PROJECT REPORT

TEAM LEONARDO

Akshath Singhal, Akhil Alexandar, Amina Tabbassum,
Daniel Smith, Syed Mohammad Asjad

1 Introduction

This report entails the development of an autonomous unmanned ground vehicle to aid in disaster relief and reconnaissance scenarios by performing autonomous exploration, mapping and target detection. The project is developed as part of the EECE 5550 Mobile Robotics Final Project.

2 Problem Statement

The project is based on the design and implementation of a complete autonomous system to perform reconnaissance in an unknown environment. The goals are to:

1. Generate complete occupancy grid map of the environment.
2. Report locations of victims (Apriltag IDs and poses) present in the environment.

3 System Architecture

The overall system architecture including the hardware and software components is explained below. Figure 1 and Figure 2 depicts hardware and software architectures respectively.

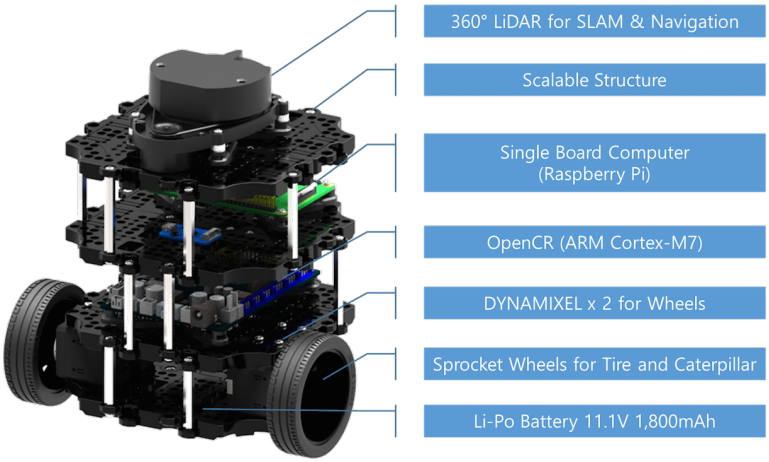

Figure 1: Hardware used (Turtlebot 3 Burger)

3.1 Hardware Platform

The implementation was carried out on the Turtlebot 3 – Burger platform equipped with a Lidar and an RGB camera. The LiDAR sensor was utilized for SLAM and navigation in conjunction with wheel odometry, whereas the RGB camera was used for AprilTag detection.

