Lab 5 Report: Localization on the Racecar

Team 5

Alvin Banh
Jialin Chen
Nevena Stojkovic
Haris Imamovic

6.4200: Robotics Science and Systems

April 12, 2025

1 Introduction

Author: Nevena Stojkovic, Editors: Alvin Banh, Jialin Chen, Haris Imamovic

In this lab, the goal was to develop a robust localization system for a racecar simulator using particle filters to estimate the vehicle's position in a 2D map. Localization is critical as it allows the vehicle to locate and navigate itself within its environment. The challenge in this lab was to handle uncertainty in both the odometry and light detection and ranging (LIDAR) data for real-world applications.

To achieve localization, we implemented a Monte Carlo Localization (MCL) algorithm that uses particles to represent possible positions of the vehicle. Each particle has a weight that represents the likelihood of it being the true position based on sensor readings. By continuously updating the particles using a motion model (odometry-based) and a sensor model (LIDAR-based), the filter refines the position estimate over time. The motion model accounts for the vehicle's movement, while the sensor model adjusts the particle weights.

The challenge in this lab was integrating noisy sensor and motion data while ensuring that the particles spread correctly and represent the vehicle's true position. We addressed this challenge by incorporating Gaussian noise into the motion model to simulate real-world uncertainties in vehicle movement.

Additionally, the sensor model was designed to handle different types of sensor readings, such as accurate distances and noisy or missed measurements.

This report presents the approach used to implement the motion and sensor models for the particle filter with performance evaluations of our algorithm. The focus is on optimizing the system for real-time performance while maintaining accurate localization even in complex environments with noisy sensor data.

2 Technical Approach

Authors: Alvin Banh, Jialin Chen, Nevena Stojkovic, Haris Imamovic

2.1 Technical Problem and System Overview

The technical problem is implementing the MCL algorithm with real-life sensor and odometry data. We had to address the noise and error accumulation from both proprioceptive and exteroceptive sensors. Our technical approach was fine-tuning noise deviations and particle amounts to see how accurately the racecar's position is in the map compared to the real world.

The pipeline of our localization implementation is as follows in Figure 1:

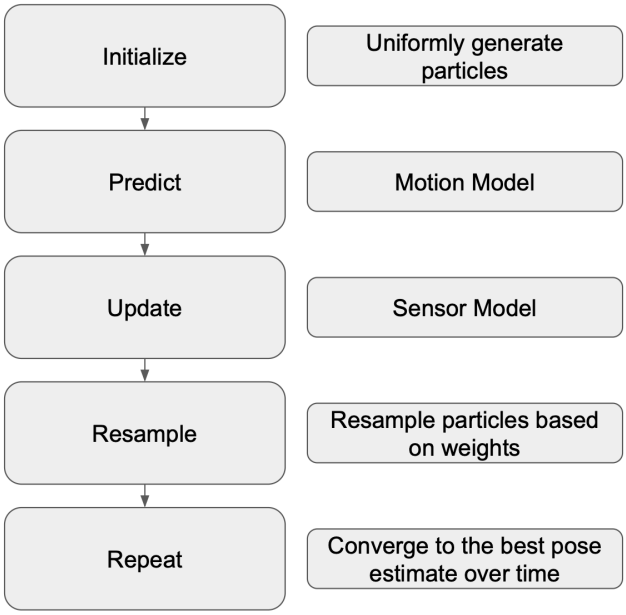


Figure 1: The left column of this figure shows the high level series of actions in implementing MCL. The right side is the method of implementation.

In this remainder of this section, we will explain the implementation for the motion model used to predict poses, sensor model used to update confidence of poses, and combining them together in a particle filter used in MCL.

2.2 Motion Model

The first step in localization is initializing a random set of particles representing potential robot poses, as shown in Figure 2. The motion model updates each particle given odometry data at each time step.

