Fall 2022

Final Project

Introductory Robot Programming

December 16, 2022

Student:
Shreejay Badshah, Joseph Thomas,
Amogh Wyawahare

Instructors:
Dr. Z. Kootbally

Course code:
ENPM809Y

Contents

1 Introduction

As a final project submission for Introduction to Robot Programming (ENPM809Y) at the University of Maryland, this report summarizes the work that was done, the methodology that was chosen, and some of the most significant problems that were encountered while working on this project. The project's goal is to move the robot in the gazebo simulator using ROS2.

The project begins by joining the separated TF trees, followed by loading parameters in YAML file. The primary goal of the project is to maneuver the robot to a spot, from where the aruco marker is to be found. After detecting the marker, the robot is to be moved to another pre-defined place based on the information read from the marker.

The project puts the student's ROS2 skills to the test for activities such as publishing and subscribing to topics, sending and receiving messages on these topics, and drawing conclusions based on the messages received on these topics. In addition, the basic understanding of ROS2 and the operation of publisher subscriber protocols is assessed.

2 Approach

The approach to accomplish this project was as per the final project guideline.
First, each one of the tasks given in the project guidelines were read and understood and broken down into small tasks over a span of 2 days.

2.1 Broadcaster

This was the first task to be completed. The TF tree upon first launching shows disconnected trees. TF maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time. Connecting the world and robot frames is the goal of this task. To do this, the package 'odom_updater' is created with the dependencies : 'rclcpp', 'geometry_msgs', 'nav_msgs', 'tf2_ros'. Then a node 'odom_updater' is created to broadcast the robot's 'base_footprint' as a child of 'robot_odom'. The next objective was to retrieve the pose in the 'odom' frame. For this, a subscriber to the 'robot1/odom' topic was used. The completion of this task results in a tree with the world and robot frame connected, as shown in Figure 1

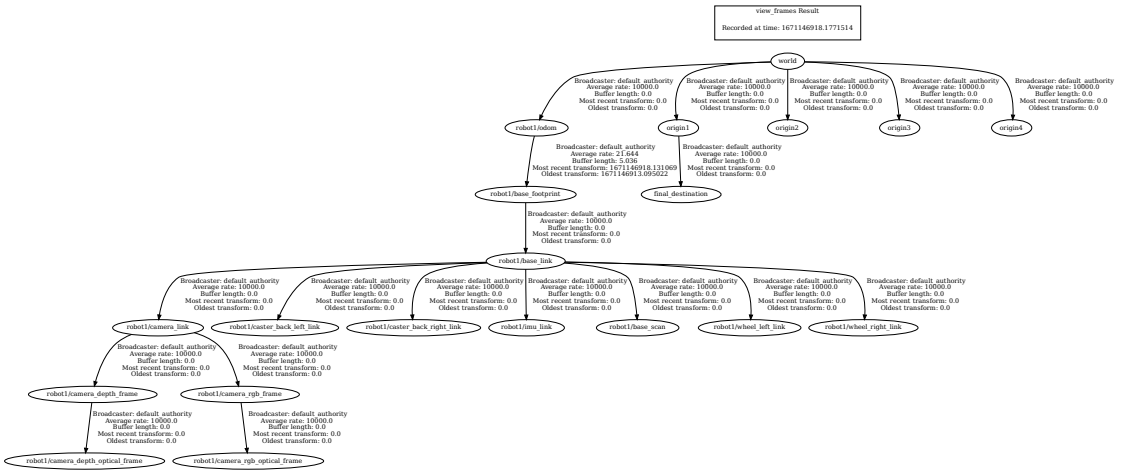
2.2 Parameters

In this task, the 'target_reacher' node needed to be started with the parameters given in 'final_params.yaml', which indicated the destination for the robot to find the fiducial markers. The parameters given in the 'final_destination' show the location of the goal to be reached after detecting the appropriate fiducial marker. Four different goals were provided corresponding to the four different fiducial markers used. All the parameters having 4 different locations of the target goal are initialized first.

2.3 Locating the Fiducial Marker

This task consists of moving the robot to the first target goal and making the robot rotate in place until detects the fiducial marker.

