

# EE478 HW3

nijatabbasov06

Apr 2025

## Table of Contents

<b>1</b>	<b>Problem 1</b>	<b>2</b>
1.1	Part 1 . . . . .	2
1.2	Part 2 . . . . .	2
1.3	Part 3 . . . . .	3
1.4	Part 4 . . . . .	3
<b>2</b>	<b>Problem 2</b>	<b>4</b>
2.1	Part 1 . . . . .	4
2.2	Part 2 . . . . .	6
<b>3</b>	<b>Problem 3</b>	<b>8</b>
3.1	Part 1 . . . . .	8
3.2	Part 2 . . . . .	9
<b>4</b>	<b>Reference</b>	<b>9</b>

# 1 Problem 1

## 1.1 Part 1

Checking the `realsense_stereo_imu_config.yaml` file in the KAIST VIO dataset github, we can see that there is written `image0_topic: "/camera/intra1/image_rect_raw"` `image1_topic: "/camera/intra2/image_rect_raw"`. Also, in the `orb_slam3_ros/config/Stereo/` folder in `orb_slam3_ros` package, we can see different yaml files for different camera models. For our assignment, we will use `RealSense_D435i.yaml` file.

```
<!-- Launch -->
<param name="use_sim_time" value="false" />

<!-- Main node -->
<node name="orb_slam3" pkg="orb_slam3_ros" type="ros_stereo" output="screen">

  <!-- Change the topics according to the dataset -->
  <remap from="/camera/left/image_raw" to="/camera/intra1/image_rect_raw"/>
  <remap from="/camera/right/image_raw" to="/camera/intra2/image_rect_raw"/>

  <!-- Parameters for original ORB-SLAM3 -->
  <param name="voc_file" type="string" value="$(find orb_slam3_ros)/orb_slam3/Vocabulary/ORBvoc.txt.xml"/>
  <param name="settings_file" type="string" value="$(find orb_slam3_ros)/config/Stereo/RealSense_D435i.yaml"/>

  <!-- Parameters for ROS -->
  <param name="world_frame_id" type="string" value="world" />
  <param name="cam_frame_id" type="string" value="camera" />
  <param name="enable_pangolin" type="bool" value="true" />
</node>

<!-- Visualization -->
<node name="rviz" pkg="rviz" type="rviz" args="-d $(find orb_slam3_ros)/config/ORB_SLAM3/ORB_SLAM3.rviz" output="screen" />
```

Figure 1: Changed topics and camera model

## 1.2 Part 2

Looking at the config files at KAIST VIO dataset github repo, following lines were added to the `orb_slam3_ros/config/Stereo/RealSense_D435i.yaml` file:

```
1 %YAML:1.0
2
3 #-----
4 # Camera Parameters. Adjust them!
5 #-----
6 File:version: "1.0"
7
8
9
10 Camera1.fx: 380.9229090195708
11 Camera1.fy: 380.29264802262736
12 Camera1.cx: 324.68121181846755
13 Camera1.cy: 224.6741321466431
14 Camera1.k1: 0.006896928127777268
15 Camera1.k2: -0.009144207062654397
16 Camera1.p1: 0.000254113977103925
17 Camera1.p2: 0.0021434982252719545
18 Camera1.k3: 0.0
19
20 Camera2.fx: 380.95187095303424
21 Camera2.fy: 380.3065956074995
22 Camera2.cx: 324.0678433553536
23 Camera2.cy: 225.9586983198407
24 Camera2.k1: 0.007044055287844759
25 Camera2.k2: -0.010251485722185347
26 Camera2.p1: 0.0006674304399871926
27 Camera2.p2: 0.001678899816379666
28 Camera2.k3: 0.0
29
30 Stereo.b: 0.0499585
31
32
33 Stereo.T_c1_c2: !!opencv-matrix
34   rows: 4
35   cols: 4
36   dt: f
37   data: [1.0, 0.0, 0.0, 0.0499585,
38         0.0, 1.0, 0.0, 0.0,
39         0.0, 0.0, 1.0, 0.0,
40         0.0, 0.0, 0.0, 1.0]
41
42 # Camera resolution
43 Camera.width: 640
44 Camera.height: 480
45
46 # Camera frames per second
47 Camera.fps: 30
48
49 Camera.type: "PinHole"
50
51 # Color order of the images (0: BGR, 1: RGB. It is ignored if images are grayscale)
```

