Final Project

Introductory Robot Programming

December 15, 2021

Students:

Dhyey Patel
Jeet Patel
Mrugesh Shah

Group: 24

Semester:
Fall 2021

Instructors:
Z. Kootbally

Course code:
ENPM809Y

Table of Contents

1.Introduction

This project is based on the challenges faced by the autonomous robots of the Urban Search and Rescue task force. Urban Search and Rescue team are the first responders after any disaster or any emergency situation. They respond within six hours. One of the core parts of the US&R team is their autonomous robots. Many times after a disaster it gets very difficult to locate humans trapped inside the building or any other situation. In such scenario an autonomous robot is sent inside the disaster area to locate the humans trapped inside and also create the map of the disaster site which is very difficult to locate from outside. The task of the autonomous robot is to explore the disaster affected area, create a map for the same and finally locate the trapped humans. The explored map and located victims is noted by the trained first responders and based on that they will rescue the victims.

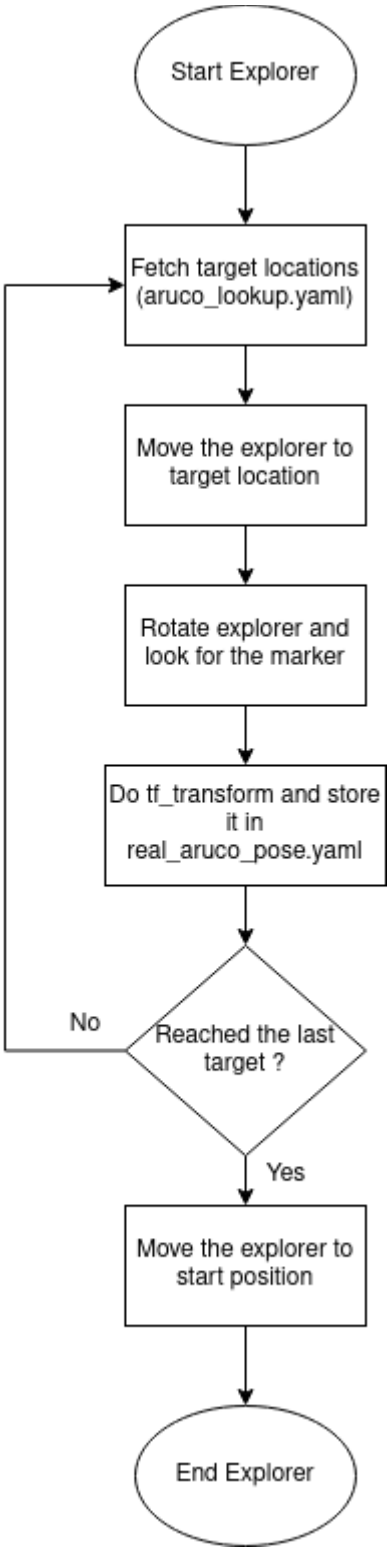
In this project the trapped humans are replaced with the Aruco markers and the autonomous robot which is tasked to explore and locate the victims is called the explorer robot and the first responder is replaced by the follower robot which is tasked to visit each and every located victim. The explorer robot is required to explore the unknown environment and simultaneously locate the Aruco markers and store their location and id. After successful completion of the exploration and identification of the Aruco markers a follower robot is required to visit the locations of the Aruco markers and complete the rescue operation.

Following is a brief overview of Aruco markers:

