Autonomous Navigation and Beyond

THE ATHENIAN SCHOOL

Matthew Thompson

Mentor: Paul Ambrose

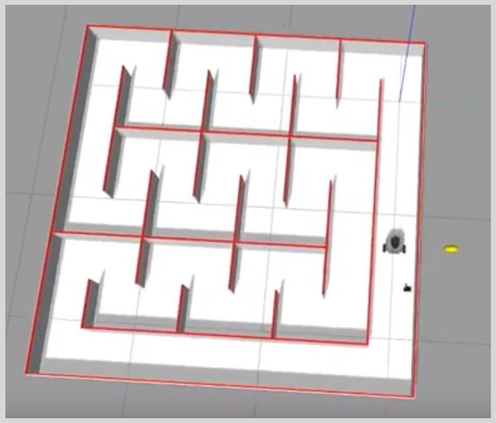
Applied Sciences, Engineering & Entrepreneurship Class

2017/2018

Instructor: Eugene Mizusawa

Zooming Out

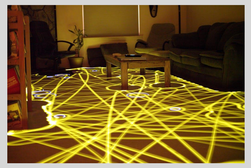
Working both in simulation and real life with a Turtlebot3, I have a software suite using ROS which can navigate through a maze while mapping it out. This software suite is easily distributable and should be easy to pick up and hack away at, allowing students and even just those curious to learn more.



General purpose of the project

If you've ever watched a Roomba, chances are you've wondered just how it manages to work its way around a room to clean a floor dotted with obstacles. Or perhaps you've seen a self-driving car with a strange spinning LIDAR on top and been fascinated to watch it route itself around an environment without any human intervention. As of recently, it's even possible that your new sedan might be able to parallel park for you. This notion of autonomous navigation around an environment captured my interest and led to the inspiration for this project.

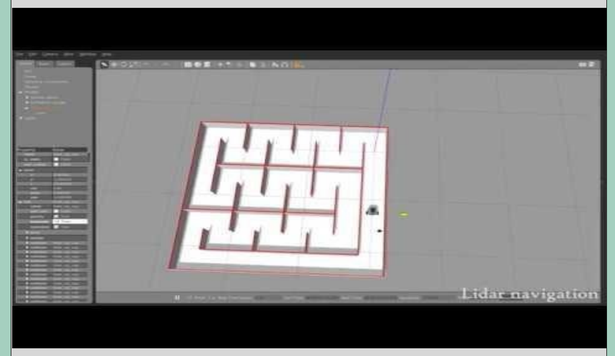
(Image credit: cleantechnica.com)



(Image credit: Chris Bartle)

My goals in the project were twofold: Firstly, I wanted to learn about this type of software, given its wide variety of applications and increasing presence in the modern world. Some use cases that I had in mind with the finished product would be implementations that could be used for the school's FIRST Robotics team, or even trying to bring it to the school's electric car.
Secondly, however, I wanted to share all of the knowledge that I gained by making an accessible and consolidated open-source software package that is very easy to get started with. There are other packages out there, but they can be a hassle and very distributed. My ideal package is easily shared, easily modified, easily understood.

Demo (1:09)



A Little About My Setup

My work has been focused on the platform of a ROBOTIS Turtlebot3 (Burger Model). The Turtlebot3 is a small robot with many positional sensors (IMU, accelerometer, servo motors), as well as a spinning LIDAR unit on top. Most of my implementations have been working with a simulated Turtlebot3 in Gazebo, as seen in the video.

Lidar Maze Navigation

The main focus of my project over the first few months was learning how to navigate it around a closed maze using just its spinning LIDAR. This was chosen as it would prove a strong proof-of-concept for understanding the systems involved, as well as one that would be testable inside of sim as well as easily replicated in a real environment.

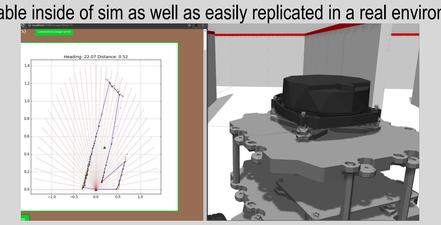


Figure 1: Point of view of the turtlebot compared with what its LIDAR is seeing.

With each LIDAR sweep, a cloud of points is returned, but in order to perform calculations with it, it must first be converted into a workable cartesian form, relative to the robot. Then, the points are sliced up and collected to form an external contour which encompasses all of the outermost points. The centroid (average) of all of these points becomes our target, and the robot adjusts its velocity and rotation according to its position relative to the robot. This tends to function very well within a maze, and should scale up well to avoid obstacles as well, as the centroid is displaced by the resulting contour distortion.

Despite its proficiency in a closed maze, however, this algorithm does not function when given just one wall to follow. Attempts to correct that issue can be found in the "Just Around The Bend" section.

