CODING EX -1

KALAIPRIYAN R – KR53

MOBILE ROBOTICS FALL 2023

Question 1.

The teaching team collected data from Terrasentia robot over a field. The data collected includes GPS data, encoder velocity, and inertial sensor data. Play the rosbag file and make a list of the topics in your report, mentioning their publishing rate and the type of message used by each topic.

Question 2.

Create a ROS node to read the messages from the rosbag file and determine the robot's position along the trajectory. Publish an odometry ROS message which includes the position, the linear and angular velocities, and the heading (in quaternions representation) of the robot. For this part, we provide two python templates called coding_ex1_part1.py and utils.py. You must complete the following tasks: - Create a ROS package and build it with the given python templates (see Tutorial_1 in the appendix section of this document). - Complete the function quaternion_from_euler in utils.py. - Complete the constructor (__init__(self):) of the odometry_node class, creating the subscribers to each of the topics in the rosbag file (see example in the template). Write the corresponding callback functions for each subscriber. - Complete the function timer_callback_odom, which is responsible for publishing the odometry message. - Complete the function broadcast_tf, which is responsible for publishing the transformation between two coordinate frames. Finally, run your node and play the rosbag file. Visualize the trajectory made by robot using RVIZ2. Add a screenshot of this trajectory to your report. After running your node (for at least 30 s), a .txt file will be saved in your ROS workspace with the name results_part1.txt. Submit this file to Gradescope, along with the files coding_ex1_part1.py, and utils.py.

Question 3.

Create a copy of your main template and name it coding_ex1_part2.py. Replace the name of the results file in line self.file_object_results = open("results_part1.txt", "w+") by results_part2.txt. In this part, publish the same odometry message but instead of computing the position based on the wheel odometry, use the GPS measurements. You can use the function lonlat2xyz(lat, lon, lat0, lon0) (in utils.py) to compute the displacement in east-west direction (x) and north-south direction (y) with respect to a fixed point with coordinates (lat0, lon0). Choose wisely the initial coordinates, so that x and y start at zero value. Uses RVIZ2 to visualize the final trajectory and add it to your report. Submit the files coding_ex1_part2.py, utils.py and results_part2.txt through Gradescope.

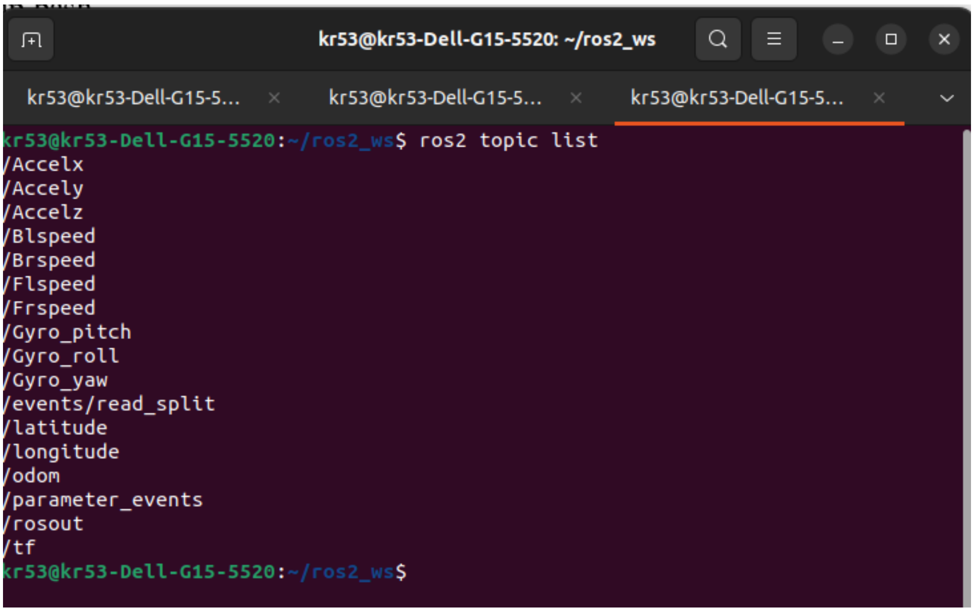
Question 4.

