EECE 5550 Project Final Report

Jordan Gittleman, Sameer Bhatti, Nathaniel Gordon, Shoghair Manjikian

I. PROBLEM DEFINITION

This project sough to solve various tasks related to the interaction between the Robot's Turtlebot3 platform and AWS Hospital World. Portioned into 4 Parts: mapping, navigation, guidance, and creativity. To add to the complexity of this problem, while the master robot (robot M) has a full sensor suite and ability to use SLAM, the slave robot (robot S) only has access to a simple communications channel and an RGB camera.

II. TASK DETAILS

A. Task 1

Task 1 was to map the AWS-Hospital World environment both autonomously and through teleportation while simultaneously recording the locations of 7 spot of interest (SOIs) through the strategic placement of Apriltags [1]. The Apriltags were placed in front of the main rooms of the hospital world. To accomplish this task, our group researched into existing open source repositories available online and added them to the virtual environment with the ImageMagick editing tool [5]. The RIVeR lab's Apriltag library and Hector SLAM were utilized to create a SLAM map of the essential portions of the Hospital world [2]. This map enabled the robot to effusively navigate to the SOI's and recorded the tag locations in a YAML file as a dictionary. The relative distances given by the `tag_detections` topic were converted to the global frame with a custom transformation script that mapped the Turtlebot's relative

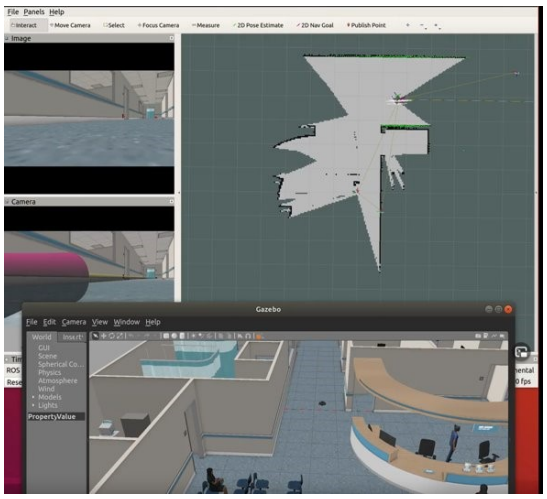

Fig. 1. Robot mapping the environment using HECTOR SLAM.

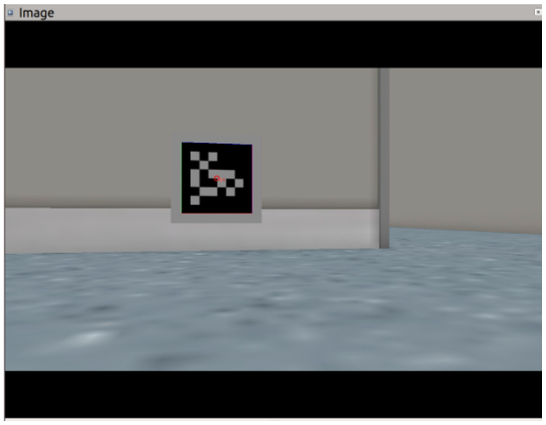

Fig. 2. View of Apriltag being sensed by robot camera, it's global location is simultaneously added to YAML file.

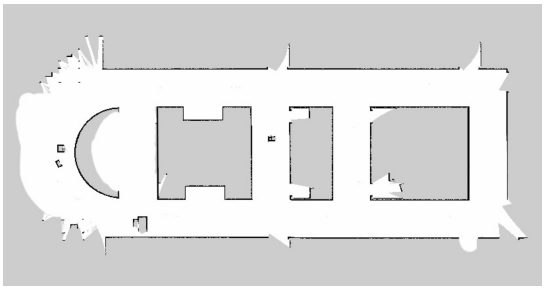

Fig. 3. View of finalized map.

tag measurements and its global odometry to a location in the global frame. To map the environment, we utilized both teleportation and autonomous navigation. For teleoperation, we employed a virtual joystick by converting its `/joy` messages to `/cmd_vel` through a custom script [4]. For autonomous navigation, the open source package explore-lite was used [3]. This package used a greedy frontier search algorithm combined with the standard SLAM costmap to explore the environment until no new frontiers remained.

