FINAL PROJECT

Introductory Robot Programming

December 16, 2021

Instructors:
Zeid Kootbally

Students:
Prasanna Thirukudanthai Raghavan

Ninad Harishchandrakar

Sai Sandeep Adapa

Group:
22

Semester:
Fall 2021

Course code:
ENPM809Y

Contents

1 Introduction

The project is based on Urban Search and Rescue (US&R) application where after the occurrence of any disaster, an explorer robot scouts the location in which the accident took place in search of humans that might be in danger. This allows the First Responders to know the location of the victims and reach them from the information collected from the explorer.

In this interpretation of that problem we have the initial explorer robot which goes to certain locations in a map made up of walls and other objects simulating the type of environment talked about above, which contain a square fiducial marker called Aruco Marker(A unique marker type of markers that contain their own id's), in place of actual human victims whose location is captured. The robot used for this purpose is called a "turtle bot" and it has a camera mounted on top of it which allows it to check and capture the location of the ArucoMarker. Another similar robot, namely a follower then visits the locations of these markers taken earlier from the explorer in an orderly manner.

The robots were simulated in a virtual environment and with the help of a node inside the Robot Operating System, the operations of the explorer and follower were carried out.

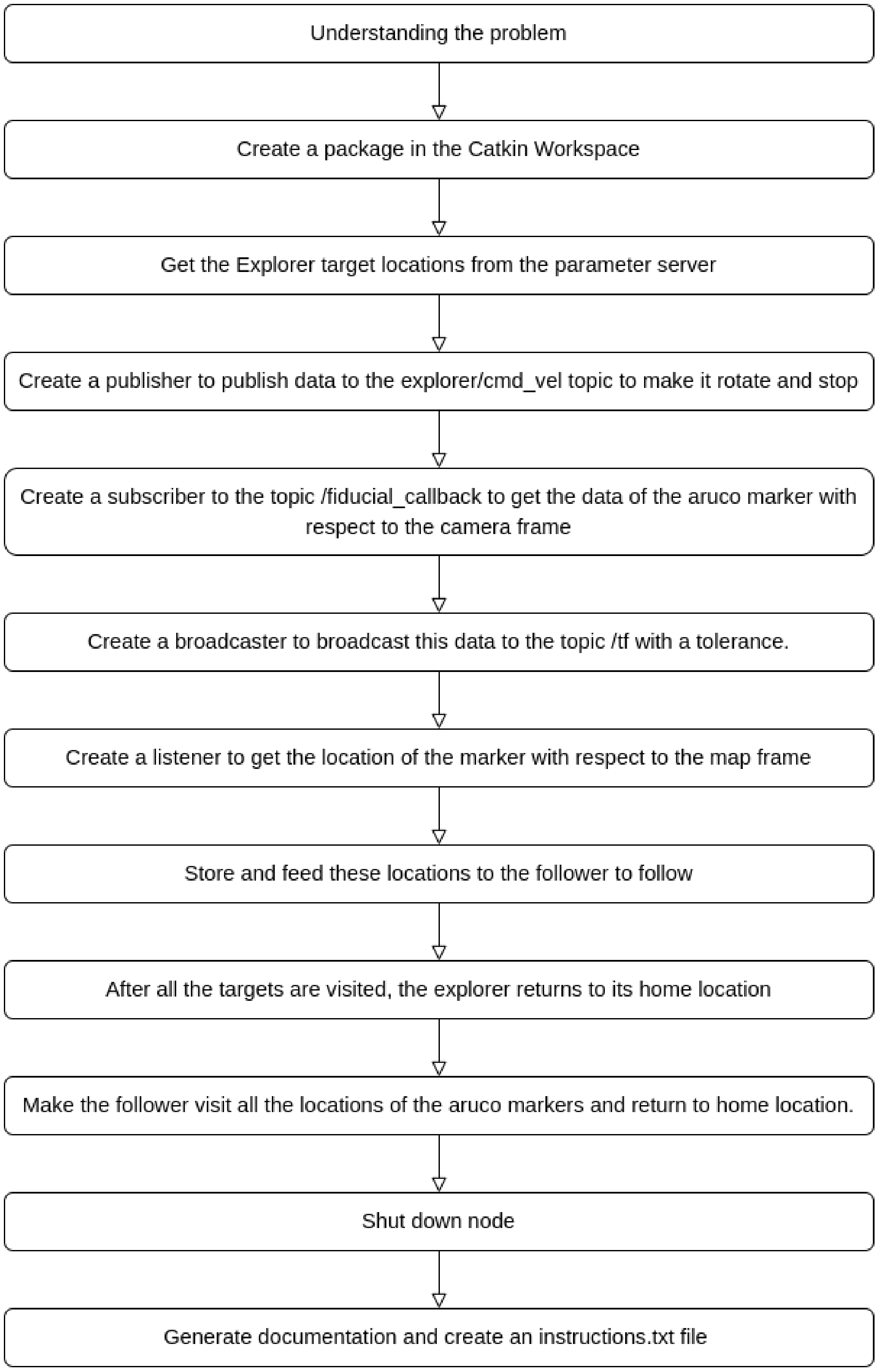
2 Approach

As given to us in the objectives we proceed with the same flow of our program where:

- We first want to make the Explorer move to the target locations.
- Then check for ArucoMarkers near the target locations.
- Then the Follower robot reaches these locations once the Explorer comes back.

To perform these tasks we implore them individually in steps. These steps are elaborated in the chart below:

2.1 Project Flowchart


Understanding the problem
Create a package in the Catkin Workspace
Get the Explorer target locations from the parameter server
Create a publisher to publish data to the explorer/cmd_vel topic to make it rotate and stop
Create a subscriber to the topic /fiducial_callback to get the data of the aruco marker with respect to the camera frame
Create a broadcaster to broadcast this data to the topic /tf with a tolerance.
Create a listener to get the location of the marker with respect to the map frame
Store and feed these locations to the follower to follow
After all the targets are visited, the explorer returns to its home location