Compare the results from questions 2 and 3. Explain the reasons for any similarities or differences.

SOLUTIONS:

Question 1:

The topic list after playing the rosbag was:



The publishing rate and type is given through the screenshots below:

ACCELX.png
ACCELY.png
ACCELZ.png
BLSPEED.png
BRSPEED.png
events-read_split .png
FLSPEED.png
FRSPEED.png
GYROPITCH.png
GYROROLL.png
GYROYAW.png
LATITUDE.png
LONGITUDE.png
PARAMETER EVENTS.png
ROSOUT.png
TF.png

The above process was carried out with the help of terminal commands such as:

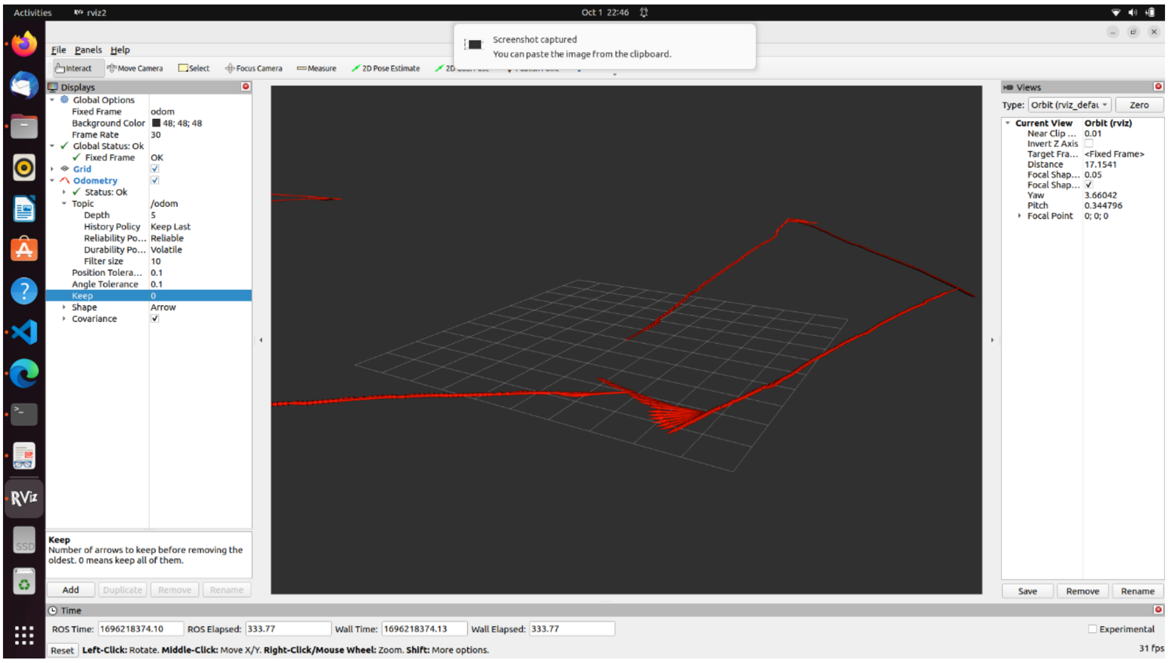
- ros2 topic info /(topic name) - (Topic Type)
- ros2 topic hz /(topic name) - (Topic's publishing rate)

