Lab 5 Report: Localization

Team 3

Jinger Chong
Nina Gerszberg
Ethan Hammons
Juan Rached
David von Wrangel

6.141 Robotics: Science and Systems

April 4, 2022

1 Introduction

Ethan Hammons

In previous labs, we implemented wall following, a safety controller, and a parking controller. This allows our robot to avoid obstacles and park in front of designated objects. Our next step, which we accomplished this lab, was to implement the ability for the robot to localize itself and draw a map of the surroundings at the same time. We implemented and tested two algorithms, Monte Carlo Localization and SLAM. Our MCL implementation consists of three main parts: the motion model, the sensor model, and the particle filter. SLAM was implemented through google cartographer. Through successfully deploying these algorithms we now have multiple ways to get our robot to localize and map its environment.

2 Monte Carlo Localization

2.1 Motion Model

Juan Rached

Given initial pose x_{k-1} and odometry Δx such that,

$$x_{k-1} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix}, \qquad \Delta x = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix}$$

We convert them to the expanded pose form so that,

$$\tilde{x}_{k-1} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \\ 1 \end{bmatrix}, \qquad \tilde{x}_k^{k-1} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \\ 1 \end{bmatrix}$$

And use the pose transform matrix,

$$\tilde{x}_k^w = \begin{bmatrix} R_{k-1}^w & \tilde{x}_{k-1}^w \\ 0 & 1 \end{bmatrix} \tilde{x}_k^{k-1}$$

Here R_{k-1}^w is a 3x3 rotation matrix, so our transform matrix is a 4x4. $\tilde{x}_k^{k-1}$ is our Δx, and $\tilde{x}_k^w$ is our pose at the next step k, a 4x1 column vector. The w superscript indicates that the poses and rotations are relative to the world frame. Expanding the transform we get,

$$\tilde{x}_k^w = \begin{bmatrix} cos(\theta_{k-1}) & -sin(\theta_{k-1}) & 0 & x_{k-1} \\ sin(\theta_{k-1}) & cos(\theta_{k-1}) & 0 & y_{k-1} \\ 0 & 0 & 1 & \theta_{k-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \\ 1 \end{bmatrix}$$

Which is the expanded pose of the racecar with respect to the world frame at time-step k,

$$\tilde{x}_k^w = \begin{bmatrix} x_k \\ y_k \\ \theta_k \\ 1 \end{bmatrix}$$

Converting it back to the standard pose form we get the vector we use in our code,

$$x_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}$$

2.2 Sensor Model

Jinger Chong

The sensor model determines the probability of recording a given sensor reading from a certain position in a known map. This is done by taking the range measurements from the liDAR scan and evaluating the product of the probabilities of each measurement. Mathematically, this is equivalent to

$$p(z_k|x_k, m) = p(z_k^{(1)}, ..., z_k^{(n)}|x_k, m) = \prod_{i=1}^{n} p(z_k^{(i)}|x_k, m)$$

For each range measurement, there are four types of probabilities considered. p_{hit} accounts for the detection of known objects in the map. p_{short} accounts for accidental measurements right at the liDAR exit, such as hitting the robot or a scratch on the liDAR. p_{max} accounts for missed measurements due to the light not reflecting back to the liDAR. Lastly, p_{rand} accounts for a random incorrect measurements in any location. Their values are computed using the equations below.

$$p_{hit}(z_k^{(i)}|x_k, m) = \begin{cases} \eta \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(z_k^{(i)} - d\right)^2}{2\sigma^2}\right) & 0 \leq z_k^{(i)} \leq z_{max} \\ 0 & \text{otherwise} \end{cases}$$

$$p_{short}(z_k^{(i)}|x_k, m) = \frac{2}{d} \begin{cases} 1 - \frac{z_k^{(i)}}{d} & 0 \leq z_k^{(i)} \leq d \text{ and } d \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$p_{max}(z_k^{(i)}|x_k, m) = \begin{cases} \frac{1}{\epsilon} & z_{max} - \epsilon \leq z_k^{(i)} \leq z_{max} \\ 0 & \text{otherwise} \end{cases}$$

$$p_{rand}(z_k^{(i)}|x_k, m) = \begin{cases} \frac{1}{z_{max}} & 0 \leq z_k^{(i)} \leq z_{max} \\ 0 & \text{otherwise} \end{cases}$$

