GFR Driverless Competition

Tech Review

CS461 Senior Software Engineering Project

Fall 2018

Jillian Emard (Group 61/62)

November 2, 2018

Abstract

GFR (Global Formula Racing) is an international competition that intertwines expertise across several fields of engineering. With that said, the Computer Science portion primarily deals with software development and more specifically, working on SLAM (simultaneous localization and mapping). This paper will discuss the various implementations of SLAM, how to test it, and possible software solutions for an implementation known as the Kalman filter.

Contents

1 Implementing SLAM

SLAM has two important factors that contribute to its success, mapping and localization. Mapping creates an environment in which the car (robot) lives. Localization is knowing where the car is within that environment. This idea of mapping the environment while figuring out the localization is a common problem in robotics and as a result, has several different solutions that all incorporate filters. The word filter is ambiguous when used in this setting, however, it is "just a fancy word for an algorithm that takes an input (typically, a sensor signal) and calculates a function of that input" (1). For the purpose of this paper, the focus will be on the Kalman, particle, and Grid-based filters.

1.1 Kalman Filter

The Kalman filter can be found in tracking missiles, estimating the position of an aircraft, surveillance traffic, and mobile robot localization (1). It is able to predict the possible future path of a robot by taking in data points from different sensors such as lidar and camera vision. The filter increases its accuracy by re-correcting itself as new measurements become available (1). The Kalamn filter is built upon a recursive algorithm that functions dynamically with the data as it enters the SLAM environment. The Kalman filter works "to fuse the distance estimate from each feature to a matching object in the map" (1). It implements a Gaussian process. This is best modeled by a singular location where the filter has calculated the location with the highest probability. This probability is placed in the center of an oval/circular. The actual surrounding shape represents the other data points and the margin of error based upon the data received (2). Many approaches to mapping attempt to match the data points to create multiple possible locations but Gaussian takes in one state, choosing one location with highest probability,and using a normal distribution over the data points (1). The Kalman filter is less computationally involved but results in a less flexible margin - if the location with the highest probability turns out to be the wrong data point, the ability to correct oneself is nearly impossible (2).

1.2 Particle Filter

The particle filter usage in SLAM has become increasingly popular amongst the driverless community. It provides a more flexible probability analysis and makes the fewest assumptions about the location of the robot when comparing it to other types of filters (3). The particle filter also works well with models that do not fall under the linear model umbrella (2). It takes in many data points - the amount of data points fluctuates based upon the system - and calculates the probability of each state (i.e. data point) being the correct location (4). The more data points there are in a certain area, the higher the probability that the location is correct. This requires a lot of computational overhead but can result in more accurate results since it adheres to much fewer assumptions and takes into account the probability of more states. In order to avoid exponential calculations, down-sampling needs to occur (5). The particle filter "can keep track of as many hypotheses as there are particles, so if new information shows up that causes you to shift your best hypothesis completely, it is easy to do" (6). This allows the particle filter to be much more tractable than the Kalman filter. Additionally, there are many implementations for the particle filter in SLAM including FastSLAM which increases the number of data points that can be processed, decreases the memory used, as well as process both color and data points from Lidar and computer vision (7).

1.3 Grid-based Filter

Grid-based SLAM is the oldest of the three types of SLAM implementation. The area of the environment is placed into a grid and each portion of that grid is given a probability of that being the correct target location based upon the data inputs. Each block in the grid can be weighted based upon the probability of the accuracy of that grid being the correct

location (5). This is able to give a more "accurate target position uncertainty distributions [than the particle filter] since they are captured directly into the grid structure" (5) and they work with non-linear distributions, unlike the Kalman filter's Gaussian probability distribution. The grid-based filter employs multiple grids in order to complete each update as data comes in and is processed (5). This update will use the previous grids data and overlay it with the new grid data. The process will complete once the grid is "recentered" based upon the merging of the old and new data in order to produce more accurate results (5). There will be a grid for every target - in this case cone (i.e. edge of track). Each time a new grid is created it is overlayed with the correct, correlating previous grid. This correlation is based upon assigning the new data to each grid and only keeping it matched with the grid with the highest total probability with the new data points (5). The new measurements (i.e. temporary grids) will be taken away from the other grids with lower probabilities of matching (5). Like the particle filter, grid based filtering can be very computationally heavy. However, unlike the particle filter, it does not use down-sampling to avoid an exponential amount of computations (5). This grid approach also requires a lot of memory. These are some of the reason the grid-based filter is not commonly researched or used (5).