Figure 1: TF Tree



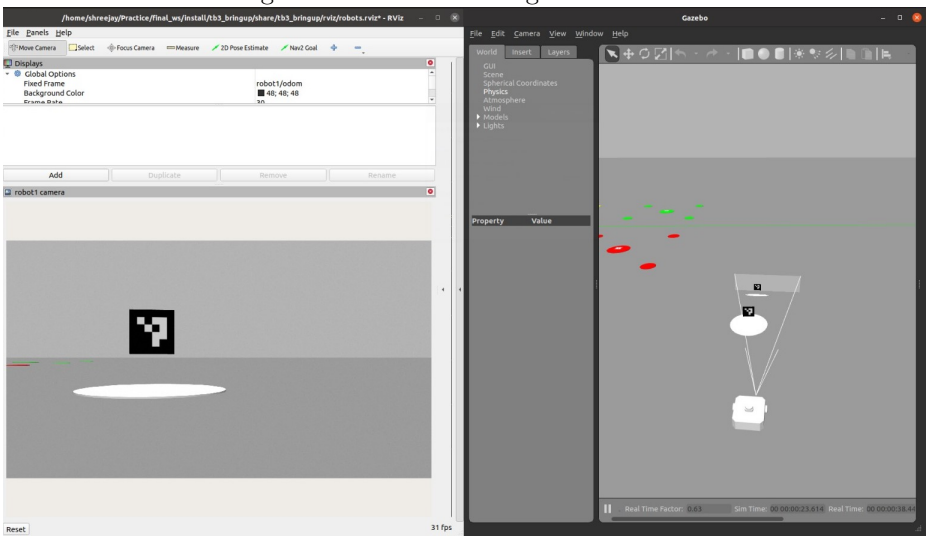
2.3.1 Move the robot to first goal

Here, the task was to move the robot to a location next to the fiducial marker for it to detect. The parameter 'set_goal(x, y)' from the class 'BotController' is used, where x, y are the fiducial marker location parameters that were initialized in 'target_reacher'. Once the goal is reached, a message is published to the 'goal_reached' topic. A subscriber to the topic is also created to receive messages.

2.3.2 Finding the Fiducial Marker

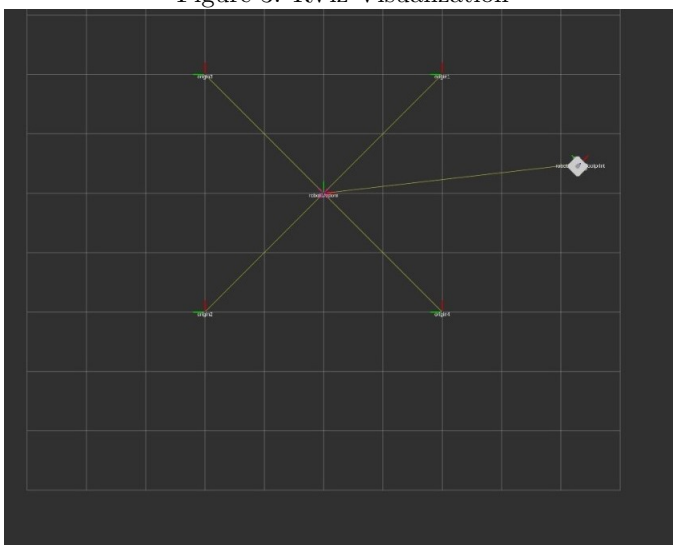
After reaching the first goal, the robot looks for the fiducial marker. To find the marker, a "twist" message is sent to the topic "/robot1/cmd_vel," which gives the robot an angular velocity of 0.2; no linear velocity is provided; which turns the robot in place till it detects the marker as shown in Figure 2. A visualization of the robot in RViz can be seen in Figure

Figure 2: Robot detecting aruco marker



3. The robot finds the fiducial marker, and once the marker is found ,"aruco_node" sends a

Figure 3: RViz Visualization



message on topic "aruco_markers".Figure 4 depicts an example of the message.
The message "marker_ids" indicates the id of the marker, which is later used to retrieve

Figure 4: Message published on 'aruco_marker'

```
---
header:
  stamp:
    sec: 25
    nanosec: 575000000
  frame_id: /robot1/camera_rgb_optical_frame
marker_ids:
- 1
poses:
- position:
    x: 0.10611783047972573
    y: -0.03761293563947064
    z: 0.25681271045873383
  orientation:
    x: 0.9877588578267713
    y: 0.0009406345645420837
    z: -0.15595293567937693
    w: -0.0031993506141633245
---
```

the coordinates of the second location robot needs to move to.

2.4 Moving the robot to second goal

This is the final destination of the robot. Here, the following tasks need to be completed in order to reach the second and final goal.

2.4.1 Retrieve the goal coordinates

Using the markerID retrieved from the aruco marker, the corresponding goal coordinates are retrieved from the parameters 'final_destination.aruco_i.x' and 'final_destination.aruco_i.y' which are given in 'final_params.yaml' file where 'i' denotes the markerID of the aruco marker.

