ECE4804AMR Lab Report 3

Localization

Hong Ray Teo, hteo6@gatech.edu, Fan Han Hoon, fhoon3@gatech.edu, Ethan Tse, etse8@gatech.edu

***Abstract*—This lab delves into the topic of localization. The team went through coding exercises for the Extended Kalman filter, particle filter and scan matching to get a better understanding of how these algorithms worked. Thereafter, the team successfully implemented a particle filter first in the F1Tenth simulator and then on a physical robot car.**

I. Introduction

LOCALIZATION is the process whereby a robot determines its pose relative to a known environment. Localization must be done accurately before path planning can take place, thus forming an integral part of the sense, think and act cycle in robotics. Bayesian filters are one way to perform this task of localization by recursively estimating the belief, represented as a probability density function, of the robot's pose using a motion model and measurement model. Two popular kinds of Bayesian filters include Kalman filters and particle filters. Another technique to carry out the task of localization is scan matching, where the pose that gives rise to the best alignment between the given map and the actual scan measurement is the estimated pose.

In this lab, our team explored the workings of the Extended Kalman Filter, the particle filter and scan matching via coding exercises. Then, we implemented a particle filter and tested it on the F1Tenth simulator before testing it on a physical robot car equipped with a LiDAR. Given a preexisting map, the car was able to successfully locate itself at various points in a race track.

II. Background

Team 7 worked on the lab in Klaus 2446. The team was provided with updated AMR packages by TA Matthew and starter code for the particle filter localization.

A Python prelab on the Extended Kalman filter and particle filter was provided for the team to understand the impact of different parameters on filter performance. In addition, the team was provided with a MATLAB prelab on scan matching to increase our familiarity with the algorithm.

For the Extended Kalman filter, it is noted from Fig. 1 that filter performance improves as the number of landmarks increases. This is because the filter is better able to localize the position of the car when the measurement model has more landmarks to refer to while calculating the posterior.

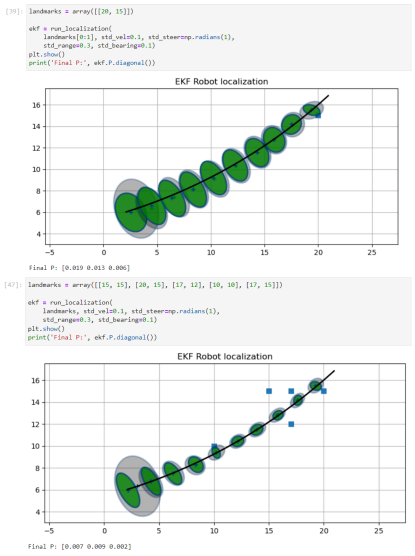

Fig. 1. Effect of having more landmarks on the Extended Kalman Filter.

On the other hand, Fig. 2 shows that increasing the standard deviation of the modelled motion model noise results in a prior with larger uncertainty but roughly the same posterior. This is because the measurement model is able to correct for the uncertainties in the motion model.

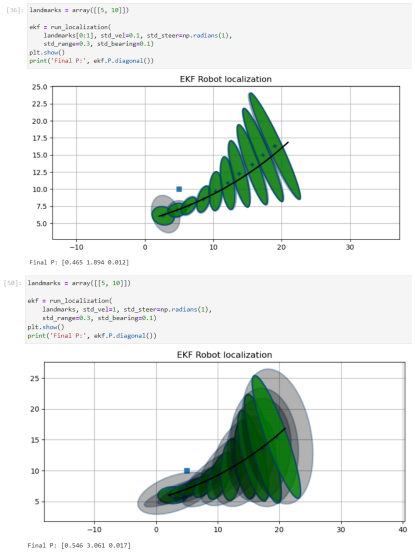

Fig. 2. Effect of increasing motion model noise on the Extended Kalman Filter.

For the particle filter, reducing the number of particles results in the predictions being completely off as seen in Fig.

3. This is a result of particle deprivation where there is no particle generated near enough to the ground truth location, causing predictions to be completely nonsensical.