3.2 Software Architecture

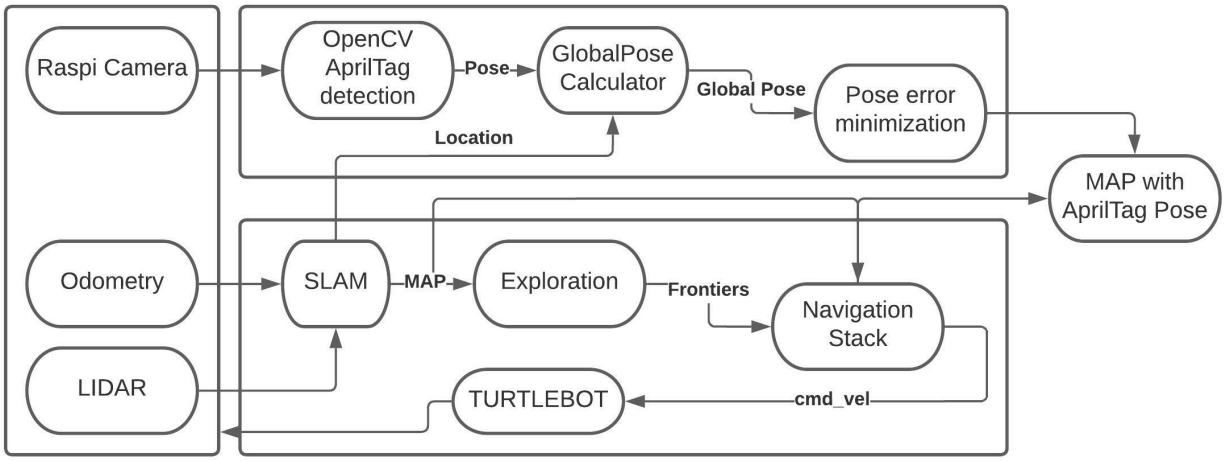

Figure 2: Software Architecture

3.2.1 SLAM with GMapping

Credit to the particle filter capabilities of GMapping, we decided to use it as our SLAM method. Having found that it is particularly effective for planar environments whilst having a LiDAR sensor it was a very feasible action. Additionally, since our exploration occurred in an environment that was of relatively pretty small size, the lack of robustness of GMapping will not affect the results.
The particle filter approach also aided in the scenario of the turtlebot running into the legs of a chair during testing which are not detected by the laser scanner, and we had to move the robot by hand. The default parameters were used for initialization.

3.2.2 Frontier Exploration

We used frontier-based exploration approach to explore the unknown areas of the map. The frontiers were defined to be regions between open space and unexplored regions of the boundary.
The frontier selection was done by choosing the frontier in the middle (this was modified in the yaml file). This gives more possibility to visualize apriltags that were behind the turtlebot since the turtlebot might need to turn around and go back towards the frontiers n+1 once it is done exploring frontier n.
The turtlebot3 github repository provided uses the frontier_exploration package for exploration, however, it is not supported by ROS Noetic. We modified the launch file to support explore_lite since we used it as an exploration package.

3.2.3 AprilTag Detection with OpenCV

OpenCV played a pivotal role in the detection of Apriltags. Using pre-developed ROS noetic packages did not seem to provide a good pose estimate of Apriltags during exploration, had more transmission delays and were not ideal for real-time display of poses on the map.
We have used OpenCV for real-time April tag detection. We followed the following algorithm for April tag detection:

1. Receive April tag image using sensor_msgs/Image message topic.
2. Convert the image topic to cv2 image.
3. Define parameters such as Apriltag size and family for detection.
4. Decompose the homography matrix using camera parameters to estimate the pose (distance and orientation) of detected tag in the camera frame.
5. Convert the tag from camera frame to map frame
6. If there are multiple detections of the same apriltag, perform error minimization to estimate the position closest to the true position. (More instances of detection of the same apriltag results in better final position estimation)
7. Display the image of detected April tag and visualize it in Rviz.

3.2.4 Navigation: DWA Local Planner

The DWA Local Planner package provides a controller to move a robot in its environment. We modified these parameters in the planner that did not fully utilize the capabilities of the turtlebot according to our requirement. The modifications were as follows:

1. Decrease the maximum available speed from 0.22 m/s to 0.15 m/s to avoid motion blur.
2. Increase minimum available speed from –0.22 m/s to 0 m/s to prevent the robot from going in reverse since the camera cannot see what is behind it.
3. Decrease the minimum and maximum rotation speed to 0.55 radians/sec to help counteract lag in the transmission of the image topic (if the robot turns away slowly, it will not have moved far away from the original pose it was in when it saw the apriltag and hence improving the pose accuracy).

4 Demonstration and Results

The test environment as shown in Figure 3 below was comprised of a total of 15 Apriltags out of which 11 were detected (we miscounted the number of Apriltags as 10 during demonstration but the robot had detected 11, the map and detected tag list is shown in Figure 4). This is the same ISEC environment where we demonstrated our results. We observed that the robot does explore nooks and crannies however, it is not guaranteed to detect all apriltags with this approach alone.