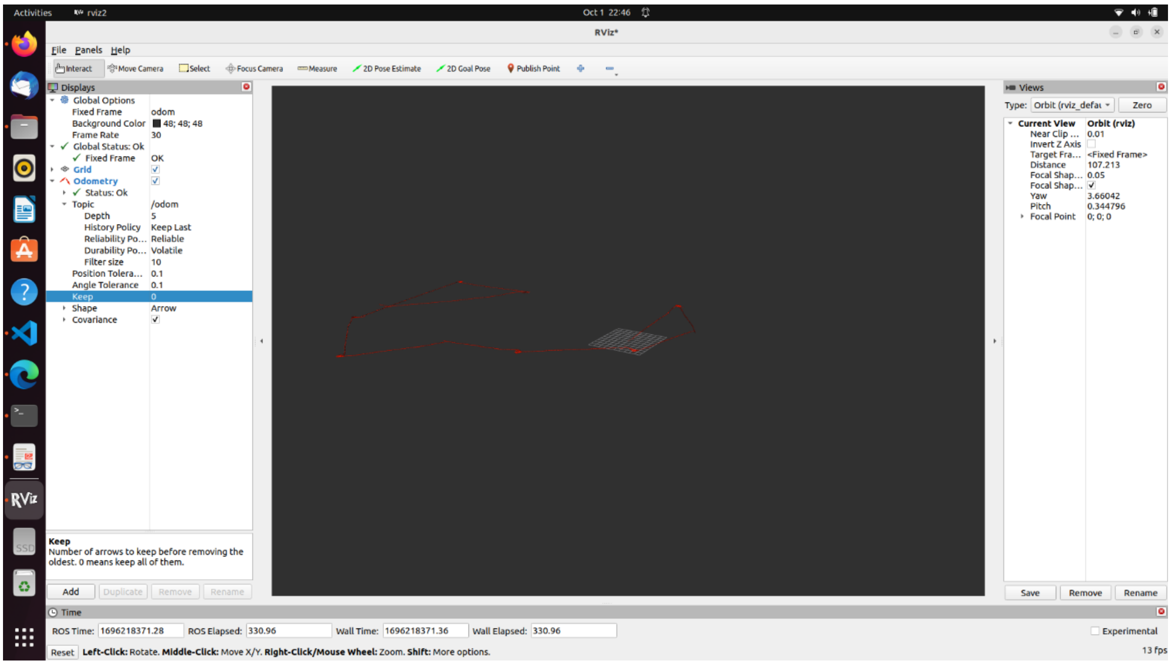
IF THE IMAGES ARE NOT LOADING:

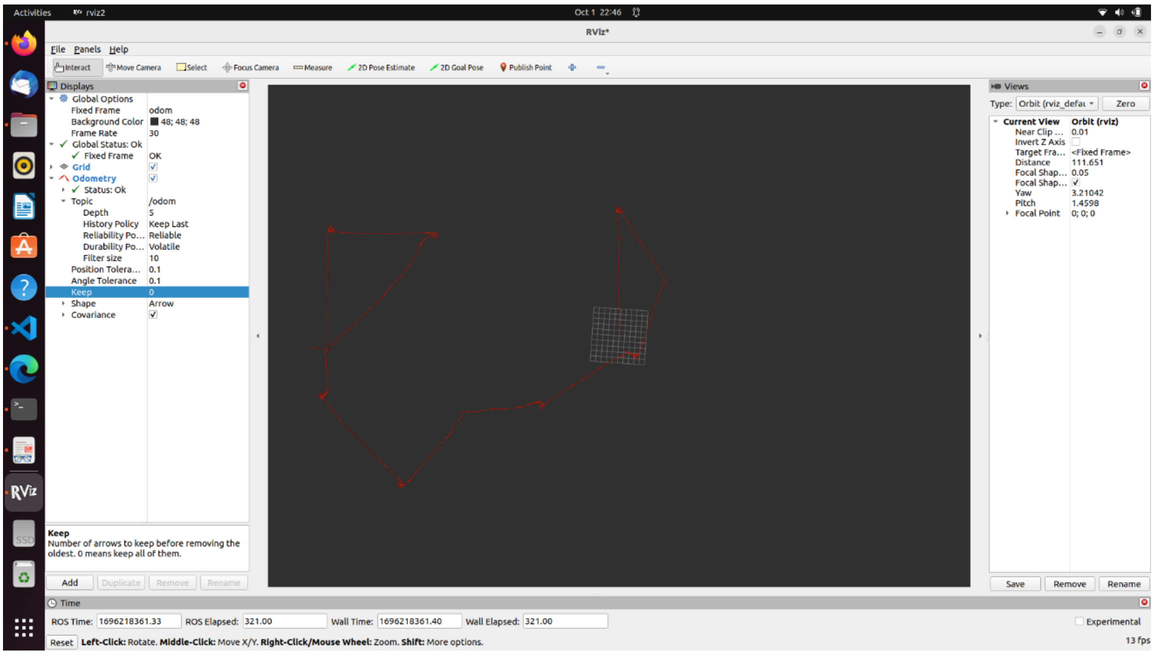
https://drive.google.com/drive/folders/18mykjkOc7bHyhk3BEA2YiF2ZOCy1IM6T?usp=drive_link