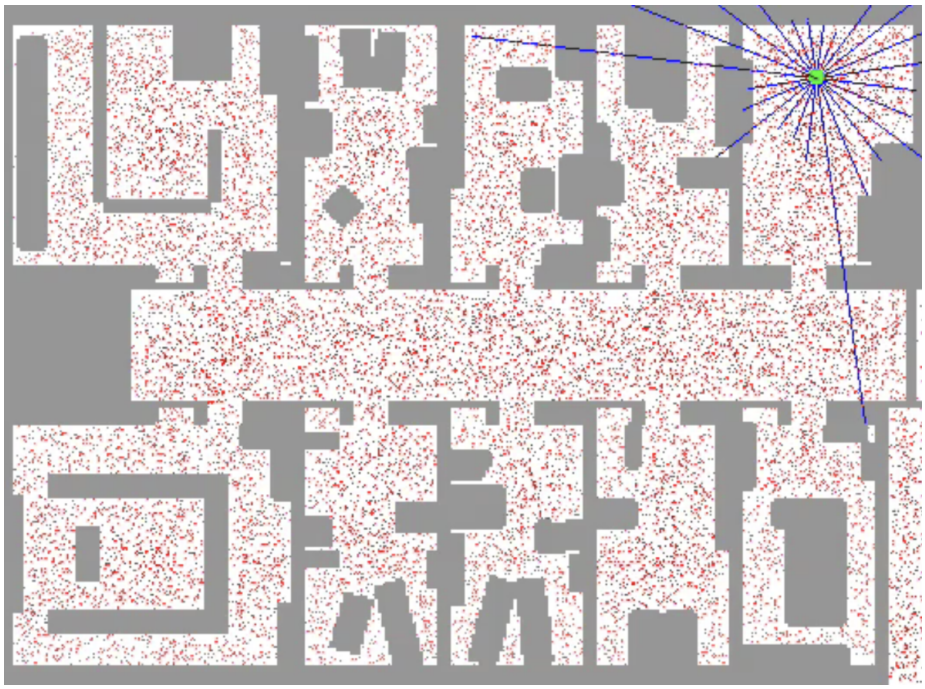


Figure 2: Start by uniformly initializing the particles on the map. Each red particle represents a possible robot pose.

The odometry data is used to update the locations of these particles by this equation:

$$x_k^{(j)} = f(x_{k-1}^{(j)}, u_k, \bar{\epsilon}_u) \tag{1}$$

Where $x_k^{(j)}$ is the pose of particle j that is at time step k which is dependent on a function accounting for its previous pose of particle j $x_{k-1}^{(j)}$, odometry data u_k, and average Gaussian noise $\bar{\epsilon}_u$. This noise accounts for uncertainty in real-time odometry measurements.

The coordinate transformation function that we implement to obtain the pose of particle j is shown below:

$$\begin{bmatrix} \Delta x_{bodyframe} \\ \Delta y_{bodyframe} \end{bmatrix} = R^{-1}\Theta_{k-1} \begin{bmatrix} \Delta x_{worldframe} \\ \Delta y_{worldframe} \end{bmatrix} \tag{2}$$

To transform the change in the robot's position from its world frame to its

body frame, we apply an inverse rotation matrix $R^{-1}\Theta_{k-1}$ where Θ_{k-1} is the orientation of the robot in its previous time step.

We then add Gaussian noise to model uncertainty for our VESC measurements. This noise follows a normal Gaussian distribution, denoted as $N(0, \sigma^2)$, where the mean is zero and the variance is σ^2. We add independent noise terms to the deterministic update.

