Devon Hills

COMP.4510

Mobile Robotics

19 December 2017

Turtlebot Navigating To Pre-mapped Locations

ABSTRACT

For my project I chose to investigate the available ROS simultaneous localization and mapping packages, in order to construct a map of a custom world, plot locations of objects within it, and then navigate to those locations with a script. Under certain restrictive conditions, I was able to achieve a basic implementation of this goal. I used a S.L.A.M. package, the main topic of study we have been building up to all semester in class. I also used a monte carlo localization package after building the map, which uses the particle filter algorithm we studied and employed in class.

INTRODUCTION

In my project I simulated a robot that could act as a delivery tool, or “butler”, to deliver an item to a specific location in a room, such as a cafe. I based my project off of tutorials from *learnturtlebot* (Silliman). I used the Turtlebot robot for this project, and chose to use the Gazebo environment to both try something new other than the default Stage, as well as better simulate a real world scenario, although my scanning, mapping, and localization was all done in two dimensions. I created custom world files, each containing at least one “table” (a simple cube shape, to show up more clearly in scans). In my original problem statement, I wanted to tag

these tables with a feature that the robot could understand and move towards without having the location stored. However, I was unable to accomplish this task, and instead designed the project around navigating to pre-mapped positions. Despite this setback, I still learned a good amount about actually implementing a SLAM package myself and attaining tangible results.

PROJECT DESCRIPTION

I started off my project by launching the default Gazebo empty world,and then entering the Gazebo editor to create my own simple simulated cafe. After creating my world, I launched the SLAM gmapping package, which creates a two dimensional occupancy grid from the laser and robot pose data. I tried autonomous navigation with a slow and steady wall-following algorithm, but I found it to be far too unreliable, I am not sure if it is because of certain settings not being tweaked correctly, hardware limitations, or a flaw in my navigation. I found many causes in online help forums for my inaccurate maps, but the best recommended solution I found to work for me was to manually drive the robot around myself. I also edited the *~minimumScore* parameter to 50.00 in gmapping, which was recommended in the ROS wiki to avoid jumping pose estimates in open areas.

I created a simpler world to test on at first, and was able to successfully realize a map using the gmapping package with the above settings and manual navigation. After the map was created, I launched the amcl package to track the pose of my robot against my known map. I used the default particle range of 100 minimum to 5000 maximum, with the initial value set to 800. After visualizing the amcl output in Rviz and determining the point my “table” (cube) was located, I passed that point to a script and was able to watch the turtlebot navigate successfully to it.

RELATED WORK

The adaptive monte carlo localization package I used in my project implements the following algorithms from the text *Probabilistic Robots:* sample_motion_model_odometry, beam_range_finder_model, likelihood_field_range_finder_model, Augmented_MCL, and KLD_Sampling_MCL (ROS wiki).

In researching my specific project goal of locating and navigating to an object (or piece of furniture) in an environment, I found that many real world robots accomplish this task using radio frequency identifier (RFID) tags. These tags have the benefit of not relying on the robot having to distinguish any visual characteristics in order to correctly identify something. It also allows for localization, and the robot not needing to know where the object is ahead of time: “First, the robot wanders around an assigned search area, making notes of wherever it picks up signals from RFID tags. Then it goes to the spot where it got the hottest signal from the tag it was looking for, zeroing in on it based on the signal strength that its shoulder antennas are picking up” (Ackerman). Using these tags could be a real world solution to my project, i.e. tagging tables in a cafe with unique RFID tags.

ANALYSIS OF RESULTS

As stated above in my introduction, my goals for this project changed as I went along and encountered difficulties. However, I set out for a more attainable goal, and accomplished it. I was able to successfully map multiple environments using SLAM gmapping, and then navigate those maps with amcl particle filtering localization to run a scripted route to up to 4 different “tables” in my “cafe”. My presentation will include a visual of the project running, but I have included captures from Rviz with amcl running which shows the robot in the process of navigating to its scripted locations in the simple world (figure 1), and then in the more advanced

world with multiple tables (figure 2). I have also included the results of my gmapping navigation, both the failed autonomous attempt, and the initial success on my smaller world.