The four distributions are then combined by taking a weighted average according to the equation below. For this specific laser scan, the parameter values are $\alpha_{hit} = 0.74$, $\alpha_{short} = 0.07$, $\alpha_{max} = 0.07$, $\alpha_{rand} = 0.12$, and $\sigma = 8.0$.

$$\begin{aligned} p(z_k^{(i)}|x_k, m) = \alpha_{hit} \cdot p_{hit}(z_k^{(i)}|x_k, m) + \alpha_{short} \cdot p_{short}(z_k^{(i)}|x_k, m) \\ + \alpha_{max} \cdot p_{max}(z_k^{(i)}|x_k, m) + \alpha_{rand} \cdot p_{rand}(z_k^{(i)}|x_k, m) \end{aligned}$$

These overall probabilities are ultimately used to reduce the particles that the motion model tries to spread out by selecting only those with high enough probability.

2.3 Particle Filter

Nina Gerszberg

The particle filter combines the motion and sensor models discussed above. The purpose of the motion model is to give the car movement information. The purpose of the sensor model is to fix the noise added by the motion model and assign probabilities to the each of the points. The particle filter will then sample these points, repeating points with higher probabilities and eliminating points

with lower probabilities. Noise is a critical component of this algorithm. We've added noise to cancel the noise created through dead-reckoning, thus allowing the car to find its true trajectory.

The particle filter starts off with 500 points scattered around the car. With each iteration, the points get closer and closer to the car until eventually they converge to the car position and form a pretty good estimate of the car's location.

3 Experimental Evaluation

Jinger Chong

To evaluate the performance of the motion model and the particle filter, the predicted and actual trajectories of the car in simulation were compared. The trajectories were obtained by recording the `/odom` and `/pf/pose/odom` topics and extracting the X and Y positions from the pose. Figures 1 and 2 show the plots of these trajectories. Evidently, the particle filter can predict the position of the car more precisely than just the motion model.

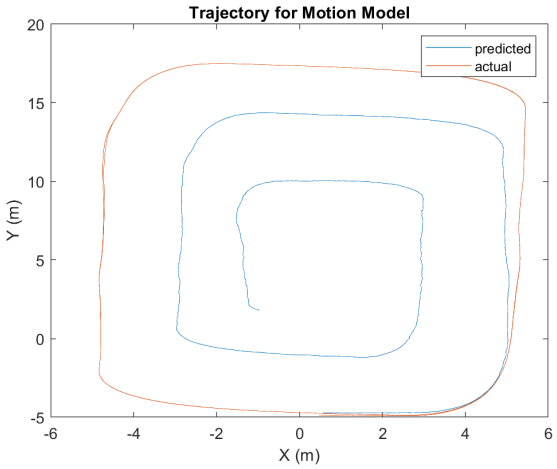

Figure 1: Predicted and actual trajectories using only the motion model.

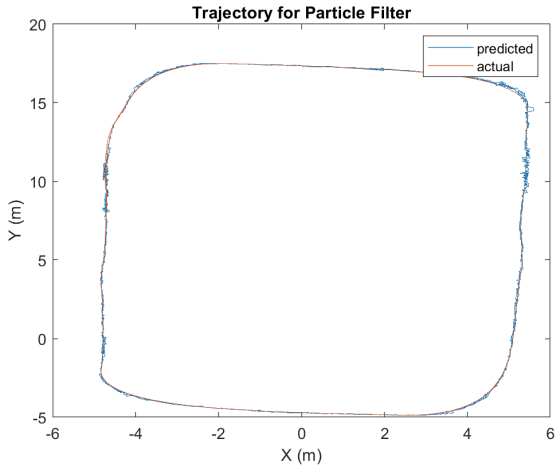

Figure 2: Predicted and actual trajectories using the particle filter.