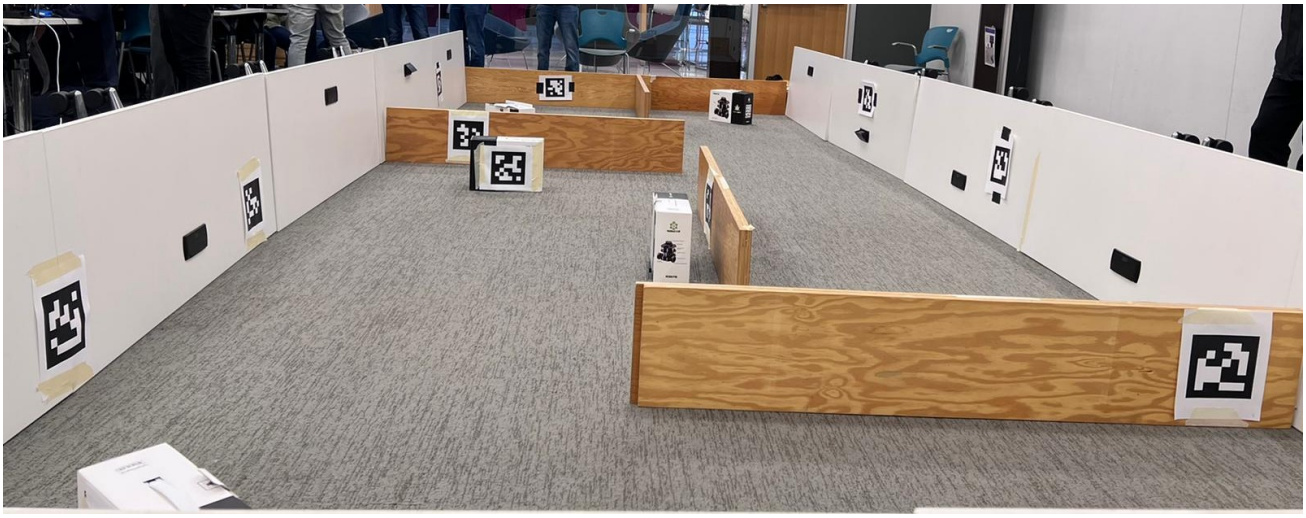

Figure 3: Testing Environment Arena

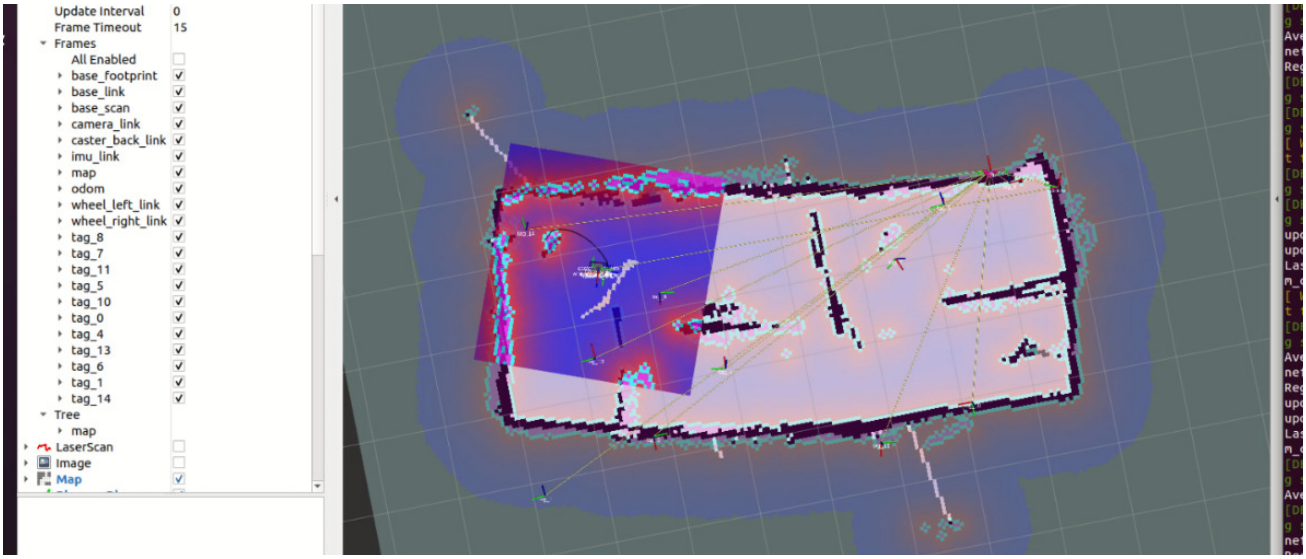

Figure 4: Generated map and AprilTag Poses visualised in RViz

5 Discussion

5.1 Other Experiments

1. We tried exploring visual SLAM methods to have more assurance that the robot will be exploring all parts of the map. Unfortunately, ORB SLAM was not supporting the creation of an occupancy grid and RTAB SLAM that used a LIDAR sensor for the aid in construction of occupancy grid required a depth camera for visuals.