2 Testing SLAM

Testing is important in robotics and crucial in the GFR driverless car. The calculations made for the edges of the track and the car's location need to be robust and extremely accurate in order to optimize our speed and stay within the lines. Our GFR team is made up of a US team and a German team. The car we will be using currently resides in Germany; therefore, the testing we will completing here does not include track or outdoor testing. That type of testing will be completed with the car in Germany.

2.1 CG Rendering

This type of testing includes a simulation environment that we - the developers - create. It allows us to create data points from previous competitions and simulate a real-life scenario without having the car present or the sensors running (8). But it is a synthetic environment which introduces the issues with non-realistic scenarios (i.e. not accurate input). Everything about the environment is created by the software developers which may create pitfalls as they will have the same blind-spots while creating the software as they do when testing it.

2.2 Black Room

The black room testing brings the environment one step closer to reality but may reduce the usefulness of testing. This testing scenario needs a moving camera stand (with sensors to read in the data) and a dark room void of light and reflections - this reduces the noise coming into the sensors(8). Now the person running the tests will move the camera on a track in both linear directions and/or put it "on a turntable to simulate cyclic motion" (8). The margin of error for this type of testing is high and it relies on the expertise of the tester in order to maintain accuracy and a good testing environment.

2.3 HTC Vive Setup

In order to complete this type of testing, external equipment needs to be purchased. A HTC Vive track will be connected to our sensors and camera (8). This is a virtual reality setup and an environment will be used to simulate real-life scenarios which our localization and mapping software will then be tested on. This will permit "us to continuously track the orientation and trajectory of the camera using the HTC Vive tracker and log its position with much greater accuracy than the Black Room method" (8). With that said, although this more accurate than the black room method; it requires the purchase of additional equipment and a greater level of setup to create a fake environment.

3 Extended Kalman Filter

The Driverless GFR team is very large. There are currently three people working on the same area as me. With that said, we are each elaborating upon one of the implementations of SLAM and demonstrating three software options of that chosen implementation. The extended Kalman filter is a subset of the larger Kalman filter. Since the original Kalman filter only works for linear models, it is far from applicable to a driverless car. Therefore, an extended Kalman filter is used in order to implement the Kalman filter algorithm but in non-linear settings. Below are three ROS (Robot Operating System) packages that can be used to implement the extended Kalman filter.

3.1 Robot_pose_efk

This package for ROS has six steps for the implementation of its algorithm: get data from sensors, validate data, convert data to "base frame of reference" (9)- only if data is valid, wait for the rest of the data from other sensors to come in, apply extended Kalman filter to fuse the data, and convert fused data "into an odometry msg" (9). This will allow a 3D pose of the car to estimated - using a third-dimensional position and orientation in order to apply "an extended Kalman filter with a 6D model" (9). The sensor input is coming from "wheel odometry, IMU sensor, and visual odometry" (9).

3.2 Robot_localization

This package is made of two state nodes, efk_localization_node and ukf_localization_node, and includes a GPS adaptability with the navsat_transform_node (10). Additionally, it is quite versatile in the number of sensors' input it will accept. In fact, if "your robot has multiple IMUs or multiple sources of odometry information, the state estimation nodes within robot_localization can support all of them" (10). In addition it permits various message types, pre-sensor the data coming in from the various sensors in order to optimize the filtering done once inside SLAM software (10).