Quantitatively, the errors were then computed by taking each actual position and finding the Euclidean distance to the nearest predicted position. While this does not guarantee that each pair of predicted and actual positions necessarily correspond with each other in the model, it eliminates any additional errors caused by mismatch in the start time and publishing rate of the two nodes. Figures 3 and 4 show the plots of these errors with respect to time. The average error of the motion model is 1.239m, while that of the particle filter is 0.023m.

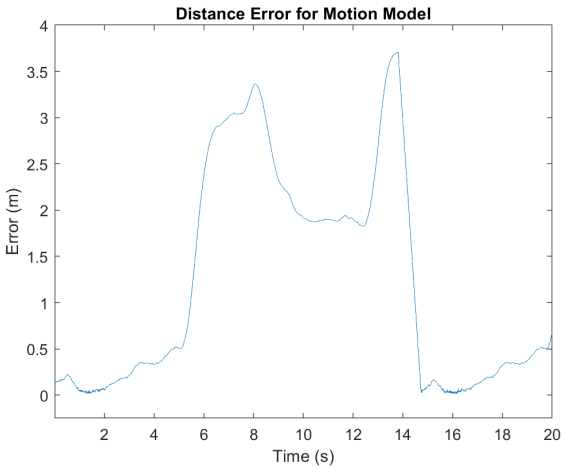

Figure 3: Euclidean distance between predicted and actual position in simulation using only the motion model.

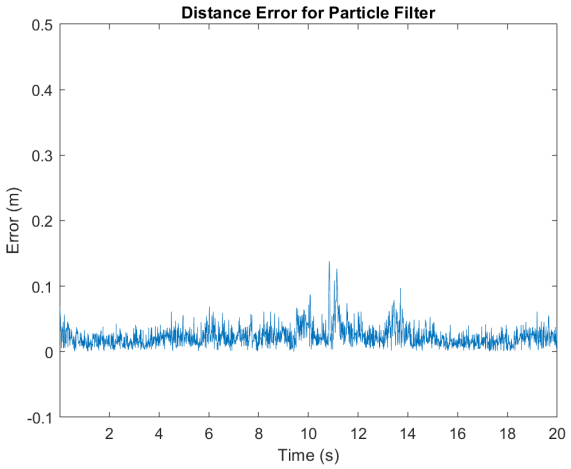

Figure 4: Euclidean distance between predicted and actual position in simulation using the particle filter.

4 SLAM

David von Wrangel

Simultaneous Localization and Mapping (SLAM) is a technique that solves the following optimization problem.

$$min_{x_0,..,x_k,l_1,..,l_m} \sum_{i=1}^{k} ||x_i - f(x_{i-1}, u_i)||^2 + \sum_{(i,j)\in lidar} x_{i,j} - h(x_i, l_i)||^2$$

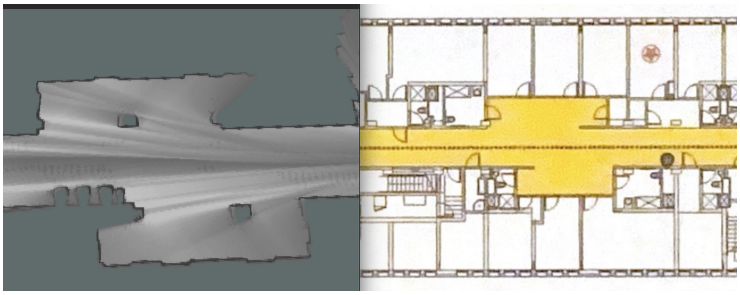

Figure 5: Side by side comparison of a snippet of the SLAM map with the official floor plan of simmons

We used Cartographer ROS, which implements real-time loop closure in 2D with Lidar data. The cartographer is made of a local SLAM optimizer that tries to build multiple submaps where each is locally consistent. On top of that a global SLAM provides the loop closure that matches the submaps. In Figure 5, we can see it mappend the corridor in Simmons more accurate than the actual map. Our map includes the trash bins and the concrete columns on the side of the hallway.