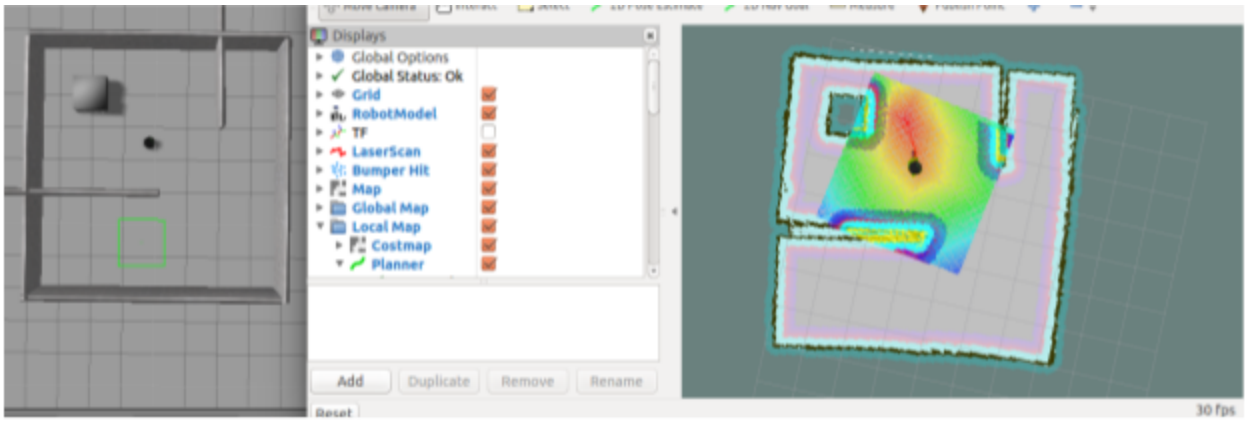


Figure 1

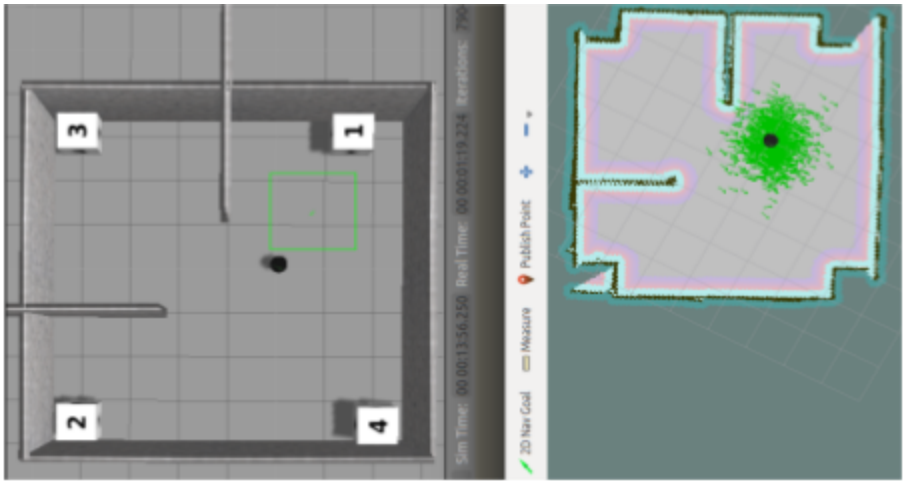
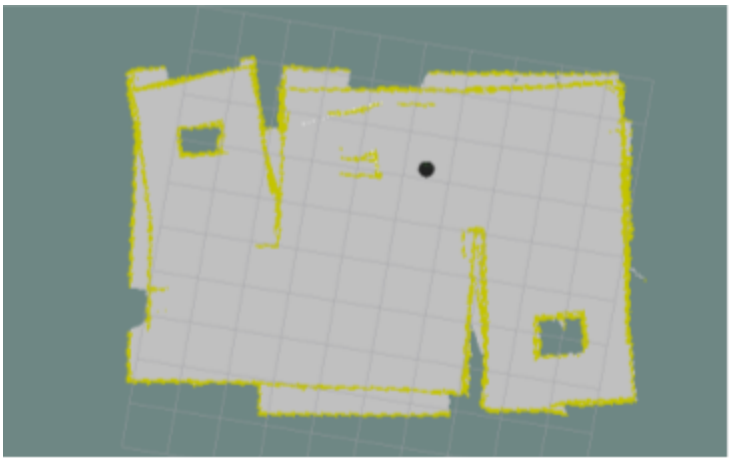
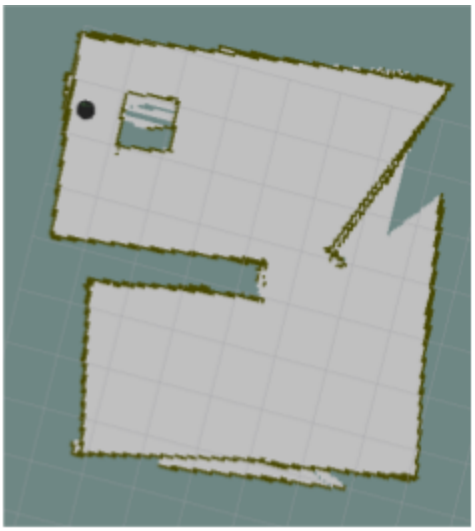