Question 2:

The part1 was coding w.r.t robot’s dimensions and velocity. It involved quaternion form euler function from utils.py. The code was run and using RVIZ2, we were able to visualize the path of the robot that was carried out by dimensions and velocity.





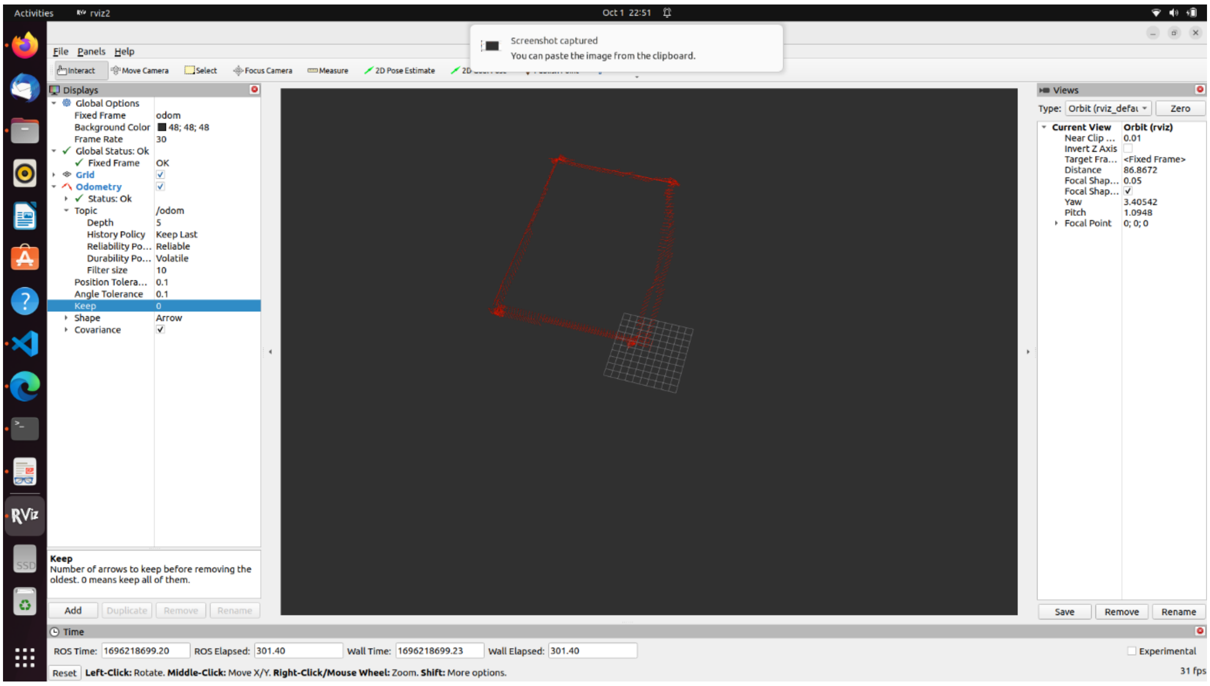


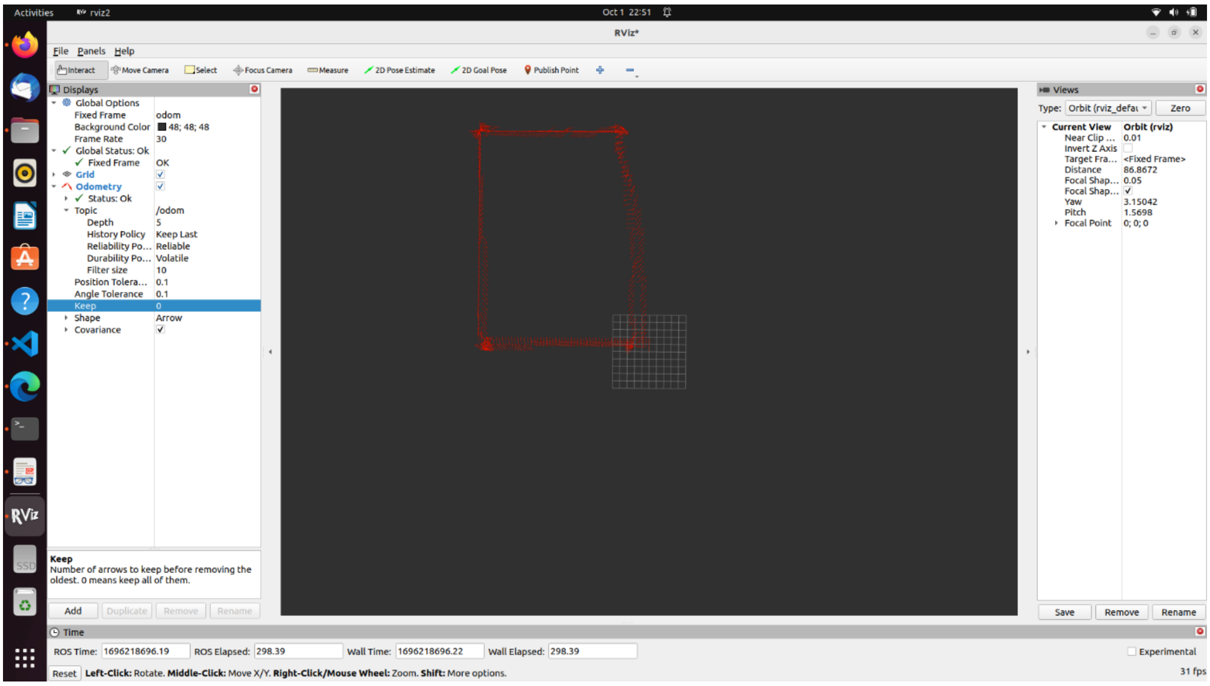
Activities
rviz2
Oct 1 22:46
RViz*
File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Goal Pose
Publish Point
Displays
Global Options
Fixed Frame odom
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid
Odometry
Status: Ok
Topic /odom
Depth 5
History Policy Keep Last
Reliability Po... Reliable
Durability Po... Volatile
Filter size 10
Position Tolera... 0.1
Angle Tolerance 0.1
Keep 0
Shape Arrow
Covariance
Keep
Number of arrows to keep before removing the oldest. 0 means keep all of them.
Add
Duplicate
Remove
Rename
Views
Type: Orbit (rviz_defau
Zero
Current View Orbit (rviz)
Near Clip ... 0.01
Invert Z Axis
Target Fra... <Fixed Frame>
Distance 111.651
Focal Shap... 0.05
Focal Shap...
Yaw 3.21042
Pitch 1.4598
Focal Point 0; 0; 0
Save
Remove
Rename
Time
ROS Time: 1696218361.33
ROS Elapsed: 321.00
Wall Time: 1696218361.40
Wall Elapsed: 321.00
Experimental
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.
13 fps