Deterministic updates would yield:

$$x_{\text{new}} = x + \Delta x, \quad y_{\text{new}} = y + \Delta y, \quad \theta_{\text{new}} = \theta + \Delta\theta \tag{3}$$

Stochastic updates (using Gaussian noise)would yield:

$$x_{\text{new}} = x + \Delta x + \epsilon_x, \quad \epsilon_x \sim N(0, \sigma_x^2), \tag{4}$$

$$y_{\text{new}} = y + \Delta y + \epsilon_y, \quad \epsilon_y \sim N(0, \sigma_y^2), \tag{5}$$

$$\theta_{\text{new}} = \theta + \Delta\theta + \epsilon_\theta, \quad \epsilon_\theta \sim N(0, \sigma_\theta^2). \tag{6}$$

In our implementation, the noise parameters are set as follows:

- $\sigma_x = 0.1$: Standard deviation for the x-axis movement.
- $\sigma_y = 0.05$: Standard deviation for the y-axis movement.
- $\sigma_\theta = 0.1$: Standard deviation for rotational motion.

we use NumPy's np.random.normal() function to get a normal curve centered around 0 with a standard deviation σ_x above to generates noise for each particle's x-coordinate. The same function is performed for the y-coordinate and the orientation θ using their respective standard deviations. This stochastic model is useful for accounting for the imperfection of the sensor data in the robot.

2.3 Sensor Model

Each time we receive sensor data from our 2D LIDAR, we need to update the probability of the pose being each particle from the motion model. We create a 201×201 table of probabilities $P(z|d)$ or the probability that the measured distance z matches the expected distance d. The reasoning for this is to be able cross validate our sensor model data to our motion model to determine which particles are more like to be the pose of the robot. We then resample to get better approximations of the robot's position. The probability table is composed of the weighted sum of 4 different cases.

$$p_{\text{hit}}(z \mid d) = \frac{1}{\sqrt{2\pi\sigma_{\text{hit}}^2}} \exp\left(-\frac{(z-d)^2}{2\sigma_{\text{hit}}^2}\right) \tag{7}$$

$$p_{\text{short}}(z \mid d) = \begin{cases} \frac{2}{d}\left(1 - \frac{z}{d}\right), & \text{if } 0 \le z \le d \text{ and } d \neq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

$$p_{\text{max}}(z \mid d) = \begin{cases} 1, & \text{if } z = z_{\text{max}}, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

$$p_{\text{rand}}(z \mid d) = \begin{cases} \frac{1}{z_{\text{max}}}, & \text{if } 0 \le z \le z_{\text{max}}, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

p_{hit} in Equation (7) resembles the likelihood of a sensor z being an accurate reflection of the distance d. p_{short} in Equation (8) resembles the likelihood of a sensor z being cut short through potential obstacles, ranging from 0 to d. p_{max} in Equation (9) resembles the case that the sensor records at z_{max}. p_{rand} in Equation (10) resembles the case that the sensor records at z_{max}

For each z and d value, each of these probabilities are calculated with an tuned α weight and summed together to populate the probability table. The α_{hit}, α_{short}, α_{max}, and α_{rand} values are 0.74, 0.07, 0.07, and 0.12 respectively. The σ_{hit} value is set to 8. These parameters are set to match the probability of these sensor readings occurring on the LIDAR scans of the robot.