B. Task 2

For task 2, the simple action client "move base" was used to manage the navigation of the robot to a particular goal [6]. The robot in this task has information on the map of the hospital as well as the tag locations of the 7 SOIs. The implemented node loads the YAML dictionary containing 7 SOIs that mapped each Apriltag ID to an coordinate position that could be given to the move base client. Additionally, a

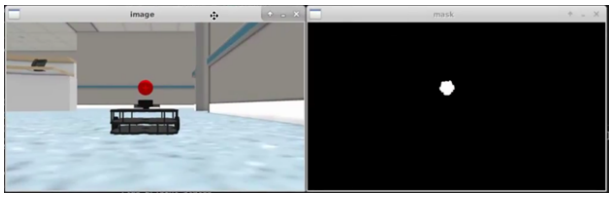

Fig. 4. View of open CV color masking alongside unmodified camera image

queuing system was implemented to handle multiple target requests in quick succession. When the robot reaches a destination, it checks if the queue is empty. If so, it navigates back to the coordinates of it's rest position, but otherwise proceeds to the next SOI. A publisher node was created to publish a topic `/target` on which the id of a target SOI is published. The node reads the YAML file to find the position corresponding to the tag then executes the navigation to the desired target and queues any subsequent target ids it receives.

A scenario was implement where the robot navigated to three SOIs in the hospital world and returned back to the rest position in the nurses station successfully.

C. Task 3

Task 3 saw a major increase to the level of challenge faced in the project. Given robot S could only have access to an RGB camera and limited communications with robot M meant the only feasible option to implement a following feature revolved around open CV. Our first iteration of the task was to convert an Apriltag to a STL schematic and include it in the Turtlebot URDF to attach it to the robot frame. This implementation however proved to be too finicky as robot S could not reliably track the Apriltag regardless of parameter tweaking. Our ultimate solution to this problem was to implement and modify a fairly common open CV introductory problem, tracking of a high contrast ball. Using a ball (bright red in this case) provided both a highly visible target to identify as well as a geometry that as uniform at all viewing angles. By Implementing an HSV mask and contour identification, we were able to measure the distance from the centroid of the ball to the center frame of the camera view and implement a proportional velocity control to maintain the proper velocity and yaw to keep the ball in frame on center.

After this implementation had proven effective. Our next steps were to design a simple script for robot S to announce it's desired tag location on the '/comm' topic, and have robot M receive the message and execute it's navigation developed previously to guide S to the target location. Once robot S is in view of the Apriltag, it's guide script will control the robot away from the master and towards the Apriltag and robot M will return to base. Our group experienced significant difficulty in this phase of the project as we could not successfully implement the parallel control of both

