EECE 5550 Mobile Robotics Final Project - Team Leonardo

Anush Sriram Ramesh
MS Robotics, College of Engineering
Northeastern University
Boston, USA
ramesh.anu@northeastern.edu

Cholpady Vikram Kamath
MS Robotics, College of Engineering
Northeastern University
Boston, USA
kamath.c@northeastern.edu

Aryaman Patel
MS Robotics, College of Engineering
Northeastern University
Boston, USA
patel.aryam@northeastern.edu

Anuj Shrivatsav Srikanth
MS Robotics, College of Engineering
Northeastern University
Boston, USA
srikanth.anu@northeastern.edu

***Abstract*—Mapping potentially dangerous areas are key to planning rescue missions and mobile robots allow us to do so without any risk to human life. We propose a system that can autonomously explore and map an environment, while also simultaneously tagging the location of possible targets in the estimated map.**

***Index Terms*—robotics, SLAM, computer vision, ROS**

I. Introduction

The goal of this project was to build a system that would map a hazardous environment quickly and efficiently, while also reporting the location of potential targets within this map. To that end, we used the Turtlebot system as our mobile robot to map out an area, using April Tags [1] as stand-ins for human targets. We created a system that distributed processing between the raspberry pi on the Turtlebot and our desktop systems to ensure low latency. We tested this system first in a simulation before attempting to map a real-world location. We found that our implementation was fairly robust and managed to tag most targets in the map in the simulation.

II. Motivation

Mobile robots have been experiencing increased adoption in the last few years in applications such as disaster responses and reconnaissance in hazardous environments. This is by virtue of the growth in the processing power of hardware used in robotics which enables these systems to robustly perform online estimations of the environment better than ever before.

III. Proposed Solution

We propose an approach to tackle the problem of Simultaneous Localization and Mapping (SLAM) of unexplored environments with the additional task of finding locations of April Tags scattered in the environment. In our approach, we generate a new map using the lidar data that falls under the camera's field of view (FOV) and then mask it over the map generated by `Gmapping` to create a new map. This map that we create is then sent to `explore_lite` package to detect the frontiers, according to the camera, and perform the exploration. Thus, we have proposed a camera-based exploration technique to map the entire environment, this will be discussed in further detail in later sections.

A. SLAM methods

Gmapping is an efficient Rao-Blackwellized particle filter to learn grid maps from laser range data. This approach uses a particle filter in which each particle carries an individual map of the environment. It computes a proposal distribution taking into account not only the movement of the robot but also the most recent observation. This decreases the uncertainty about the robot's pose in the prediction step of the filter.

Cartographer is a SLAM method developed by Google, which utilizes grid-based mapping together with a Ceres-based scan matcher to reconstruct the environment across various sensor configurations [2]. Cartographer uses both IMU and odometry data to perform localization. Instead of using particle filter-based pose estimation, the cartographer accumulates the pose errors in each submap and runs a pose optimization every time to rectify and get an accurate pose. Apart from this, Cartographer also uses a scan matcher on the finished submaps and scans to perform loop closure.

B. Navigation Subsystem

For navigation, we use `move_base` as the local planner for all turtle bot movements. It uses a combination of a Probabilistic Road map and Growing Neural Gas for global path planning, and can be run in parallel with the SLAM algorithm while still building a map. The goal selection and filtering are done by the explore lite package, which is a greedy frontier-based exploration node. This node works by calculating the frontiers of free and unexplored space and greedily exploring them all until the map no longer has any unexplored space. In our use case, the map is built on the laser points that are streamed in from the lidar sensor, and hence the

`explore_lite` package will explore spaces until all areas are hit by the laser points. This immediately causes an obvious problem for us, since the April tag detection happens only via camera, and the front-mounted camera lens has a much lesser FOV (60 degrees) as compared to the lidar's full 360-degree coverage. To rectify this, we instead used a method where we registered the lidar points with the camera's current FOV, and then used only those lidar points to build the map. To correctly assign space that the robot has already seen, we implemented a simple raycasting algorithm where we set all points in that camera FOV between the robot and the lidar points to a visited or seen state. This map was then passed to `explore_lite` to generate goals for exploration.