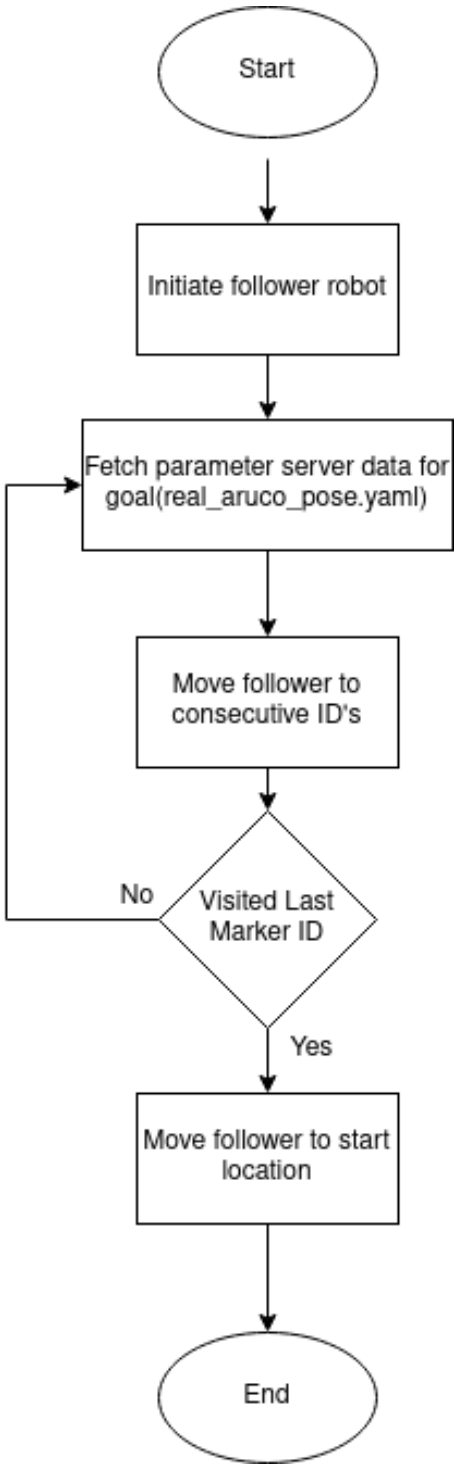
- A synthetic square marker with a wide black border and an inside binary matrix that defines its identifier is known as an Aruco marker (id). The image's black border makes it easier to spot, and the binary codification makes it possible to identify it and apply error detection and repair procedures. The internal matrix's size is determined by the marker size. A 4x4 marker, for example, is made up of 16 bits.
- It should be emphasized that a marker can be found rotated in the environment; nonetheless, the detection method must be able to ascertain the marker's original rotation in order to identify each corner indisputably. This, too, is done using binary codification.

2.Approach

Following is the flow chart of the program for the explorer as well as follower:



Flow Chart Of The Explorer Robot



Flow Chart Of The Follower Robot

Object Oriented Programming concepts were used in this project for approaching both the explorer and follower part of the project. To begin with, two classes were created called explorer and follower class for explorer and follower portions of the project respectively. Both the robots are turtlebot waffle. And the robots are equipped with a camera and also laser scan.

Given an unknown Gazebo environment and explorer and follower robots it is required to perform the exploration and following task in the given environment. Once the package is run the explorer and follower robots are initialized by default. At first the explorer robot is required to explore the unknown environment and build a map of the environment and subsequently it is required to localize itself. The robot navigates the aforementioned map automatically using Adaptive Monte Carlo localization (AMCL). For a robot moving in two dimensions, AMCL is a probabilistic localization mechanism. It uses a particle filter to track a robot's pose against a known map, as described by Dieter Fox's adaptive (or KLD-sampling) Monte Carlo localization approach. AMCL receives a laser-based map, scans and transforms messages, and generates pose estimates. AMCL initializes its particle filter according to the parameters specified when it starts up. To task both robots to the objective point, the move base is used .Once the whole environment is explored the map is stored in the memory of the robot. Also while exploring the map the explorer robot avoids the obstacle by using the laser scan.

To begin with, an explorer robot equipped with camera to detect the aruco markers has been already provided. Also the locations of four aruco markers has been provided. Once the explorer robot knows how the map region is laid up. The explorer robot's next mission is to identify the fiducial markers and detect their id and their pose, which is required to be transformed with respect to the world frame after detection. The explorer is required to visit the four target locations in order. The target locations can be retrieved from the parameter server. And the target spots of the fiducial markers have been given in the form of x and y coordinates. And these are not the exact location of the aruco markers.

After reaching the given target location the robot will rotate very slowly in order to detect the aruco marker. For this the linear velocity is reduced to zero and very minute angular velocity is provided. Now the robot will try to detect the aruco marker. Here for Aruco detection a built-in ROS package library has been utilized. This provides the ability to control the height and width with respect to the xy plane. The fiducial markers are recognized by their fiducial id which in the aruco library are shown in different forms. Once the marker is detected then the node aruco detect will publish the detected marker’s configuration on the topic fiducial_tranforms. But since the aruco marker is detected by the camera, the pose of the marker will be obtained with respect to the camera frame which is camera_rgb_optical_frame. But since the detected locations of the aruco marker has to be given to the follower robot the pose of the robot has to be converted with respect to the frame of the given map.