Make the follower visit all the locations of the aruco markers and return to home location.
Shut down node
Generate documentation and create an instructions.txt file

2.2 Pseducode

```
Begin

Define client, flag arucomarker round & aruco_id

function changeGoal
{
    //This is used to output the Goal state fed to the move_base
   client
    based on the goal counter

    Based on the Goal counter, define the parameters for the MoveBase
    object to be fed to the client to send the explorer to the goal
}

function fiducial_callback
{
    //This is called when subscriber to the topic fiducial-transform
    gets messages if an aruco marker is found
    {
        Get location of an aruco marker with respect to camera
        with an offset of 0.6 and broadcast it to the /tf topic

        Save the arucomarker id in the global variable
    }
}

function main
{
    //To send the explorer to find arucomarker & make the
    followers visit these arucomarkers

    Initialize node

    Set flags, goal counters for explorer and follower, explore
    and follower client, arrays for explorer and follower goals,
    listener objects, explorer home portion, follower goal object,
    subscriber to topic/fiducial transforms, publisher
    to topic explorer/and_vel

    Get explorer target locations from parameter server and store them
    in an array

    while node is running
    {
        if (explorer goal counter < 4)
        {
            Send target location to explorer.
            Once explorer reaches target, make it rotate.
            Once Arucomarker is found in camera, stop rotating.
            Get the coordinates of the arucomarker (with offset)
            with respect to map frame by listening to the topic /tf.
            Store this inside follower goal array.
            increment explorer goal counter and set flags.
        }

        if (explorer goal counter >= 4)
        {
            Send the explorer to home position and set flag
            for explorer work done.
        }

        Once the explorer work is done
        {
            if (follower goal counter < 4)
            {
                Set follower goal or a new location got from the
                listener based on the ID of the ArucoMarker.
                Send the follower to each of these locations.

            }

            if (follower goal counter >= 4)
            {
                Set follower goal home position.
                Send the follower to home position.
                Once reached, shut done node.
            }
        }
}
// End of main
```

3 Output

Here are the screenshots from the simulation:

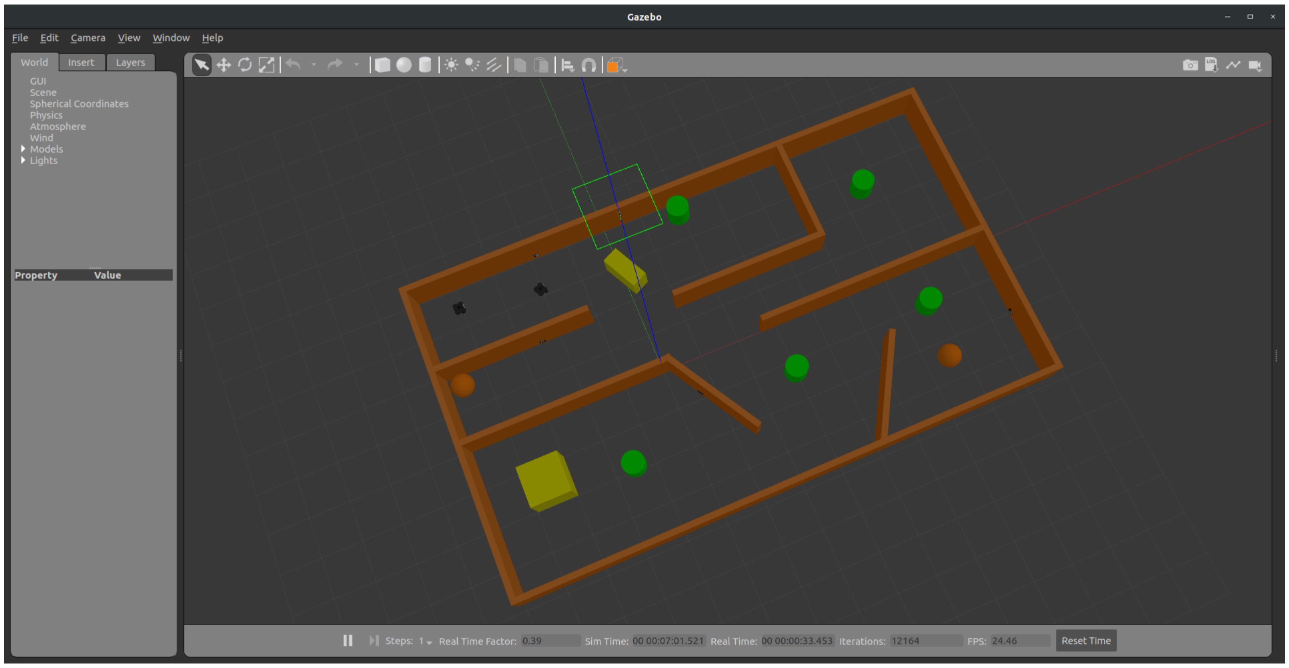

Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Steps: 1 Real Time Factor: 0.39 Sim Time: 00 00:07:01.521 Real Time: 00 00:00:33.453 Iterations: 12164 FPS: 24.46 Reset Time

Fig 1: Explorer going to the target location.