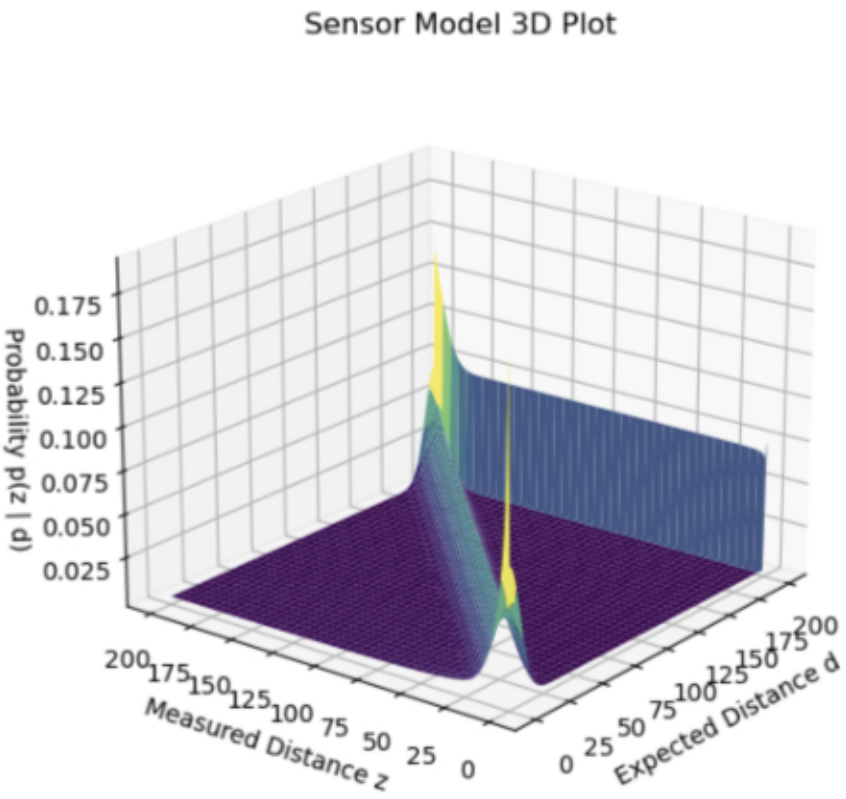


Figure 3: 3 dimension probability plot with measure distance (z), expected distance (d), and probability of these measurements matching $P(z|d)$. The spike in probabilities in the middle are from normalizing the sum of p_{hit}

2.4 Particle Filter/Monte Carlo Localization

The particle filter is implemented to perform MCL based upon updates from the motion and sensor model as the robot moves, shown in Figure 4. It creates pose estimates from a set of particles from the odometry data from the VESC and sensor data from the LIDAR. The purpose of a particle filter is to compute likelihoods of a set of particle and resamples to get better positional estimates. We have added noise to our motion model to account for odometry drift on the real car.

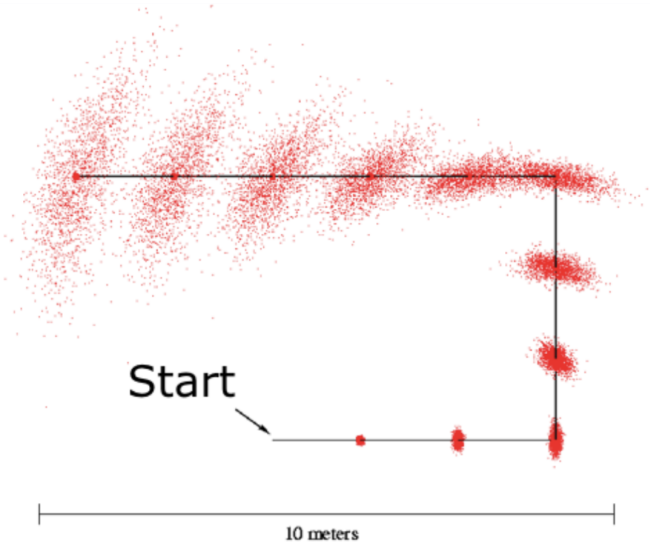


Figure 4: There is better pose estimate for locations closest to the robot as sensor data can more clearly update the weights of those particles.

There are four callback functions: laser callback, odometry callback, pose callback, and timer callback. The laser callback updates the particle weights by LIDAR data. The odometry callback updates the position of the particles. The pose callback reinitializes particles when a new pose is given. The timer callback is used for debugging with ground truth positions.

We then compute the average of the particle pose and publish the odometry message. This process calls upon the previous sensor and motion model and processed in the order outlined in Figure 1.

3 Experimental Evaluation

Authors: Alvin Banh, Haris Imamovic, Editor: Jialin Chen

3.1 Testing Methodology

Performance Metrics

We evaluated performance using:

1. **Simulation Metrics:**
    - **Cross-Track Error:** Deviation of the predicted pose from the actual pose of the robot
2. **Real-life Metrics:**
    - **Mean Error:** Deviation of the predicted parking pose from the actual parking pose of the robot, determined by the distance from a wall perpendicular to the robot

3.2 Particle Filter Parameter Optimization

Motion Model Noise Optimization

The noise added on the motion model impacts the behavior of the predicted particles, and thus the overall performance of localization. Figure 5 shows the relationship between the noise across the odometry parameters (x, y, θ).