Figure 2



Autonomous navigation results



Successful gmapping

DISCUSSION

I learned that the problems of mapping, localization, and navigating an environment, even an incredibly simple one, are very difficult tasks with a huge number of moving parts all working in concert to achieve one ultimate goal. Even something as basic as moving a robot between points on a map requires a large overhead of computational work and multiple layers of complexity. Obviously others have solved much more difficult tasks than mine, but I still learned a great deal about how difficult it is, as well as a more intuitive understanding of both monte carlo localization, and simultaneous localization and mapping, by watching the output occur in real time based on my commands or scripts.

I was unable to get Vlabs to consistently work for me, and was unable to download the navigation stack without it timing out or freezing/crashing, so I installed ROS on my own virtual linux machine at home. This ended up being a necessity, and while I was successful in running it and making a functioning project on my own, I am unsure how many of my problems were due to errors or incompatibilities of my own system or installation. If I were to redesign my project, I would need more time to investigate all the intricacies of navigation, specifically with the move_base command, which I had many troubles with, and fine tune all my files for more streamlined navigation. This would allow me more time to make my project more autonomous.

CONCLUSIONS

Under all of these stated circumstances, I was able to achieve the more modest goal of simply mapping custom worlds, and navigating to known points within them. While this is not entirely impractical for real use, it would be far better and more realistic and pragmatic for the robot to be able to discover an object without knowing where it is first. Going forward, I would have liked to implement a more dynamic navigation and searching approach.

ACKNOWLEDGEMENTS

The work described in this paper was conducted as part of a Fall 2017 Mobile Robotics course, taught in the Computer Science department of the University of Massachusetts Lowell by Prof. Holly Yanco.

REFERENCES

Ackerman, Evan. “Robots Use RFID to Find and Navigate to Household Objects.” *IEEE Spectrum: Technology, Engineering, and Science News*, 22 Sept. 2014, spectrum.ieee.org/automaton/robotics/robotics-hardware/robots-rfid-find-and-navigate-objects.

“ROS Wiki.” Ros.org, Open Source Robotics Foundation, 5 Apr. 2017, wiki.ros.org/ROS/StartGuide.

Silliman, Mark. “A ‘Getting Started’ Guide for Developers Interested in Robotics.” *Learn TurtleBot and ROS*, 8 Mar. 2016, learn.turtlebot.com/

Thrun, Sebastian, et al. *Probabilistic Robotics*. MIT Press, 2006.