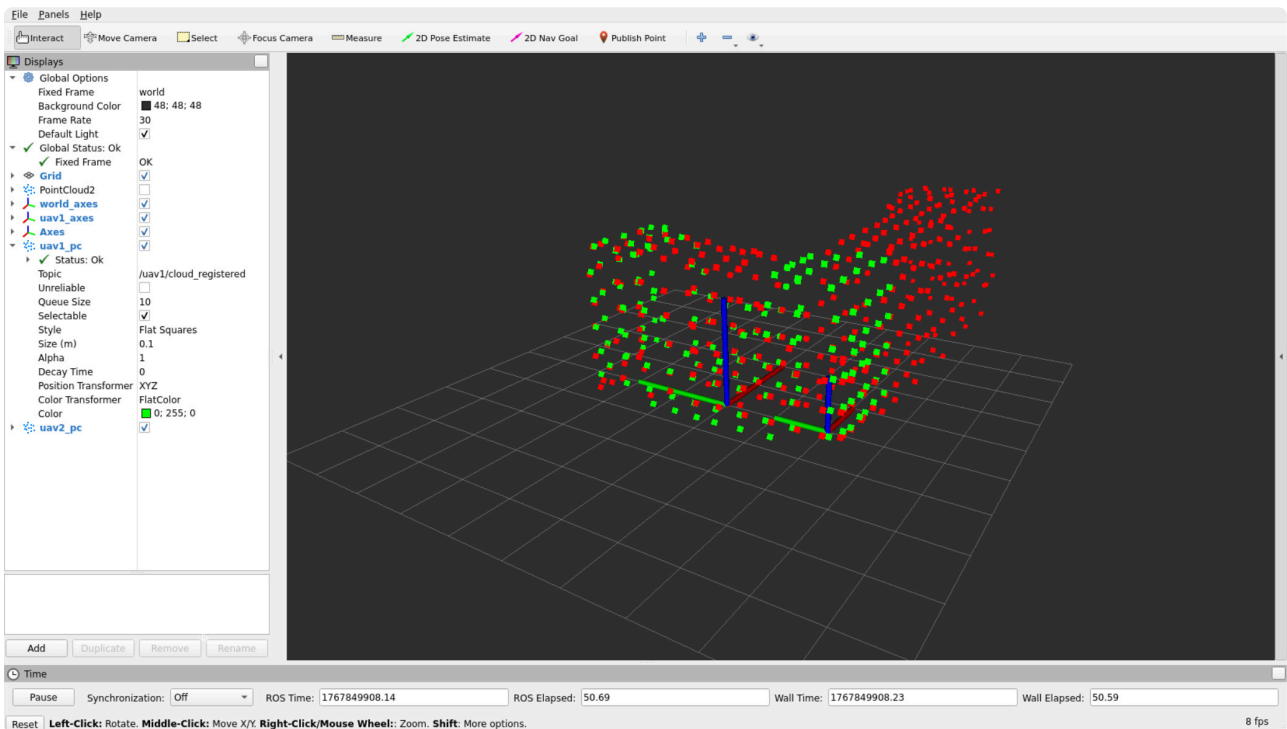
```

root@Byzeal: /mnt/d/RflySim/
/uav1/mavros/tunnel/out
/uav1/mavros/vfr_hud
/uav1/mavros/vision_pose/pose
/uav1/mavros/vision_pose/pose_cov
/uav1/mavros/vision_speed/speed_twist_cov
/uav1/mavros/wind_estimation
/uav1/path
/uav1/rflysim/imu
/uav2/cloud_registered
/uav2/cloud_registered_body
/uav2/cloud_registered_effect_world
/uav2/mavlink/from
/uav2/mavlink/gcs_ip
/uav2/mavlink/to
/uav2/mavros/actuator_control
/uav2/mavros/adsb/send
/uav2/mavros/adsb/vehicle
/uav2/mavros/altitude
/uav2/mavros/battery

```



特别说明:程序运行过程中,运行mavros的窗口汇报错误tf连接错误,但是不影响使用。



5. 关键知识点

本实验的核心是构建双无人机系统的TF树,通过ROS的TF (Transform) 系统实现坐标变换。整体框架包含两个命名空间 (uav1和uav2),每个无人机都具有独立的传感器数据发布系统。

整体思路和框架:

1. 使用RflySim仿真平台提供双无人机SITL仿真环境

2. 通过MAVROS连接仿真无人机与ROS系统
3. 使用Python脚本获取传感器数据并发布到ROS话题
4. 利用TF系统建立各坐标系之间的变换关系

关键Python程序解析：

imu1.py - 无人机1的IMU数据获取脚本：

```
1  # 启用ROS发布模式
2  VisionCaptureApi.isEnableRosTrans = True
3
4  req = ReqCopterSim.ReqCopterSim() # 获取局域网内所有CopterSim程序的电脑IP列表
5  StartCopterID = 1
6  TargetIP = req.getSimIpID(
7      StartCopterID
8  ) # 获取CopterSim的1号程序所在电脑的IP, 作为目标IP
9  # 注意：如果是本电脑运行的话, 那TargetIP是127.0.0.1的本机地址；如果是远程访问, 则是192打头的局
10 域网地址。
11  # 因此本程序能同时在本机运行, 也能在其他电脑运行。
12  # TargetIP = "172.27.144.1"
13  print(TargetIP)
14  print("Request CopterSim Send data.")
15  req.sendReSimIP(StartCopterID) # 请求回传数据到本电脑
16
17  vis = VisionCaptureApi.VisionCaptureApi(TargetIP)
18
19  vis.sendImuReqCopterSim(
20      StartCopterID, TargetIP
21  ) # 发送请求, 从目标飞机CopterSim读取IMU数据, 回传地址为127.0.0.1, 默认频率为200Hz
```