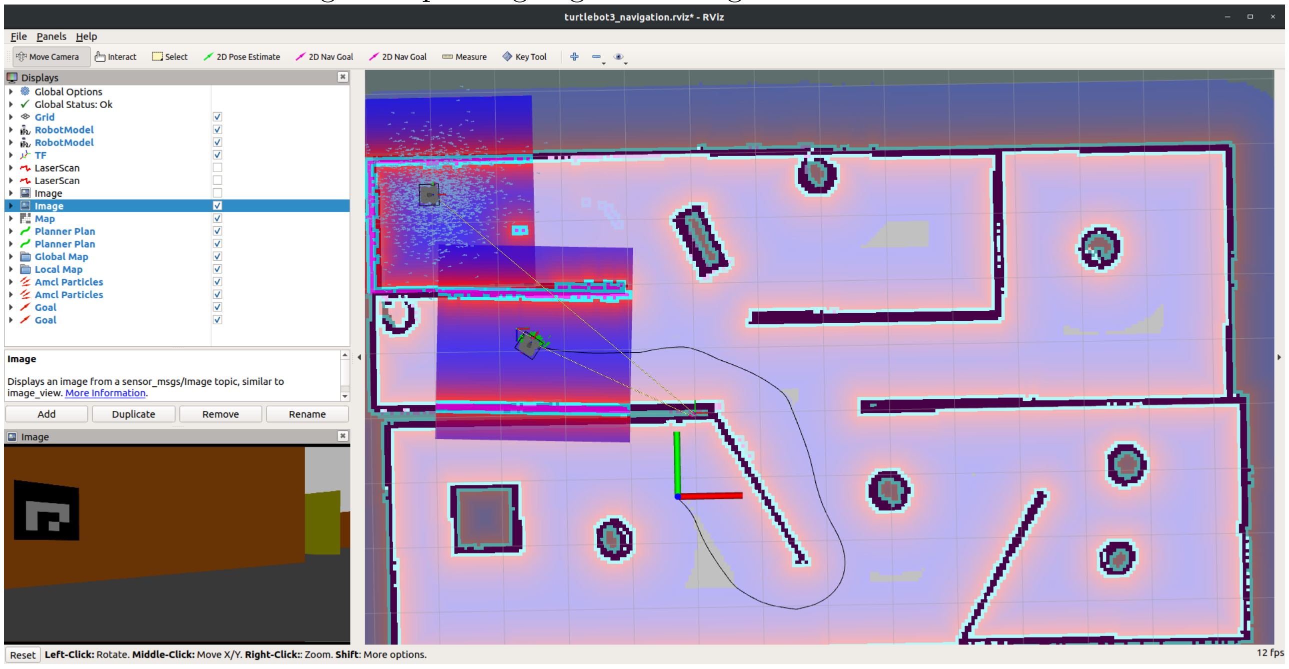

turtlebot3_navigation.rviz* - RViz
File Panels Help
Move Camera Interact Select 2D Pose Estimate 2D Nav Goal 2D Nav Goal Measure Key Tool
Displays
Global Options
Global Status: Ok
Grid
RobotModel
RobotModel
TF
LaserScan
LaserScan
Image
Image
Map
Planner Plan
Planner Plan
Global Map
Local Map
Amcl Particles
Amcl Particles
Goal
Goal
Image
Displays an image from a sensor_msgs/Image topic, similar to image_view. More Information.
Add Duplicate Remove Rename
Image
Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.
12 fps

Fig 2: Explorer finding an ArucoMarker

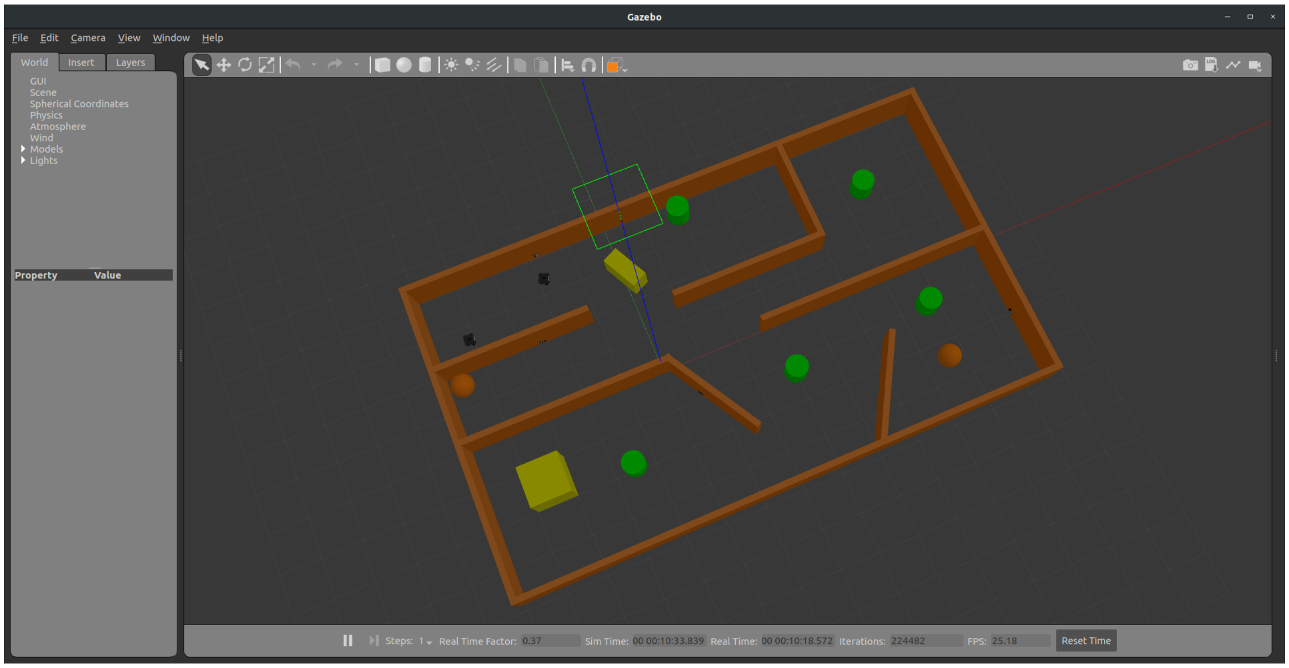

Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Steps: 1 Real Time Factor: 0.37 Sim Time: 00 00:10:33.839 Real Time: 00 00:10:18.572 Iterations: 224482 FPS: 25.18 Reset Time

Fig 3: Follower visiting the ArucoMarker

4 Challenges

- Problem with XmlRpc class: It took some time for us to understand how the XmlRpc:: XmlRpcValue class was to be used to get an array from the parameter server.