The Cartographer takes the lidar ranges as inputs, crops it according to a minimum and maximum range, and uses a voxel filter to downsample the lidar data similarly to an occupancy grid for computation sake. A second adaptive voxel filter is applied to optimize the voxel sizes. The inertial measurement unit of the racecar is used to get an initial guess for the orientation between scans.

Rather than using the car's dynamics, the local SLAM is extrapolating sensor data to match different scans. In uncertain situations, the correlation can be computed to minimize the object further, but it is more computationally demanding. To save time, only selected scans are considered. If there is not a lot of movement between consecutive scans, it will be dropped. A submap is considered to be complete once a certain number of scans has been received. They are getting saved to truncated signed distance fields which the global SLAM will then use to close the loop.

Global SLAM is treated like a constraint graph where nodes are the submaps, and its edges are pose transformations. To close the loop, the correlations are matched first, and the poses will be computed by minimizing the objective. Figure 6 is an example where we turned around and drove back, thus creating new submaps which are locally consistent but did not match globally. Shortly after, global SLAM kicked in and fixed the relative poses, as seen in Figure 7.

We found that the loop closing step is slower than we want and can cause large drifts. With those drifts, tracking a global path planning problem would be hard. So if we were to use SLAM for localization, we should make sure to exploit the local consistency in the inner SLAM loop via a local planner.

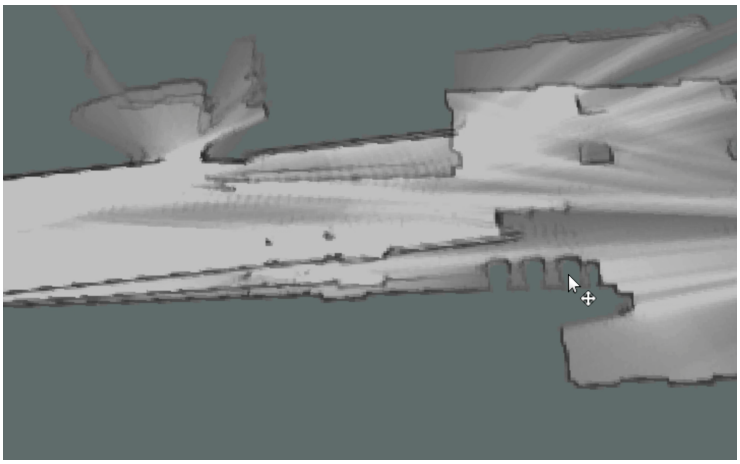

Figure 6: Simmons Floor shortly before loop closure

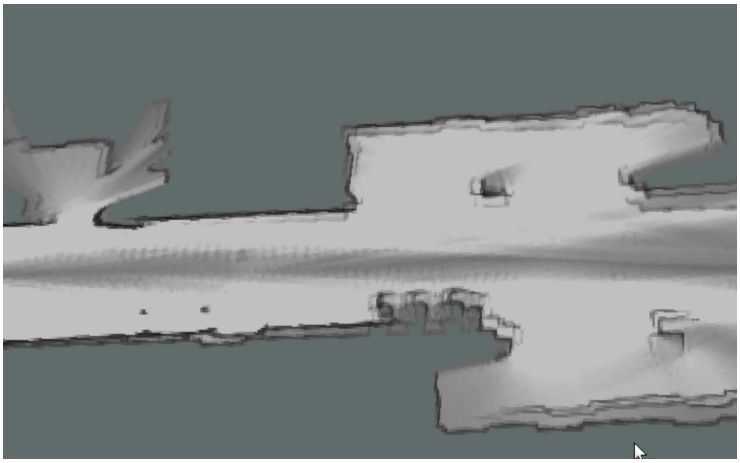

Figure 7: Simmons Floor after global SLAM performed loop closure

5 Conclusion

Ethan Hammons

Through MCL and SLAM our team has successfully implemented two algorithms that allow our robot to localize itself and map the environment. There is some minor tweaking to be done when testing more on the physical car, such as changing the co-variance matrices, but for the most part these implementations

of the algorithms are complete. Our experimental data shows that our MCL implementation is quite successful in localization with an error of only 0.023 meters on average. Having this algorithm as well as a successful implementation of SLAM means that we have flexibility in the future for how we want our robot to find where it is. Now that the robot can localize itself in an environment, the next step is to give it paths in those environments to follow.