Figure 2: Additions to the RealSense\_D435i.yaml file

### 1.3 Part 3

Following lines were added to the common.cc to change the rotation matrix accordingly:

```
void publish_camera_pose(Sophus::SE3f Tcw_SE3f, ros::Time msg_time)
{
    geometry_msgs::PoseStamped pose_msg;
    pose_msg.header.frame_id = world_frame_id;
    pose_msg.header.stamp = msg_time;

    Eigen::Matrix3f R_cam_to_enu;
    R_cam_to_enu << 0, 0, 1,
                    -1, 0, 0,
                    0, -1, 0;

    Eigen::Matrix3f Rwc = Tcw_SE3f.rotationMatrix();
    Eigen::Vector3f twc = Tcw_SE3f.translation();

    Eigen::Matrix3f Rwb = Rwc * R_cam_to_enu;
    Eigen::Vector3f twb = twc;

    Eigen::Quaternionf qwb(Rwb);

    pose_msg.pose.position.x = twb.x();
    pose_msg.pose.position.y = twb.y();
    pose_msg.pose.position.z = twb.z();

    pose_msg.pose.orientation.w = qwb.w();
    pose_msg.pose.orientation.x = qwb.x();
    pose_msg.pose.orientation.y = qwb.y();
    pose_msg.pose.orientation.z = qwb.z();

    pose_pub.publish(pose_msg);
}
```

Figure 3: Publishing camera pose

### 1.4 Part 4

To run the bag file, we first need to run roscore. But as the euroc\_stereo.launch file already starts the launch file, we can first run the launch file, and then run the bag file. The launch file opens rviz, map viewer, current frame windows and waits for the necessary topics to publish camera images. Then, we can run the bag file so that those camera images would be published to the topics that the orb\_slam3\_ros is listening. However, when we launch the both files we can see that there is no orb\_slam3\_ros/camera\_pose topic in the rviz. So we add the topic in rviz because it is asked in the assignment:

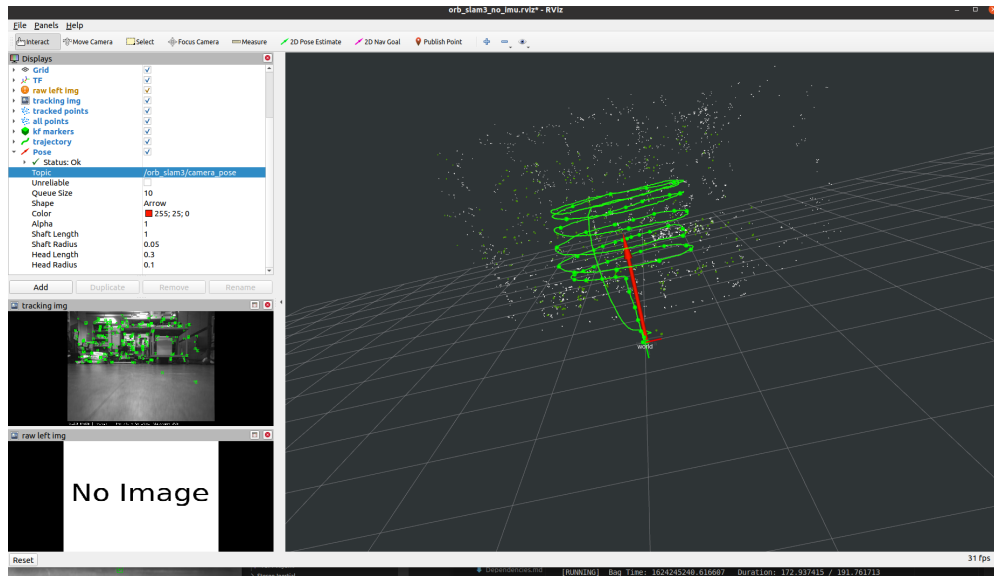


Figure 4: Circle

Here, I am using bag file corresponding to circular motion. The big red arrow is the position odometry for the drone.