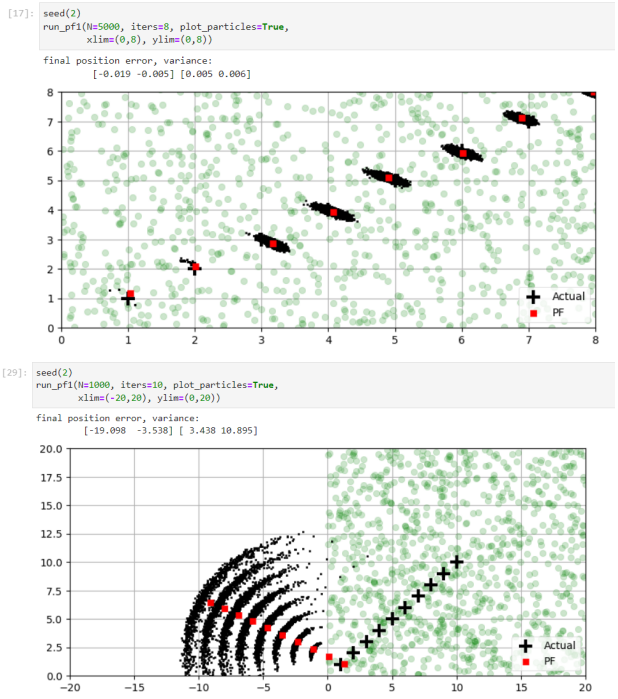

Fig. 3. Effect of decreasing the number of particles on the particle filter.

Fig. 4 illustrates how sensitive the particle filter is to a change in the modelled noise in the measurement model. By halving the standard deviation of the modelled measurement model noise, the particles converge a lot slower since almost all particles have a very low probability of seeing what the sensor actually sees, thus the weights for most particles are equally low, which means that bad particles are not filtered out during resampling.

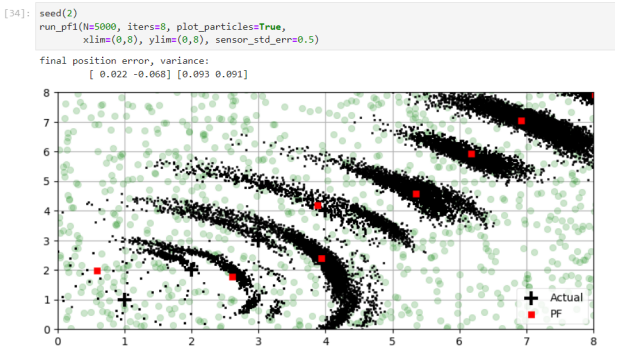

Fig. 4. Effect of halving the measurement model noise on the particle filter.

Besides that, the team also worked on a scan matching prelab in MATLAB where we tried to recover the rough map of the environment that the robot traverses.

Initially, the sensor model was activated to receive the laser scan data. Fig. 5 shows the LiDAR scan data collected by the mobile robot in an enclosed environment where obstacles and boundaries of the environment were controlled.

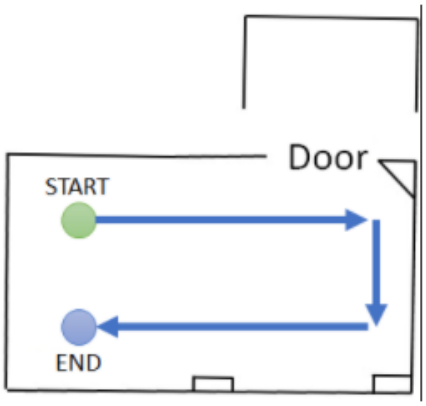

Fig. 5. Floorplan of the area and the robot's path through the space.

Then, two LiDAR scans would be picked to be matched as in Fig. 6. Despite the two scans having translational and rotational offsets, similar features are present in both scans. If the scans are not well-aligned, the current scan will be transformed to a relative pose with respect to the reference scan. The results would also be visualized to check if the scan matching was successful.

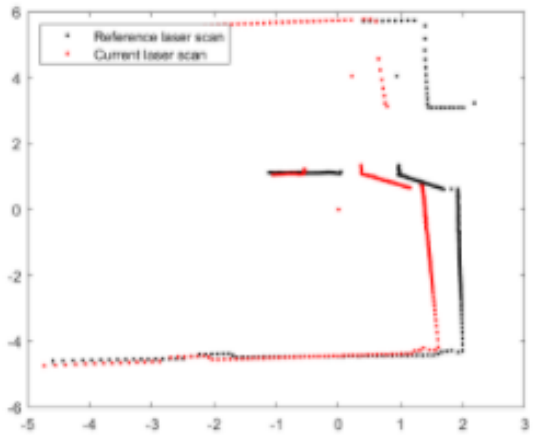

Fig. 6. Two LiDAR scans to be matched.

Thus, by applying scan matching to a sequence of scans, the team could build a probabilistic occupancy grid map that represents the rough map of the environment of the moving agent shown in Fig. 7.

By pre-allocating an array to capture the absolute movement of the robot with the 1st pose set as [0 0 0], subsequent poses will be referred to relative to the first measured scan. The team also looped through all the scans and calculated the relative pose between them. Eventually, the cumulative absolute robot pose was calculated and passed into the occupancy grid.