3.3 Hector_localization

The hector_localization package is the most difficult to find additional resources for. It appears to be the least well known of the three packages mentioned for the extended Kalman filter. This is possibly due to the fact that it is comprised of many packages that are combined to create the extended Kalman filter SLAM (11). Similarly to the aforementioned packages, hector_localization uses a variety sensors and fuses them together. However, this package provides "the full 6DOF pose of the robot" (11) and primarily uses the IMU sensoring to calculate the acceleration and angular rates of the robot. Additional features of the hector_localization include: "GPS, magnetometer, barometric pressure sensors and other external sources that provide a geometry_msgs/PoseWithCovariance message via the poseupdate topic" (11).

4 Conclusion

Ultimately, GFR is a competition; speed and accuracy are the primary goals of this driverless vehicle. Those two criteria will hold the greatest amount of weight when deciding the technology that will be used in our SLAM software. For that reason, the Kalman filter will most likely not be used due to its lack of accuracy and limited linear ability - even though the extended Kalman filter can work on non-linear models the algorithm becomes quite complex and it does not apply to extreme non-linear scenarios (12). Additionally, the grid-based filter is exponentially more complex in implementation and storage; this will cause it to most likely not be used. The particle filter appears to be the best choice for the team's criteria. However, the more data points it is given the more accurate it becomes. This is the reasoning behind the possibility of

choosing a superset of the particle filter known as FastSLAM. This computes more data points, faster using the particle filter. The more testing we are able to complete, the better the car will perform. In an ideal scenario we would be able to apply all tests listed in the section above. However, we will be taking into account the amount of time and cost that goes into each test and the ability of the test to mimic the reality of the race track. Even though there is good chance the Kalman filter will not be chosen for the SLAM implementation, there is a criteria applied to the type of ROS package that we will use no matter which filter is actually implemented. An extremely quick computational ability is crucial as well as the capacity to read as much data as possible from all of the lidar and computer vision sensors in order to create a 3D model of the car and track. This is a detailed criteria for the previously noted sections and each decision made regarding the options will ultimately decide how well we are able to compete. This stresses the importance of choosing the best tools for each situation the GFR team will encounter.

References

[1] (2008) Introduction to kalman filters and slam. [Online]. Available: http://web.eecs.utk.edu/ leparker/Courses/CS594-fall08/Lectures/Nov-20-Localization-Mapping-III.pdf

[2] J. Reich. (2015) What is the difference between a particle filter and a kalman filter? [Online]. Available: https://www.quora.com/What-is-the-difference-between-a-particle-filter-and-a-Kalman-filter

[3] U. S. N. Academy. (2015) Simultaneous localization and mapping (slam). [Online]. Available: https://www.usna.edu/Users/cs/taylor/courses/si475/class/slam.pdf

[4] C. Stachniss. Slam course - 11 - particle filters - a short intro (2013/14; cyrill stachniss). Youtube. [Online]. Available: https://www.youtube.com/watch?v=eAqAFSrTGGY

[5] T. M. Mark Silbert and S. Sarkani. (2016) Comparison of a grid-based filter to a kalman filter for the state estimation of a maneuvering target. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.823.473rep=rep1type=pdf

[6] H. d. P.-S. Kaijen Hsiao and J. Miller. (2005) Particle filters and their applications. [Online]. Available: http://web.mit.edu/16.412j/www/html/Advanced%20lectures/Slides/Hsaio$_p linval_m iller_P articleFiltersPrint.pdf$

[7] H. V. R. F. V. M. I. S. E. Z. R. D. A. G. M. B. Miguel de la Iglesia Valls, Hubertus Franciscus Cornelis and R. Siegwart. (2018) Design of an autonomous racecar: Perception, state estimation and system integration.

[8] Kudan. (2018) How to evaluate slam accuracy. [Online]. Available: https://www.kudan.eu/kudan-news/b2-evaluate-slam-accuracy/

[9] (2012) robot_pose_ekf. [Online]. Available: http://wiki.ros.org/robot$_p ose_e kf$

[10] (2018) robot_localization. [Online]. Available: http://wiki.ros.org/robot$_l ocalization$

[11] (2013) hector_localization. [Online]. Available: http://wiki.ros.org/hector$_l ocalization$

[12] V. K. Singh. (2014) What are the drawbacks of a kalman filter? [Online]. Available: https://www.quora.com/What-are-the-drawbacks-of-a-Kalman-filter