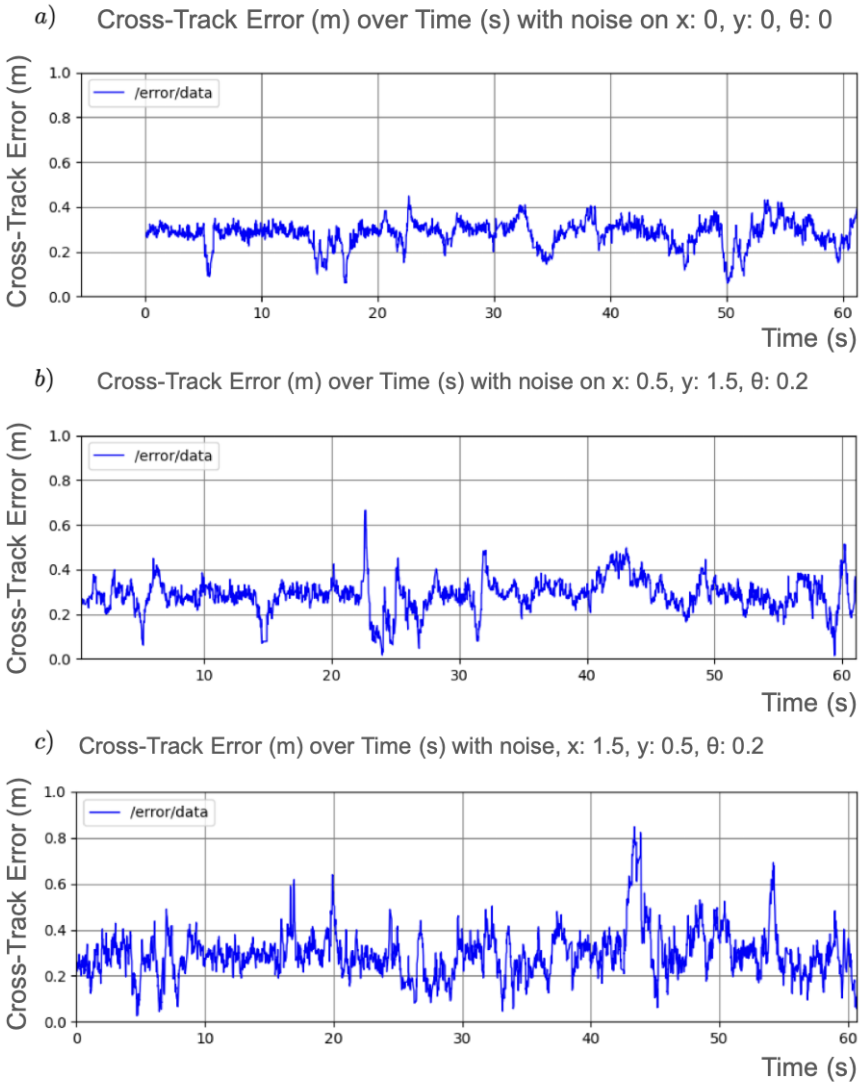


Figure 5: The effect of increasing noise from a) 0 on all x, y, θ to b) 0.5 on x,1.5 on y and, 0.2 on θ, as well as c) 1.5 on x,0.5 on y and, 0.2 on θ, demonstrating a consistent performance in the simulation with small differences in maximum error due to increasing noise while turning

Particle Optimization

Another optimization evaluated was the amount of particles used. The tradeoffs

to using more or less particles is the amount of resolution and the costs of computational load. High computational load would be a problem due to error accumulation from latency issues and speed of determining an accurate pose estimate, which is important for more complex navigation. Table 1 summarizes these tradeoffs.