## 2 Problem 2

### 2.1 Part 1

In this problem, we can realize that we need to compare `/pose_transformed` (groundtruth data) and `/orb_slam3/ros/camera_pose` topics. However, running the `rqt_bag` 'circle (1).bag' file we get the following:

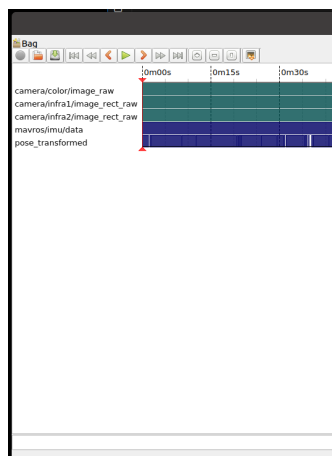


Figure 5: recorded topics for the bag file

So, we realize that we need to record the `camera_pose` topics ourselves. For this reason, I run the following command:

```

Problems 76 Output Debug Console Terminal Ports
'circle (1).bag' package Videos
acss@acss-B760M-DS3H:~/ACSS_Proj/EE478$ rqt bag 'circle (1).bag'
PluginManager: load plugin() could not load plugin "rqt bag/Bag": RosPyPluginProvider._init_node() could not find ROS master
acss@acss-B760M-DS3H:~/ACSS_Proj/EE478$ rqt bag 'circle (1).bag'
acss@acss-B760M-DS3H:~/ACSS_Proj/EE478$ rosbag record /orb_slam3/camera_pose /pose_transformed^C
acss@acss-B760M-DS3H:~/ACSS_Proj/EE478$ rosbag record /orb_slam3/camera_pose /pose_transformed

```

Figure 6: rosbag record

Then, using that file for comparing in evo\_traj, I use the command with the following arguments:

```

acss@acss-B760M-DS3H:~/ACSS_Proj/EE478$ evo_traj bag 2023-05-21-01-22-57.bag /orb_slam3/camera_pose --ref /pose_transformed -p --align
name: /orb_slam3/camera_pose
infos: 3208 poses, 34.030m path length, 149.758s duration
name: /pose_transformed
infos: 8408 poses, 29.990m path length, 182.664s duration

```

Figure 7: evo\_traj arguments

Here, -p is used for plotting, -align is for telling evo\_traj that I want the trajectories to be aligned. -ref is used for telling the evo\_traj that the groundtruth is the topic that comes right after -ref so that the evo\_traj can use that ground truth topic for aligning the trajectories. Following is the resulting plot:

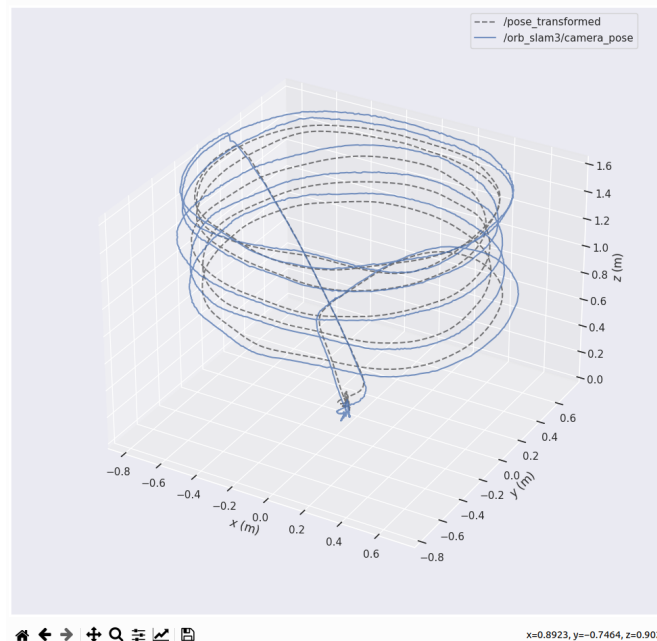


Figure 8: trajectories

## 2.2 Part 2

```
# ORB Parameters
#-----
# ORB Extractor: Number of features per image
ORBextractor.nFeatures: 1250

# ORB Extractor: Scale factor between levels in the scale pyramid
ORBextractor.scaleFactor: 1.2

# ORB Extractor: Number of levels in the scale pyramid
ORBextractor.nLevels: 8

# ORB Extractor: Fast threshold
# Image is divided in a grid. At each cell FAST are extracted imposing a minimum response.
# Firstly we impose initFAST. If no corners are detected we impose a lower value minFAST
# You can lower these values if your images have low contrast
ORBextractor.initFAST: 20
ORBextractor.minFAST: 7

#-----
# Viewer Parameters
#-----
Viewer.KeyFrameSize: 0.05
Viewer.KeyFrameLineWidth: 1.0
Viewer.GraphLineWidth: 0.9
Viewer.PointSize: 2.0
Viewer.CameraSize: 0.08
Viewer.CameraLineWidth: 3.0
Viewer.ViewpointX: 0.0
Viewer.ViewpointY: -0.7
Viewer.ViewpointZ: -3.5
Viewer.ViewpointF: 500.0
```