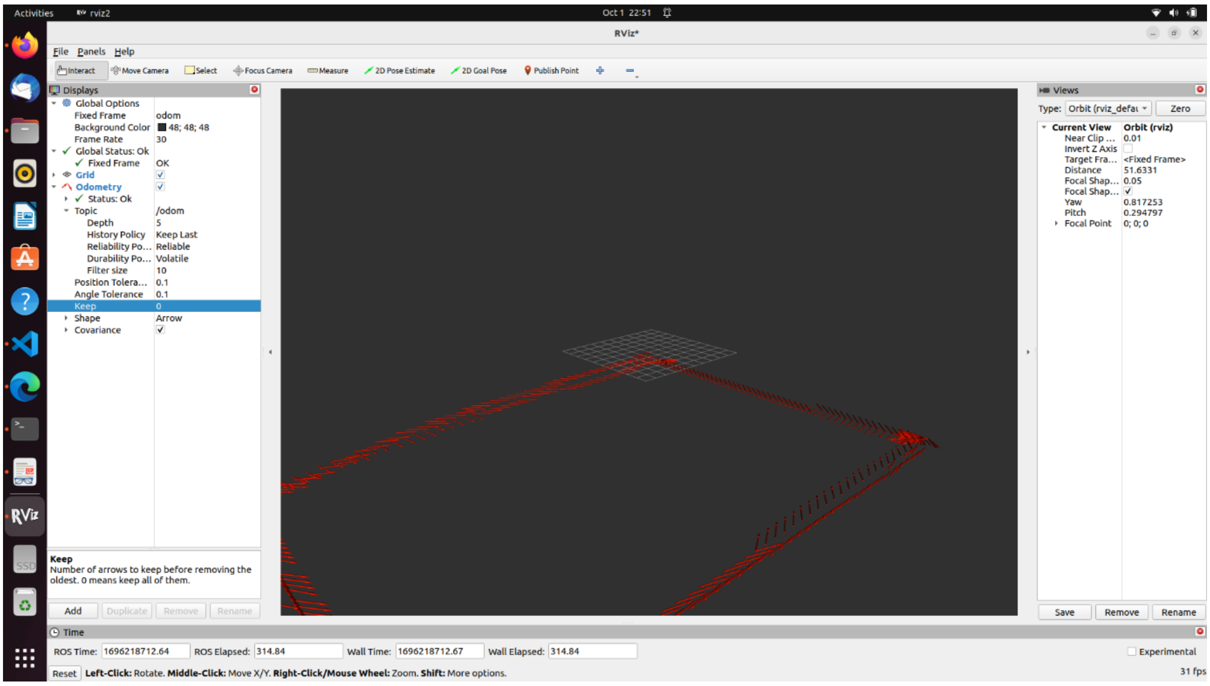
QUESTION 3:

The part2 did the exact same job as part1, but instead traced the path with the help of GPS instead of dimensions. Used lonlat2xyz(lat, lon, lat0, lon0) Function from utils.py.
After running the rosbag and nodes for the part2, we were able to visualise the path of the robot using RVIZ2.

The path looked like:






Activities
rviz2
Oct 1 22:51
RViz*
File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Goal Pose
Publish Point
Displays
Global Options
Fixed Frame odom
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid
Odometry
Status: Ok
Topic /odom
Depth 5
History Policy Keep Last
Reliability Po... Reliable
Durability Po... Volatile
Filter size 10
Position Tolera... 0.1
Angle Tolerance 0.1
Keep 0
Shape Arrow
Covariance
Keep
Number of arrows to keep before removing the oldest. 0 means keep all of them.
Add
Duplicate
Remove
Rename
Views
Type: Orbit (rviz_defau
Zero
Current View Orbit (rviz)
Near Clip ... 0.01
Invert Z Axis
Target Fra... <Fixed Frame>
Distance 51.6331
Focal Shap... 0.05
Focal Shap...
Yaw 0.817253
Pitch 0.294797
Focal Point 0; 0; 0
Save
Remove
Rename
Time
ROS Time: 1696218712.64
ROS Elapsed: 314.84
Wall Time: 1696218712.67
Wall Elapsed: 314.84
Experimental
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.
31 fps

QUESTION 4:

In comparing the results between Part 1 (Wheel Odometry) and Part 2 (GPS), it becomes evident that the GPS-based approach outperforms the Wheel Odometry method in several key aspects. i.e the GPS method and Wheel odometry methods gave different outputs.

The reasons could be:

- GPS has high accuracy and the wheel odometer is prone to drifts and local variations.
- GPS is not affected by surface conditions and wear and tear of the robot.
- GPS generates clear, structured trajectories, crucial for navigation, while Wheel Odometry often produces distorted paths.