C. Target Detection

The stand-ins for targets in our project were apriltags, which are a type of fiducial marker that can be used as a reference object. We used the `April tag` python library to detect April tags in an image, and given their relative (known) size, estimate the relative pose of the tag with respect to the camera. We chose the 36h11 tag family to generate objects in our environment.

D. Extrinsic Calibration

In the present turtle bot setup, our sensors are not aligned together and hence there is no common frame of reference for the sensors. We have to bring all these to a common frame of reference since we need to report the pose of the April tag on the map. To achieve this, we perform an extrinsic calibration between the camera and the lidar to estimate the pose of the camera with respect to the laser. The point in a camera frame ${}_CP$ can be defined in the laser coordinate frame ${}_LP$ by the following equation that describes the transformations:

$$ {}_LP = \Phi {}_CP + \Delta \tag{1} $$

where Φ is the rotation matrix that describes the camera's orientation relative to the laser, Δ describes the translation vector, these are the extrinsic parameters that we solve for that define the camera's pose with respect to the laser. A geometric constraint can be developed on the system based on the normal vector from the calibration plate. Since all the rays must fall on the calibration plate, the geometric constraint on the transformation equation defined above can be expressed as:

$$ N \cdot \Phi^{-1}({}_LP - \Delta) = \|N\|^2 \tag{2} $$

Where N is the normal defined by 3 vectors and its magnitude gives us the distance between the calibration plane and the robot. Taking several measurements of the calibration plane; solving for extrinsic parameters can be done by the Non-linear Optimization method of minimizing the Euclidean distance of the calibration plane as measured by the laser and the camera. For the number of measurements i, for every laser point j we can write the cost function as the following:

$$ \sum_i \sum_j \left(\frac{N_i}{\|N_i\|} \cdot (\Phi^{-1}({}_LP_{ij} - \Delta) - \|N_i\|\right)^2 \tag{3} $$

This cost function describes a least square problem that can be minimized by Levenberg-Marquardt method. Thus, essentially we are solving an optimization problem on the product manifold spanned by the special orthogonal group (SO^3) and euclidean group (R^3). After obtaining the extrinsic parameters, we can transform the points in the camera to the laser, thus helping us get the accurate pose of the April tag on the map. Our implementation follows from the paper [3].

E. Simulation

The gazebo is an open-source 3D robotics simulator. It integrated the ODE physics engine, OpenGL rendering, and support code for sensor simulation and actuator control. We used a custom world created to specifically test out April tag detection.

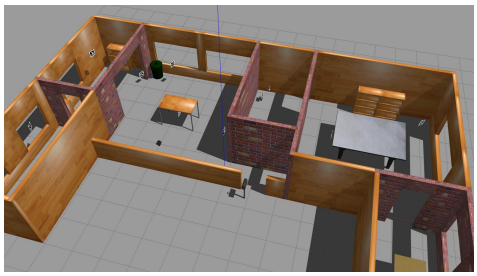

Fig. 1. Isometric view of our simulation world in the gazebo. April Tags can be seen spread around the environment.

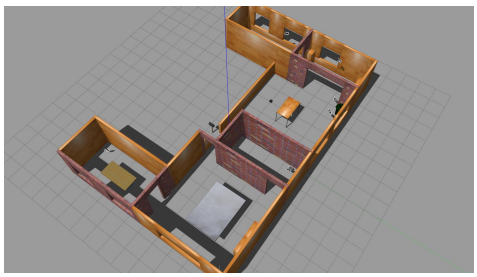

Fig. 2. Isometric view of our simulation world in the gazebo. April Tags can be seen spread around the environment.

F. Method

We create two maps as mentioned previously and efficiently use them together to map the environment and find all the April tags. Our method is structured in the following way:

- The `gmapping` node receives the data from lidar in the `scan` topic and begins to create an occupancy grid map along with publishing the current pose of the robot in the map as a transform from `odom` to `base_footprint` frames.
- We take advantage of the localization performed by `GMapping` and create an auxiliary map with laser points that fall under the camera FOV. This is determined with the help of extrinsic calibration that is in place by the node `Mapping_from_laser`.