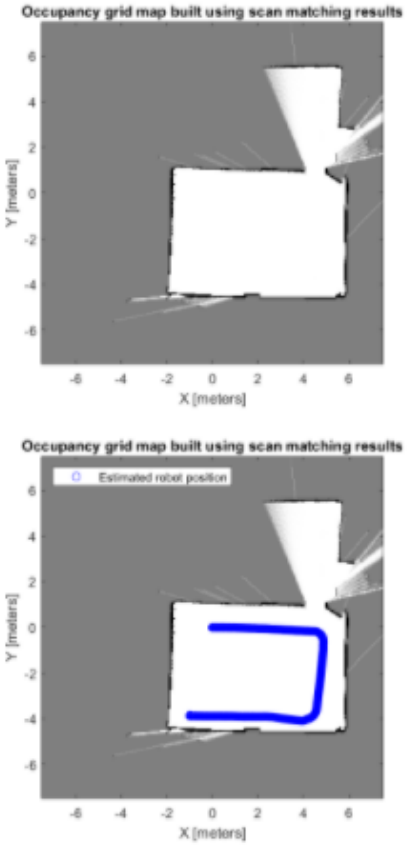

Fig. 7. Occupancy grid map populated with the laser scans and the path that the robot took through the map of the environment.

III. Procedure

Our team started by updating the AMR package for particle filter localization. Then, the team started working on the starter code provided for the particle filter and tested it out in simulation before porting it over to the physical robot.

The idea behind the particle filter was to use it to map out the environment and determine the location of the moving agent while the sensor measurements were given. In the skeleton code provided, self.particles contains the information of the x-coordinates, y-coordinates, and angle of rotation of the moving agent.

The team began by initializing the x and y coordinates of the particles from a uniform random distribution within the permissible region of the map. The orientation of the particles was initialized from a uniform random distribution from 0 to 2π which covers all possible angles.

Then, the team worked on creating the sensor model which was a superposition of four functions since there were four possibilities for the actual range measured r compared to the predicted range d. The first is the possibility of the laser bouncing off a previously unknown object nearer than the object of interest, represented by a downward sloping line (1). The second is the possibility of the laser correctly bouncing off the object of interest, represented by a Gaussian (2) due to sensor noise. The third is the possibility of the laser not bouncing off anything and the LiDAR returning its maximum range, represented by an impulse at the maximum range of the LiDAR (3). The last is the LiDAR returning a random value due to factors unaccounted for, represented by a uniform distribution (4).

$$p(r) = 1 - \frac{r}{d} \tag{1}$$

$$p(r) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(r-d)^2}{2\sigma^2}\right) \tag{2}$$

$$p(r) = \delta(r-m), \text{ where } m \text{ is the max range} \tag{3}$$

$$p(r) = \frac{1}{m} \text{ for } 0 \leq r \leq m \tag{4}$$

This weights of the various functions that make up the sensor model can be found in Table I.

Parameter	Optimal Value
z_short	0.01
z_max	0.025
z_rand	0.1
z_hit	0.75
sigma_hit	7.0

TABLE I
SENSOR MODEL CONSTANTS.

The motion model was the next crucial part of the lab where we defined how the particles would have moved based on the previous action observed from the car odometry. Fig. 8 shows the model we used. Using trigonometry, we translated the car's velocity and steering angle data into new poses for the particles in the map reference frame.

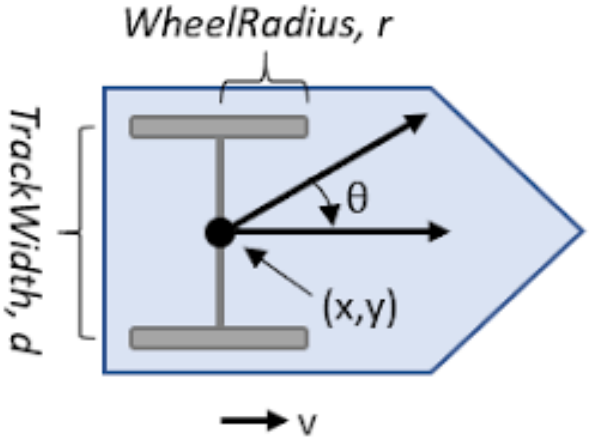

Fig. 8. Motion model used for the car.

In addition, we added a bit of Gaussian noise to the new pose of the particles in order to account for noise in the odometry data. Table II shows the standard deviations for the Gaussians used to model noise in the motion model.

Parameter	Optimal Value
motion_dispersion_x	0.05
motion_dispersion_y	0.02
motion_dispersion_theta	0.02

TABLE II
MOTION MODEL CONSTANTS.

Eventually, the team worked on resampling the particle distributions to form a new, better proposal distribution after each iteration. This step promotes particles with higher weights, or in other words, particles where the observations are more consistent with the sensor model.