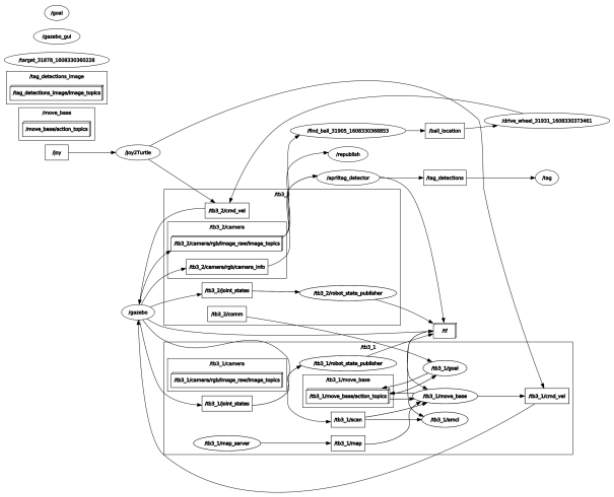

Fig. 5. View of open CV color masking alongside unmodified camera image

robot m and robot s. Launch files were altered to contain two group namespaces to launch two robots on gazebo. For robot M motion, AMCL and move base was launched with changed parameters to correspond to the correct prefixed robot. Although both robots were successfully brought up in gazebo and had correct prefixed descriptions, including for `/cmd_vel` topic, only one could be controlled at a time and differed for instance of the launch execution. While both robots could independently perform their given tasks, our program ran into a hitch most likely related to somewhat arbitrary absence of either robot's `/cmd_vel` topic. The launch file did at a period of time allow control of both velocities, however, reverted back to single robot control soon after, even though the launch file was not modified. The resultant rqt graph when both velocity commands were successful is shown in figure 5. Up until the project deadline, this remains a problem we are still investigating but currently theorize it may be due to either the AWS instance itself or some discrepancy in our remapping scheme.

D. Task 4

While task 4, the creativity task, was never reached, our team explored several promising avenues for this portion of the assignment. Our original plan was to include multiple slave robots in a line with their own independent goal locations. The master robot would be responsible for guiding all of these robots to their corresponding destinations in a last of the line first order. In theory, this queuing has already been implemented in full for Task 2 and would just require the addition of multiple robots in simulation. After learning about open CV, our group later decided it would be of great interest to attempt this task without the use of a comm channel but instead by different colored balls on each robot. Depending on the color, each robot would require a different

target SOI to be guided to.

III. CHALLENGES AND DISCUSSION

Several challenges have been addressed and overcome during this project while some could not be fully solved before the project deadline. Our group's primary initial challenge early in the semester was effective remote collaboration and development. Our original strategy to overcome this challenge was to use GitHub, but we decided later on that an individual session host over VScode coupled with a group chat and screenshare was more highly effective for real time collaboration. Our next major challenge was navigating the dependencies necessary to successfully run the AWS Hospital World. Ultimately through interaction with online forums as well as faculty from the class we were able run the world. During this time-frame, to overcome the delays brought about by issues with the world file, we experimented with simple standard gazebo worlds to test our code. Once into the full swing of the project, our main challenges involved both the open nature of the project as well as the limitation on robot S. Overall, the challenges faced during this project provided an opportunity to learn and develop our skills as roboticists.

IV. CONTRIBUTIONS

1) Jordan:

- Managed and set team meetings as well as project objectives and timeline and submission of deliverables and narrated final presentation.
- Assisted in setup of AWS world file and assets as well as debugging related tasks and packages.
- Researched into packages online for use throughout the project.
- Setup iterations of slave robot follower code and parameter tuning for both velocity control and autonomous mapping
- assisted in creation and debugging of open CV related code for tracking as well as apriltag YAML file retrieval

2) Sameer:

- Managed current working build of project and implemented Tele-operation and mapping functionality to the robot.
- Hosted environment for team collaboration sessions for Booster 1
- Completed mapping of hospital world
- Contributed to implementation of several nodes
- Implemented Apriltag detection and saving in YAML file
- Assisted in transformation formulation in localization of Apriltags
- Edited video for final presentation

3) Nathaniel:

- Managed setup and integration of GitHub and VSCode live share alongside an AWS EC-2 instance.
- Contributed to writing the launch files and managing packages necessary for the objectives of Booster 1 and assisted in debugging and finalizing of the Apriltag implementation.
- Created various ROS nodes needed for the project: go_to_goal for part 2 and track_ball and drive_wheel for part 3, and assisted in troubleshooting these components
- Learned to edit the ROS URDF files to change the Turtlebot's model

4) Shoghair:

- Contributed in implementation of launch code, turtle spawn, tele-operation,apriltags SOIs and navigation stack for Booster 1/2 and core project code.
- Assisted in debugging of code pertaining to both AWS Hospital World and the packages as noted above.
- Hosted EC2 environment for project sessions and team collaboration.
- Created the launch files for various task scenarios including multi-robot launch and debugging.

REFERENCES

[1] http://wiki.ros.org/apriltag_ros
[2] http://wiki.ros.org/hector_slam
[3] http://wiki.ros.org/explore_lite
[4] https://github.com/aquahika/rqt_virtual_joystick
[5] https://imagemagick.org/index.php
[6] http://wiki.ros.org/move_base