- The map created using the camera's FOV is then masked onto the `gmapping` map and sent to `explore_lite` package.
- Once the camera sees the April tag in its FOV, it returns the pose of the April tag with respect to the camera's body-centric frame. This is then transformed to the map frame as mentioned in 4. This is performed by node `tag_detections.py`
- This map in the topic `/map_laser` is sent in as input to the `Frontier Exploration` package, `move_base` to generate and navigate to local goals.

The idea behind this approach is to make use of the camera FOV map to find frontiers, so that the turtle bot would be able to go explore remote parts of the environment while the map is being built by mapping using lidar data and hence mapping using entire laser scan, thus mapping quickly and navigating to remote areas as well.

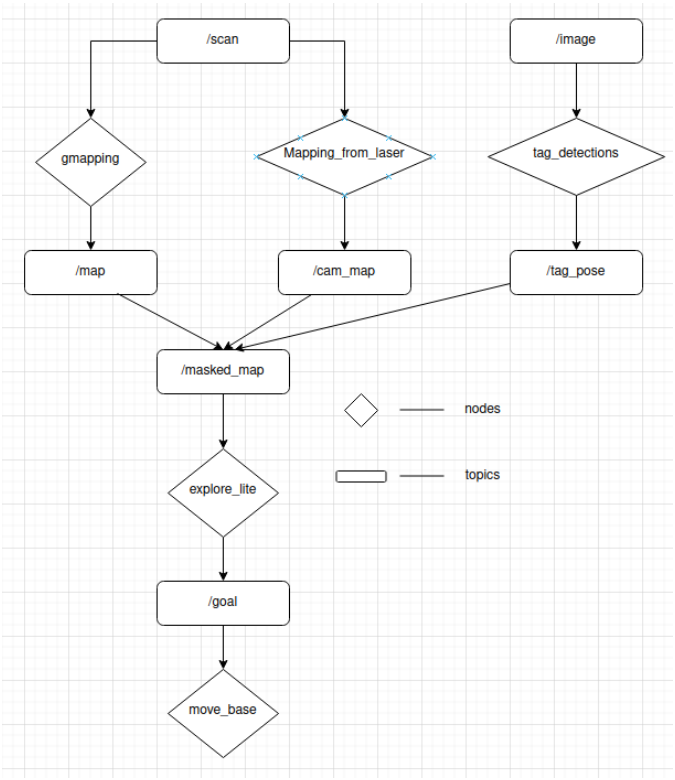

Fig. 3. Sequential working of our method

IV. Challenges

For any robotics system, the hardware and software need to go hand in hand to achieve desired results and work without any issues. Interfacing them is one of the big objectives for any project that produces useful results. Following are the issues faced, and the methodologies used by us to rectify them:

- **April Tag pose**: Obtaining the accurate pose of the April tags in the map was challenging since all the tag poses in the map seemed to be very off from their actual positions in the world. We could identify this issue was due to a wrong scaling factor that scaled the April tags to a very different position in the map and after some simple algebra, we obtained the correct Affine transformation that transforms the April tag's pose to the map frame.

$$\pi : \mathbb{R}^2 \times SE(2) \rightarrow SE(2)$$

,
π is a map that takes as input the image co-ordinates of the April tag and the pose of the camera in the map (derived from extrinsic calibration) and returns the pose April tag in the map

$$\pi_{(u,v,X_{RM})} = R_{RM} \cdot P_{TC} + t_{RM} \tag{4}$$

- **Extrinsic Calibration**: While implementing extrinsic calibration we found that the data from the laser and camera was highly misaligned. Hence we couldn't get promising results from extrinsic calibration and we also found that the lidar was not able to recognize data from nearby obstacles.
- **Interfacing exploration with Cartographer**: While trying to map the environment using Cartographer and `explore_lite`, scan messages weren't received by `explore_lite` and hence the map wasn't being built. On taking a closer look into cartographer, we found the map being published was thresholded, hence we thresholded the map using a script that accepts the present map cartographer publishes and sent it to `explore_lite`. Exploration with cartographer was achieved after making this change and the map was successfully built.
- **Noisy Lidar Measurements**: Finally when we completed our implementation and tested it out on turtlebot, we found the Lidar data returned was not able to identify obstacles nearby and the lidar could only find objects at a particular range (approx 1 meter) from itself. This was the biggest challenge that was a barrier to our work, since the Turtlebot couldn't map nearby obstacles and thinking there is free space runs into the barriers. Also, poor lidar data led to subpar localization of the turtlebot creating an even worse map of the environment. We tried a few ways to rectify this; we first tried making changes to the firmware by installing another driver that was compatible with the version of lidar given (LDS-01), but the error seemed to persist. Next, we tried changing the parameters of the lidar like max_range, min_range but this didn't help solve our issue. Finally we tried changing various parameters in packages like `AMCL`, `gmapping`, `move_base` but none of these had a better effect on the turtlebot performance.