Incorporating SLAM

Another facet of my project is incorporating SLAM (Simultaneous localization and mapping), which allows the robot to become stateful. Instead of simply reacting to a given set of data and discarding it, it uses its scans to generate a map for itself, and then uses its current map plus other sensory inputs to determine its position relative to the map, which it can then begin to navigate around. In my project, I have so far used a SLAM package built into the Turtlebot3 package, which maps out the environment.
Through the latter part of the year, my project focused almost exclusively on being able to navigate around any virtual environment by first exploring, and then pathing. Working with the Turtlebot3 packages, I developed a system that takes in map data, processes it on the fly, and converts it to a path leading to the nearest unexplored pixel on the map. The Turtlebot3 is then able to follow said path, and as a result discover its environment, even in a relatively open map.

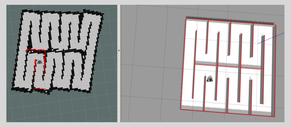


Figure 4
Side-by-side comparison of SLAM map (left) and actual environment (right)

The SLAM navigation program works as follows:
The LIDAR on the robot continually spins, collecting fresh data. Every so often, the Turtlebot SLAM module compiles that into a map. It sends a map array over ROS, as well as some other information, such as the map's origin. Another message is published occasionally as well, which is the map's offset. The SLAM module is constantly correcting the map's position due to drift, so it sends out a message containing the offset position and rotation around the origin. The simulation in Gazebo is also tracking the robot's current position and orientation. Bringing those all together, I take in the map data, crop it and process it to turn pixels into nodes, which can be navigated through. Nodes are ranked by their proximity to walls, with those a certain distance away preferred for pathfinding, due to the physical size of the robot. The robot then uses the A* algorithm to find a path to the closest unseen node. Eventually, this leads to the robot uncovering all of the map.

ROS – The Backbone

ROS (Robot Operating System) is a system which tied together all parts of my project. While not strictly an operating system, it forms an integrated pub/sub networking and package managing environment. All of the code that I have written which controls the Turtlebot3, all of the sensor data received (real or simulated), and the navigation of the simulated environment are all handled with ROS. It allowed for an extreme amount of flexibility through this project, as each node can run independently of each other. For example, the mapping node is completely separate from the navigation code, however the two both work in harmony. ROS also allows for flexibility in that any code written for the project can be run on any other device with the same sensors. A FIRST Robotics robot, given a spinning LIDAR, could use the exact same code as a Turtlebot3 to route its way around a maze.

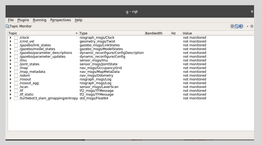


Figure 5
A program called RQT, which visualizes topics, or channels, over which messages are sent and received.

The Virtual World – Working With Simulations

Most of the project's work was done referencing a robot in Gazebo, a 3D simulator. Gazebo interfaces with and can be controlled by ROS, and can even return back information on the simulated environment such as LIDAR scans against a virtual wall. Using a maze generator, I ran all of the lidar maze navigation code through a few premade mazes, which allowed for completely reproducible testing environments, as well as the ability to pause time and move components to watch reactions and debug. Simulating the robot's environment has also caused some issues, though, mostly in the realm of falling into the trap of forgetting that the simulation is still theoretical and may not reflect the real-life behavior of a perfect tuning.

Sharing Out

This project was created with the goal of making robotics more accessible to younger and less experienced folks who are interested, and my package accomplishes that. All one needs to be able to modify and run the code that I have written is a Linux device capable of running ROS and Gazebo. With those in hand, users can run the exact same code and conditions that I have tested in by using the simulator with the same setup. All of the included software should function well straight out of the box.
Working with simulations will also make my work more accessible to others due to the fact that users will be able to simply load up a simulation rather than having to purchase an expensive robot.

Next Destination

As my project began with a goal in mind of being able to self-park a car, I believe that is one of my projects attainable end-goals much further down the line. This would most likely involve more accurate yet more complex control algorithms such as motion profiling to be able to direct the robot into a known spot along a tight path.
Also, as I have discussed in the section on SLAM, I am leaning very heavily on the Turtlebot3 SLAM package. While it is reputable, I believe that there is much more that I could be doing, and as a result I would like to change some of the methods that I use for SLAM, possibly switching to a different package such as Cartographer.

Links and Acknowledgements

Project source can be found at
https://github.com/chafla/labinav,,
https://github.com/chafla/map_routing, and
https://github.com/athenian_robotics/lidar_navigation.

Thanks to the development team at Turtlebot3 for creating the base suite underlying most of my project.

And, of course, massive thanks to Paul Ambrose, who helped me through some of the toughest parts of the project and led me to discover the joy of ROS in the first place.