此脚本通过VisionCaptureApi接口从CopterSim仿真器获取1号无人机的IMU数据, 并启用ROS发布模式, 将数据发布到ROS系统中。

imu2.py - 无人机2的IMU数据获取脚本：

```

1 # 启用ROS发布模式
2 VisionCaptureApi.isEnableRosTrans = True
3
4 req = ReqCopterSim.ReqCopterSim() # 获取局域网内所有CopterSim程序的电脑IP列表
5 StartCopterID = 2
6 TargetIP = req.getSimIpID(
7     StartCopterID
8 ) # 获取CopterSim的2号程序所在电脑的IP, 作为目标IP
9 # 注意:如果是本电脑运行的话, 那TargetIP是127.0.0.1的本机地址;如果是远程访问, 则是192打头的局
10 域网地址。
11 # 因此本程序能同时在本机运行, 也能在其他电脑运行。
12 # TargetIP = "172.27.144.1"
13 print(TargetIP)
14 print("Request CopterSim Send data.")
15 req.sendReSimIP(StartCopterID) # 请求回传数据到本电脑
16 vis = VisionCaptureApi.VisionCaptureApi(TargetIP)
17 vis.sendImuReqCopterSim(
18     StartCopterID, TargetIP
19 ) # 发送请求, 从目标飞机CopterSim读取IMU数据, 回传地址为127.0.0.1, 默认频率为200Hz

```

与imu1.py类似, 但针对2号无人机获取IMU数据, 实现了双无人机系统的数据分离。

lidar.py - 激光雷达数据获取脚本:

```

1 # 启用ROS发布模式
2 VisionCaptureApi.isEnableRosTrans = True
3
4 req = ReqCopterSim.ReqCopterSim() # 获取局域网内所有CopterSim程序的电脑IP列表
5 StartCopterID = 1
6 TargetIP = req.getSimIpID(
7     StartCopterID
8 ) # 获取CopterSim的1号程序所在电脑的IP, 作为目标IP
9 # 注意:如果是本电脑运行的话, 那TargetIP是127.0.0.1的本机地址;如果是远程访问, 则是192打头的局
10 域网地址。
11 # 因此本程序能同时在本机运行, 也能在其他电脑运行。
12 # TargetIP = "172.27.144.1"
13 print(TargetIP)
14 print("Request CopterSim Send data.")
15 req.sendReSimIP(StartCopterID) # 请求回传数据到本电脑
16
17 vis = VisionCaptureApi.VisionCaptureApi(TargetIP)
18
19 # VisionCaptureApi 中的配置函数
20 # vis.jsonLoad() # 加载Config.json中的传感器配置文件
21 vis.jsonLoad(1) # 加载Config.json中的传感器配置文件
22
23 isSuss = vis.sendReqToUE4(0, TargetIP)
vis.startImgCap() # 开启取图循环, 执行本语句之后, 已经可以通过vis.Img[i]读取到图片了

```

此脚本负责获取激光雷达数据，并将其发布到ROS系统中，为后续的SLAM或感知任务提供数据支持。

这些Python脚本共同构成了双无人机的传感器数据获取和发布系统，配合tf_cfg.yaml配置文件，形成了完整的TF树结构。

6. 参考资料

1. RflySim官方文档
2. ROS官方TF教程：<http://wiki.ros.org/tf/Tutorials>
3. MAVROS官方文档：<https://mavros.readthedocs.io/>
4. 无人机协同控制技术文档

7. 常见问题

Q1：运行过程中mavros窗口报告tf连接错误，影响使用吗？

A1：根据实验说明，mavros窗口汇报的tf连接错误并不影响整体功能，这是由于系统初始化过程中各节点启动时间差异导致的。在系统完全启动后，tf树会正常建立，可以正常进行实验。

Q2：双击运行脚本后，没有弹出rqt图形窗口怎么办？

A2：首先检查WSL环境是否正确安装ROS，确认tf_tree_ws工作空间已成功编译。其次检查是否正确启动了SITL仿真环境（[SITLRun.bat](#)）。如果仍然无法解决，可以在终端中手动运行 `roslaunch sensor_pkg tf_tree_view.launch` 来启动rqt界面。

Q3：如何确认双无人机的TF树已正确建立？

A3：可以通过以下方式验证：

1. 运行 `roslaunch sensor_pkg tf_tree_view.launch` 命令生成frames.pdf文件，查看TF树结构
2. 在rqt界面中查看TF树的可视化表示

3. 使用 `rostopic list` 确认/uav1/mavros和/uav2/mavros话题已正确建立

4. 使用 `tf_echo` 命令监听特定坐标系变换是否正常发布

1. <https://rflysim.com/> ↩

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