2. We tried using the launch file for apriltag detection from lab02 of the course. However, we saw significantly large amounts of movement of the apriltag poses since we were displaying them in real-time and not posting them on the map once the map-making was completed. This gave us hints that our final estimations of the detections may have more error than we have using the openCV method. We believe the reason for this is because we ran the detection on the raspberry Pi itself as doing it on local PC required transfer of image data over wifi leading to time delays.

6 Future Scope

Something we wanted to try to improve the apriltag detection count and aid exploration was that after the creation of the map, we would extend the interior walls of the environment in our map such that they intersect the original outermost walls. This way we have a set of virtual rooms in our map. The robot navigates to the center of each room and does a 360 degree spin to capture the apriltags that did not appear originally in the camera's field of view. The same has been illustrated in Figure 5.

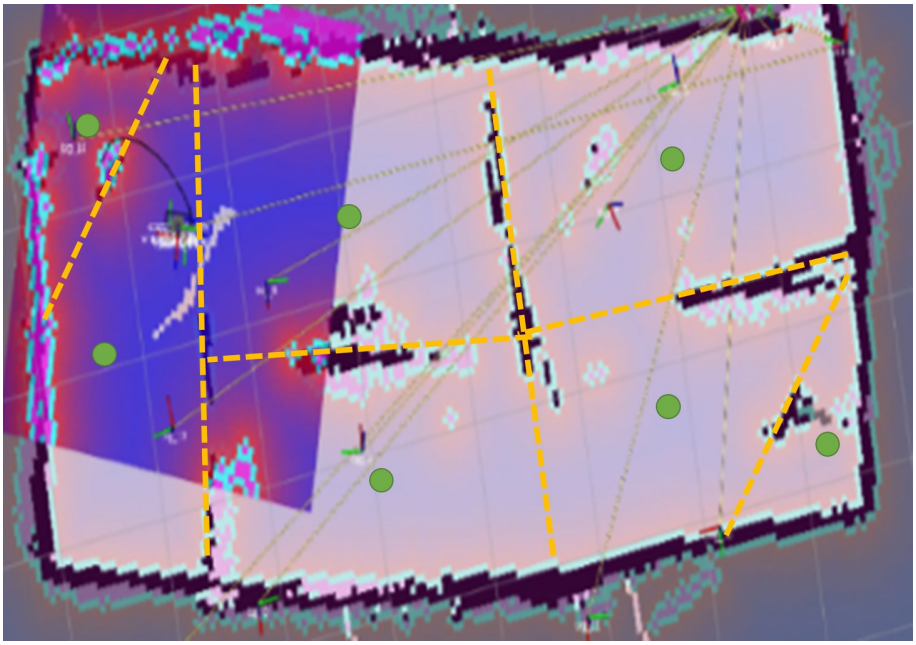

Figure 5: The yellow dashed lines represent the virtual rooms the robot will go to, and the green circles represent the center of those rooms where the robot will spin 360 degrees to see if it can find any apriltags that it missed.

REFERENCES

1. https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/
2. http://wiki.ros.org/explore_lite
3. http://wiki.ros.org/slam_gmapping
4. http://wiki.ros.org/vision_opencv
5. https://github.com/ROBOTIS-GIT/turtlebot3

APPENDIX

The noise reduction filter used on AprilTag pose estimates is based on minimization of least square errors and can be written as:

$$T_M = \min_{\hat{T}_M}(\sum_{T_O}(T_O - \hat{T}_M)^2) \tag{1}$$

where T_O is the observed tag pose and T_M is the final pose estimation after error minimization.