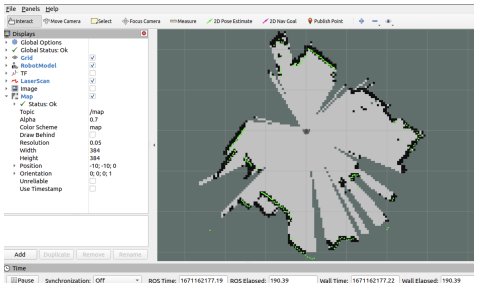

Fig. 4. Lidar capturing the object at a distance of approximately 5 ft.

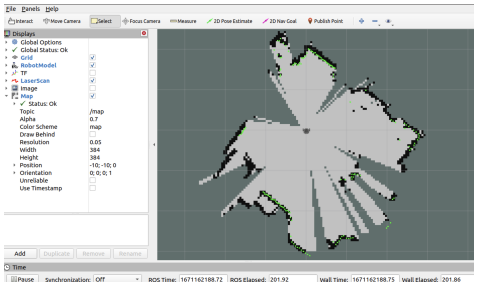

Fig. 5. Lidar not capturing the same object when at closer proximity

V. Results

- Results from the simulation run. The run was successful with the camera exploration implemented, we were able to identify and map 12/13 tags in the gazebo environment shown above.

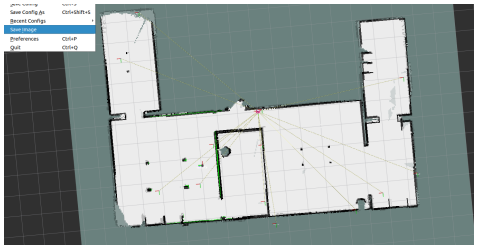

Fig. 6. Map generated by autonomous exploration with an auxiliary map created.

- Results from the run on real robot are incomplete as the robot runs into obstacles that are too close to it in the environment. Despite that, it was able to map 4/6 April tags in the environment.

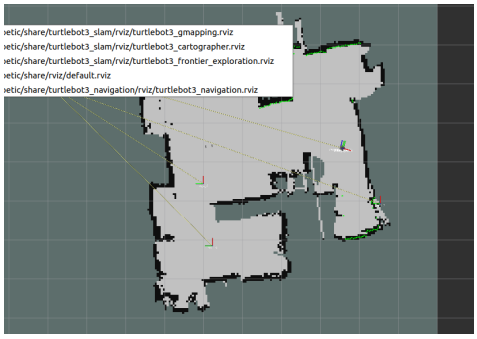

Fig. 7. Map generated in the physical robot

VI. Conclusion

An understanding regarding the working of the two most widely used sensors: camera and lidar for most of the SLAM applications, and challenges regarding hardware and software interfacing was derived from this project. More robust systems and improved SLAM algorithms need to be used in order to map better. IMU data should also be integrated for robust localization and better results. The difference between the performance of the algorithm in the gazebo world and the real world is night and day, which is a key learning. The importance of probabilistic methods for estimating the environment over deterministic methods is apparent from the results.
The code files and as well as the instructions to run the codes are included in the `ReadMe.md` of the GitHub repository: Github Repository

Acknowledgment

We thank Dr. David Rosen for being our instructor for this course. We would also like to thank the TAs, Arvinder Singh and Xavier Hubbard for their assistance configuring the robot.

References

[1] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3400–3407, 2011.

[2] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271–1278, 2016.

[3] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," vol. 3, pp. 2301 – 2306 vol.3, 01 2004.