We physically worked together for the entire lab, but the tasks could roughly be split as such:

Ethan handled the code for the probability model and localization while sending commands to the vehicle. In addition, he used his laptop to run the simulations as well as interface with the car. Moreover, he also tested different values of standard deviation to get the best-performing model.

Han was in charge of the hardware and assisted on debugging the code. Besides physically moving obstacles and the car around during testing, he debugged some problems that we were facing with our probability model and setting up the motion model.

Ray was responsible for guiding the team on the concepts of particle filter and extended Kalman Filter and shared documents on how different parameters impact the model. He also wrote code to perform uniform random distribution in numpy.

IV. SOFTWARE

The localization code is event-driven. Data is processed and commands are sent every time a new LiDAR scan is completed.

On initialization, the algorithm will:

1) Create a random uniform distribution of particles across the map with equal weight.
2) Generate and store a table which represents the sensor model. This is done only once and is used as a lookup table to reduce computation.

Fig. 9 shows a graph of our sensor model that was used to calculate the probability lookup table.

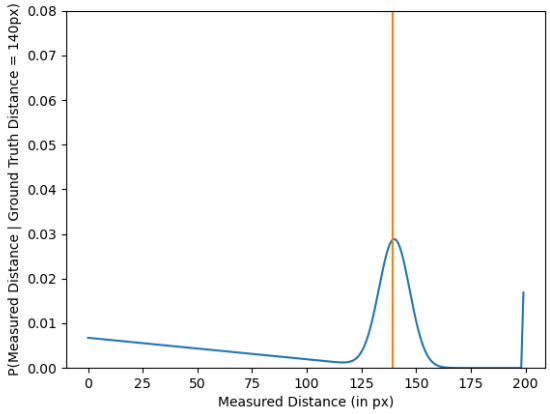

Fig. 9. Sensor model of our car given a correct value is at 140 pixels. The blue line represents probabilty and the orange line marks the 140 pixel correct measurement.

The algorithm then performs localization every time the node is updated with new LiDAR information. The localization process is as follows:

1) Create a proposal distribution by resampling from the old particles and their weights.
2) Apply the motion model to update the proposal distribution.
3) Apply the sensor model to update the proposal distribution.
4) Normalize the weights of the particles.

V. RESULTS

Team 7 was able to successfully perform localization via particle filtering. Preliminary testing was conducted in the F1Tenth simulator, where the team was able to develop our algorithm. Fig. 10 successfully demonstrates localization of the car in the F1Tenth simulator.

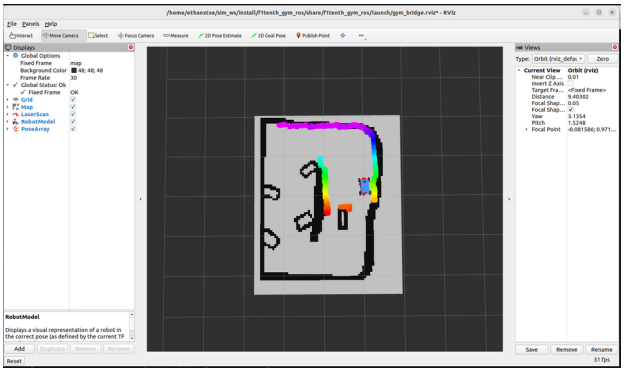

Fig. 10. Image of successful localization in f1tenth simulator.

Following success in simulation, we began to implement our particle filter code onto our actual vehicle. Unfortunately, we quickly ran into problems performing accurate localization, as the particle filter would often place our car outside of the map. After extensive debugging with assistance from ECE 4804 TA's Matthew and Sid, we determined two errors: Firstly, the

x and y coordinates of the given map were flipped, causing some particles to be initialized outside of the allowable region in the map. Secondly, there was a hardware problem with our LiDAR being misaligned with the horizontal plane. By plotting the laser scan of our LiDAR in rviz, we found that our LiDAR would get measurements from outside of the track itself wherever it was tilted up. Despite adding washers, this was not a problem we could quickly fix in the lab, and ultimately we ended up using the TA vehicle for the task. Fig. 11 shows our algorithm successfully estimating the pose of the TA car in the track.

Fig. 11. Image of successful localization in the real world.

VI. Conclusion

The objective of this lab was to design a particle filter that could perform localization of our car within the race track set up by the TA's. Our particle filter was based off a template created by the TA team. We successfully got the particle filter up and running in the F1Tenth simulation, and played around with sensor model and motion model constants to get relatively precise localization.

Our main issues arose when we tried to port over our code into the real world. In particular, we ran into a hardware issue with our LiDAR being tilted at an angle. As discussed in the results section, we ported over our code to the TA vehicle and successfully navigated through the track.

Acknowledgment

The authors would like to thank Dr. Michael West, Matthew Woodward, and Sidhant Gulati.