- We initially did not comprehend what's happening with the client, so we tried publishing the goal directly to the /move_base topic.

- Problem with Fiducial_msgs::FiducialTransformArray: we were unable to understand how to get the individual parameters from this class since it was an array and we were not providing an index. Usage of Flags: We tried to use too many flags due to the sequential flow which made it difficult for us to keep a track of them. We reused a few flags and were able to decrease their number and thereby track them better.

- Data type of the coordinate parameters was Float64 which should be passed as a Double in C++. We tried float as well as Std_msgs::msgs::Float64. It worked when we used Double.

- The explorer sometimes found the Aruco marker that was far away. This was rectified by setting a limit on the range of the distance of the Aruco marker with respect to the camera frame.

5 Project Contribution

- Ninad Harishchandrakar: Creating the program flow, writing the subscriber, publisher, broadcaster and listener
- Prasanna Thirukudanthai Raghavan: Getting the parameters from the parameter server, Feeding them to the turtlebots, and writing the report
- Sandeep Sai Adapa: Generating the documentation, writing the report
- Even this is an attempt at listing the individual contributions, everyone was involved in all aspects of the project.

6 Resources

- Documentation for class XmlRpc::XmlRpcValue wiki 2021h.
- Documentation for class geometry_msgs::TransformStamped wiki 2021a.
- Documentation for class geometry_msgs::Twist wiki 2021b.
- Documentation for geometry_msgs::Twist wiki 2021c.
- Documentation on how to use the Parameter Server wiki 2021d.
- Documentation on writing a Publisher and a Subscriber wiki 2021e.
- Documentation on writing a tf2 Broadcaster wiki 2021f.
- Documentation on writing a tf2 Listener wiki 2021g.

7 Course Feedback (optional)

- **Prasanna Thirukudanthai Raghavan(UID:118287546)**: The Lecture slides and the project/assignment instructions were ellaborate and easily understandable. I found it easier to learn for the quiz by doing the ”to-do” section in lectures which made it much simpler to grasp. I could have followed programming in the class better if it was a tad bit slower. The responsiveness of the Professor and the TA's is much appreciated.
- **Ninad Harishchandrakar(UID:118150819)**: Completing this course has upskilled me with the fundamentals of programming as well as the Robot Operating System. Professor Kootbally was very good at teaching all the topics and his presentations will always be remembered for being one of the best study resources. The Real World Application assignments and the Final Project were also very well made and I enjoyed working and learning with them.

- **Sai Sandeep Adapa(UID:118411460)**:
  - Fundamentals of OOP concepts were creatively covered with the Monster and Vampire Example
  - ROS Lectures were helpful to understand what was happening in the back-end.
  - Overall, the lectures and coursework were very interesting but were equally challenging to follow as a beginner to programming.

References

wiki, ROS (2021a). $geometry_msgs/TransformStampedMessage$. URL: `http://docs.ros.org/en/noetic/api/geometry_msgs/html/msg/TransformStamped.html` (visited on 12/01/2021).

— (2021b). $geometry_msgs/TwistMessage$. URL: `http://docs.ros.org/en/noetic/api/geometry_msgs/html/msg/Twist.html` (visited on 12/02/2021).

— (2021c). $move_base_msgs/MoveBaseGoalMessage$. URL: `http://docs.ros.org/en/fuerte/api/move_base_msgs/html/msg/MoveBaseGoal.html` (visited on 12/03/2021).

— (2021d). *Parameter Server*. URL: `http://wiki.ros.org/Parameter%20Server` (visited on 12/04/2021).

— (2021e). *Writing a Simple Publisher and Subscriber*. URL: `http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29` (visited on 12/05/2021).

— (2021f). *Writing a tf broadcaster (C++)*. URL: `http://wiki.ros.org/tf/Tutorials/Writing%20a%20tf%20broadcaster%20%28C%2B%2B%29` (visited on 12/06/2021).

— (2021g). *Writing a tf2 listener (C++)*. URL: `http://wiki.ros.org/tf2/Tutorials/Writing%20a%20tf2%20listener%20%28C%2B%2B%29` (visited on 12/07/2021).

— (2021h). *XmlRpc::XmlRpcValue Class Reference*. URL: `http://docs.ros.org/en/api/xmlrpcpp/html/classXmlRpc_1_1XmlRpcValue.html` (visited on 05/12/2021).