And it is required to transform the pose from the camera frame to the map's frame. To achieve this a broadcaster has been used. Using broadcaster is very crucial in order to get the transformation. create a child frame of the frame camera_rgb_optical_frame. Then publish the created child frame on to the topic /tf. This is required to be done because in order to compute a transform between two frames it is required that both the frames are published on the /tf topic.

Once the child frame is published on the /tf topic the transformation to the map frame was done by using tf listener. The transformed locations are stored in the real_aruco_pose.yaml. Once all the aruco markers are detected and their location has been stored the explorer robot will go back to its start location. Now, once the explorer robot comes back to the start location the follower robot will come into action. From the stored location the follower robot will travel to each location of the aruco marker. And finally after visiting all the markers in the required sequence the follower robot will simply come back to its initial position. The goal of the follower is to visit each and every aruco marker's location in the specific order, which is id 0, 1, 2 and 3. The follower is also moved using move base. And also an offset of 0.5 has been set to allow the follower to reach the location of the aruco markers.

3.Challenges Faced

- The first challenge was to store the location of all the four aruco markers in the YAML file. In our YAML file the location points were stored in form of a vector and so based on that we extracted the location points and gave it to the leader.
- The next big challenge which was faced was while detecting the aruco marker. Initially we faced difficulties to detect the Aruco marker as there were some issues in the function for detecting the aruco marker. where were getting wrong transformations.
- Also, reading the location of the four aruco markers was quite tricky. The location points were stored inside the vector and subsequently the follower robot was tasked to read them.
- Another challenge was to store the fiducial ids in sequence and appropriately send the follower robot such that it visits the aruco markers in the prescribed sequence only.
- During the fiducial transforms, the final transform between the frames was in quaternion for the rotational data which was tough to interpret. We had to convert it to cartesian to properly understand the pose values for further processing of it
- Moreover, while running multiple threads such as subscribing to fiducial transform and simultaneously running other code such as pushing a new goal to the stack. The tf frame transform was not happening. Thus, for those two separate nodes were created.

4.Work Contribution

- Dhyey Patel: Dhyey worked on detection of the aruco marker by the explorer robot. And subsequently getting the id of the marker as well getting the pose of the aruco marker. And finally transforming the pose from the camera's frame to the frame of the world.
- Jeet Patel: Jeet worked on the follower part. Reading the yaml file and subsequently making sure that the follower visits each marker in the required sequence and at the end the follower comes back to its initial location. Also, Jeet worked on debugging the main code to make sure the program runs error free.
- Mrugesh Shah: Shah coded the part for explorer to access the target positions and get the post of the explorer where it should proceed with rotation and finding the marker as well as follower goals where he tried several possible options ranging from creating a separate .yaml files and accessing / modifying them which did not work as expected and solved the problem with use of std vectors data structure.
- Equal contribution was given by all the three project partners while working on the final report and documentation .

5.Resources

- https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html
- http://wiki.ros.org/tf/Tutorials
- https://roboticsbackend.com/oop-with-ros-in-cpp/

6.Course Feedback

Dhyey Patel : Overall the course was very well structured. The course starts with very basic introduction of the C++ programming language and gradually covers the depth of the C++ programming language, which is very helpful for someone like me who is a beginner. Moreover, covering the object oriented programming aspect of C++ was one of the most beneficial parts of this course since in today's world OOP's concepts are used everywhere and are very essential. Along with that this course at the end also covers the ROS with C++ portion which is very helpful considering the robotics perspective.

Jeet Patel: This is my second course with professor Kootbally after the ENPM809E. I like the content and organization of this course along with hands-on experience in class. The material also includes the topics such as code optimizations which helps us in interview along with C++ and ROS and material was very comprehensive and easy to understand. This material is sufficient that does not require any additional

textbooks. I learned a lot in this course and this will really help me in near future in job in robotics industry.

Mrugesh Shah: The course was pretty well balanced with fundamentals as well as advanced C++ ideas. Although it would be awesome if the course were more focused on ROS services, parameter servers, or maybe doing something with different ros messages like images for future scope of using OpenCV.