2.4.2 Compute the Goal in Odom frame

The goal coordinates retrieved are not in the 'odom' frame but in a frame assigned to the 'final_destination.frame_id' parameters. Retrieval of these parameter values is thus required, therefore, the final goal destination needs to be transformed to the 'robot1/odom' frame. To do this, the following steps were followed:

1. Broadcasting frame 'final_destination' to the 'origin(i)' (where $i = 1, 2, 3, 4$) frame with a pose of ('final_destination.aruco.x', 'final_destination.aruco.x', 0) and orientation (0, 0, 0, 1).

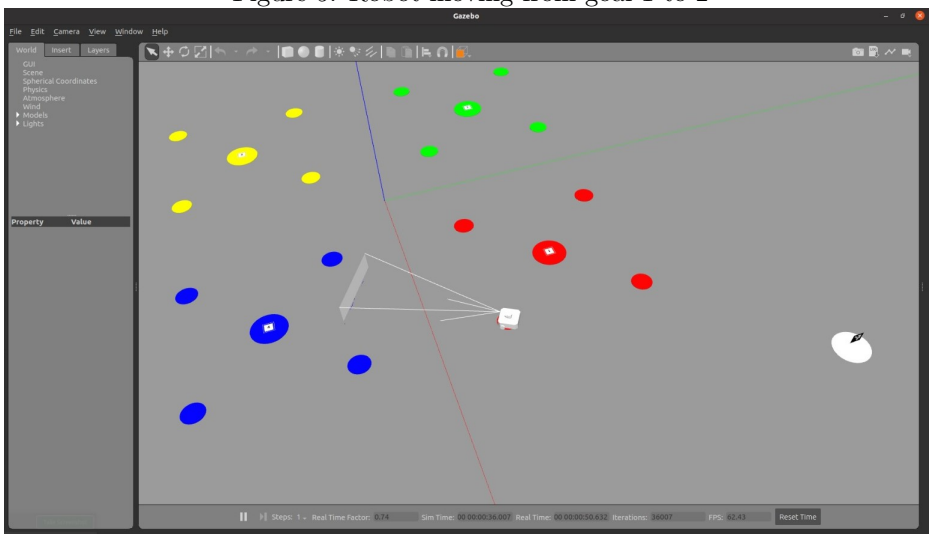
2. Writing a tf listener to create a transform between the 'final_destination' and 'robot1/odom' frames to give the pose of the final destination in 'robot1/odom' frame.

The result coordinates obtained then points the robot to one of the small targets in the environment.

2.4.3 Moving the robot to the final destination

The values were then used to send the robot to the final destination, as shown in Figure 5

Figure 5: Robot moving from goal 1 to 2



2.5 Pseudo-Codes

1. **Goal callback**

   wait for message
   if received message is "True";
   twist robot with angular vel 0.2
   set m_reached goal to "True"
   else publish goal not reached

2. **Target reacher**

   if m_reached goal is true
   check if marker id is 0; set variable 'a' to 0
   else if marker id is 1; set variable 'a' to 1
   else if marker id is 2; set variable 'a' to 2
   else if marker id is 3; set variable 'a' to 3
   broadcast marker id "a" to transform broadcaster to move robot to the final location

3 Challenges

Numerous challenges were faced in this project. Overall, 8 days were spent on this project. One of the biggest challenges that faced was to understand how simultaneous broadcasting and subscribing to frames work. One of the key challenges in figuring out this task was to write a callback function for subscriber with valid message type arguments. Overall, 3 days were spent on this task. Also, it was learned the hard way that '/' plays an important role in ROS2 topics and frames.

Another challenge was to stop the robot from looking for the aruco marker before reaching the goal. Considerable amount of time was spent understanding the origin of this issue and was later fixed by introducing more attributes that are of boolean type. Once the value 'true' was returned, then the robot would start looking for the marker.

Loading the parameters was also an issue in the implementation of callback function. It was learned that initializing parameters in the constructor was the right way to load.

The final challenge worth mentioning here was to understand a relation between a broadcaster and a listener on the final task. This issue was resolved by looking at numerous online resources on ROS.ORG and referring to the lecture slides and codes.

4 Contributions to the project

- Shreejay Badshah: connecting the tree, moving robot to location 1, identifying marker, moving robot to location 2, Publishing-subscribing topics
- Amogh Wyawahare: moving robot to location 2, Publishing-subscribing topics, Report
- Joseph Thomas: moving robot to location 1, identifying marker, Report

5 Resources

- https://ros.org
- https://docs.ros.org/en/galactic/Tutorials.html
- https://stackoverflow.com