6 Lessons Learned

Jinger Chong

Armed with the lethal combination of midterm week and spring break, this lab proved to be the most logistically challenging of them all. We started quite early by delegating tasks to subteams, but quickly encountered roadblocks due to incorrect unit tests. The team generally lacked our usual motivation and enthusiasm because we could barely see any progress day by day. On my end, I felt discouraged and lost on how to make the sensor model work. Admittedly, I could have done more to reach out to my teammates. Although it all worked out in the end, this lab highlighted any fracture points our team already had. We had a lengthy discussion after our lab briefing to clear out all the miscommunication and I believe that has made our team stronger.

Ethan Hammons

This lab was definitely the most challenging when compared to the previous ones. Unfortunately, getting sick during the first week this lab was out did not help with that. Because of this, I did not learn much in the way of technical work. I will say that every lab I get more comfortable with ROS and feel more confident writing more complex nodes. In terms of communication, I learned a lot. The combination of myself getting sick, lots of midterms, and spring break right in the middle of this assignment, resulted in the team struggling in ways we had not before. After lots of miscommunication, tasks not being done on time, and last minute integration, our team has come out of this lab with a much more solid understanding of each other. In the final lab and the final project I believe we will have a more robust team dynamic.

Nina Gerszberg

This lab turned out to be more complex than we anticipated. The amount of time we anticipated this lab needing was a gross underestimate of how much time it actually needed. I wrote part of the motion model and particle filter and feel that I learned a lot from that experience. I also helped get our code working on the physical car after it worked on the simulation. I wound up writing more lines of code for this lab than I did for previous labs. However, after break I had a huge exam in one of my other classes so a large portion of

my work for this lab was completed before and during break or after my exam (which was on Thursday morning). As a team, we struggled to meet some of the deadlines for this lab which hopefully will make us better at budgeting our time in the future. Tensions between teammates were much higher and so we met as a team to discuss what we should do going forwards which I think was an incredibly productive conversation for us to have. I'm eager to see the ways we have improved as a team as a result of this experience in the coming labs.

Juan Rached

The localization problem is the topic I have been most drawn to throughout the semester. I was very curious about probabilistic robotics so I was very happy to spend plenty of time working on this lab. I initially was assigned to work on the motion model and the overall integration of the system, but ended up helping out with the sensor model as well. In the spirit of understanding the program better, and put everything together, I took it upon myself to debug the sensor model. This took almost an entire afternoon, as I had to read over a section I was not assigned to, understand a program I did not write, and then try to find the flaw in it. As a positive, I came out of that experience with a complete understanding of how a particle filter works - what role the motion and sensor models play and how they come together. As a negative, this delayed the deployment of the code on our actual robot as I could have asked a teammate for help and gone through the sensor model code faster. I learned there are benefits and drawbacks from working on all sections of a robotics application, but that when working with a team, it is often better to err on the side of caution and choose the approach that will expedite the deployment of the project.

David von Wrangel

In previous classes, I was exposed to the particle filter and more advanced topics, which is why I decided to hold myself a bit back during the Lab and focus on getting SLAM done. Following installation, scripts were fairly straightforward, so I was done within a day. With midterms in spring break in all the weeks, we had time to prepare for the lab, I noticed that our team was challenged when it came to integration. A few days before submission, I ended up helping out quite a bit since integration was harder than expected. In retrospect, it was good to hold me back, but in the future, I will make sure to still monitor progress to kick in earlier if we drift too much into a rabbit hole. I believe our struggle in integration originated from a logistical hardship since everybodies schedule was changed too rapidly due to upcoming exams and spring break. Thus I expect us to have an easier time for the rest of the semester.