Figure 9: Parameters

Here, we can see that there are various parameters about the orb slam. The nFeatures parameter means how many features to extract from the image, we can see that, when we increase the feature amounts, the green squares in the image increases. While increasing feature numbers can help the slam do localization much faster and more accurate, it is also computationally expensive.

The followings are the examples of running the bag file with 2000 nFeatures parameter. We can see that, the green squares in the image is much more than the one with 1250 features:

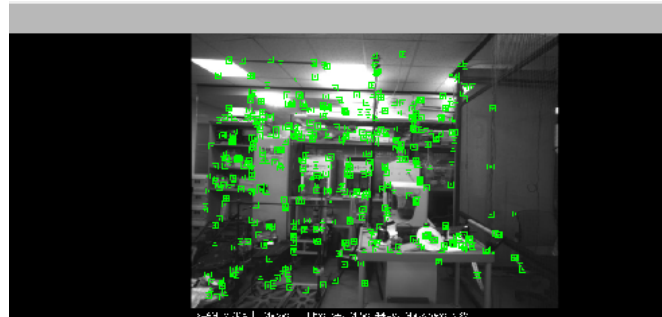


Figure 10: 2000 nFeatures

However, we cannot see much difference between the reference pose and the actual pose.



Figure 11: 2500 features

We can see that increasing the features to 2500, the features increase more.

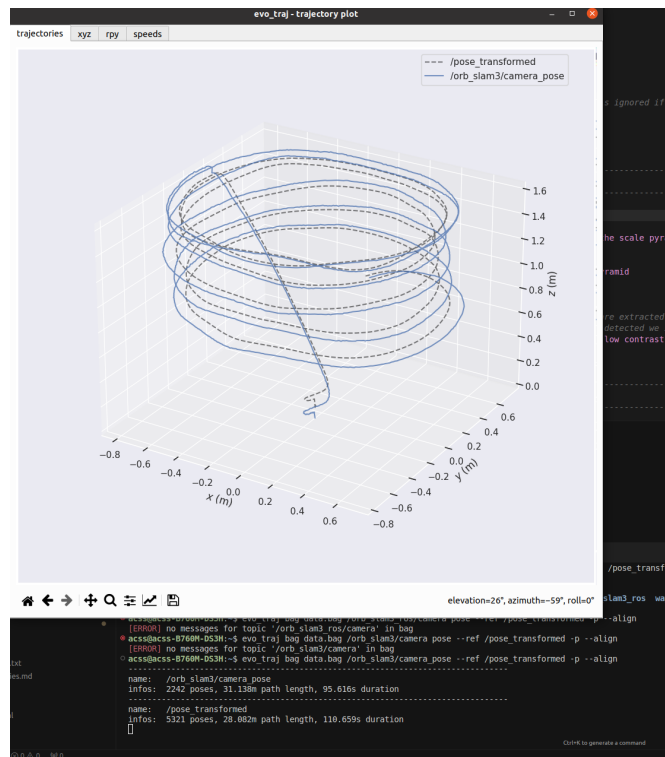


Figure 12: Trajectory

and the following is the trajectory comparison:

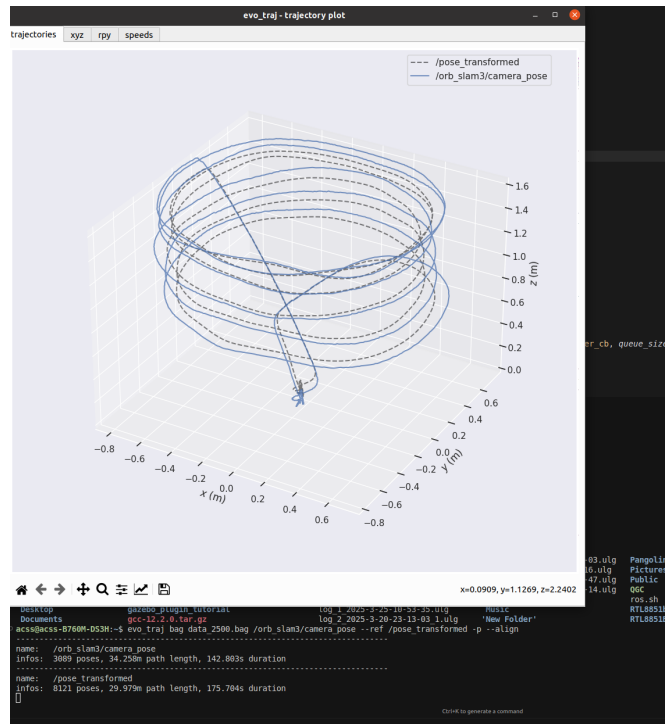


Figure 13: Trajectory comparison

### 3 Problem 3

#### 3.1 Part 1

Using the christmas\_tree.launch file in the launch directory of the px4 repository, I am using the following arguments for world and sdf arguments:

```
-arg name="vehicle" default="iris"/>
-arg name="world" default="$(find mavlink_sitl_gazebo)/worlds/christmas_tree.world"/>
-arg name="sdf" default="$(find mavlink_sitl_gazebo)/models/(arg vehicle)_stereo_camera/(arg vehicle)_stereo_camera.sdf"/>

<!-- gazebo configs -->
-arg name="gui" default="true"/>
```

Figure 14: world and sdf arguments

To find the proper image topic needed for slam, we run the launch file, and in the new terminal we write rostopic list. There we can see that there are /stereo/left/image\_raw and /stereo/right/image\_raw topics. So, we use those topics in the euroc\_stereo.launch:

```
<!-- change the topics according to the dataset -->
<remap from="/camera/left/image_raw" to="/stereo/left/image_raw"/>
<remap from="/camera/right/image_raw" to="/stereo/right/image_raw"/>

<!-- Parameters for original ORB-SLAM3 -->
<param name="voc_file" type="string" value="$(find orb_slam3_ros)/orb_slam3/Vocabulary/ORBvoc.txt"/>
<param name="settings_file" type="string" value="$(find orb_slam3_ros)/config/Stereo/NeatSense_D435i.yaml"/>
```

Figure 15: image topics

Then, we change the subscribed topic in the waypoint\_mission/waypoint\_mission.py :



```

def __init__(self):
    self.cur_waypoint_idx = -1
    self.cur_waypoint = Point()
    self.pose_real = Point()
    self.cur_yaw = 0.0

    self.waypoint_server = rospy.Service('waypoint_mission_server', Empty, self.waypoint_service)
    self.waypoint_pub = rospy.Publisher('waypoint_mission', Point, queue_size=10)
    self.cur_position_sub = rospy.Subscriber('/orb_slam/camera_pose', PoseStamped, self.auto_arrive_checker_cb, queue_size=1)
    self.local_vel_pub = rospy.Publisher('mavros/setpoint_velocity/cmd_vel', TwistStamped, queue_size=10)

def waypoint_service(self, req):
    if not self.get_next_waypoint():
        return rospy.ServiceResponse('waypoint_mission', Empty, False)
    else:
        return rospy.ServiceResponse('waypoint_mission', Empty, True)

```

Figure 16: waypoint mission.py

## 3.2 Part 2

For this part I gave the following positions and the following yaw list for the drone to follow:

```

# Waypoints [[X list], [Y list], [Z list]]
waypoint_list = [[0, 0, 0, 8, 0, 0], \
                 [0, 4, 4, -4, -4, 0], \
                 [1, 1, 1, 1, 1, 1]]

# Center of the circle (change as needed)
center_x = 4
center_y = 0

# Hardcoded yaw setpoints (in radians) for each waypoint, facing the center (4, 0)
yaw_setpoints = [0.3, 0.785, 1.5708, 2.356, 3.1416, 2.356] # Example values, adjust as

def dist(goal, pose_now):

```

Figure 17: Drone setpoints

## 4 Reference

[1] EE478 KAIST lecture notes [2]