Table I: Number of Particles on Performance

Particles	Observation:
50	Low latency, lower accuracy
75	Medium latency, medium accuracy
100	High latency, high accuracy

We settled on 50 particles as our algorithm was robust enough to handle lower-resolution localization while maintaining accuracy.

3.3 Real-life Performance Assessment

After determining the robustness of the localization in simulation, we tested how well the robot performs in the Stata basement. We conducted both qualitative and quantitative tests to determine if the robot can accurately draw its trajectory around the Stata basement as shown in Figure 6, and how accurately it displays its position relative to the map using the setup shown in Figure 7 determined by the mean error between its measured distance in Rviz and in reality.

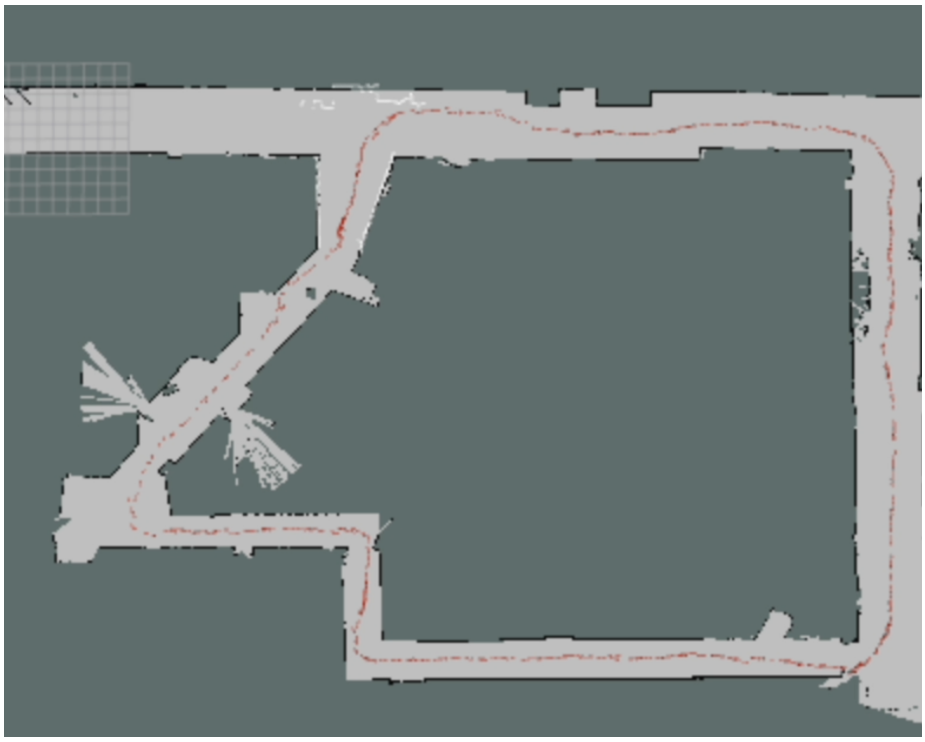
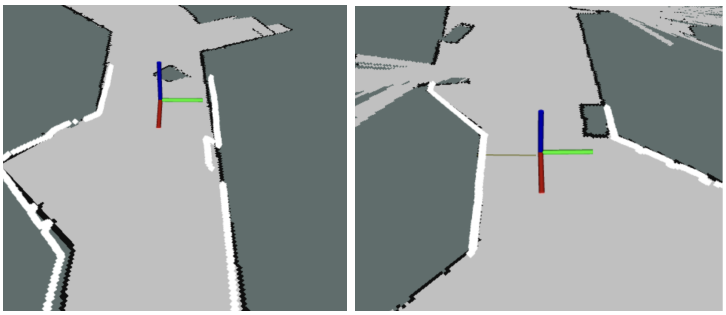


Figure 6: The robot can successfully trace its predicted path in Rviz that aligns with its actual movement through the Stata center basement.

a)



b)

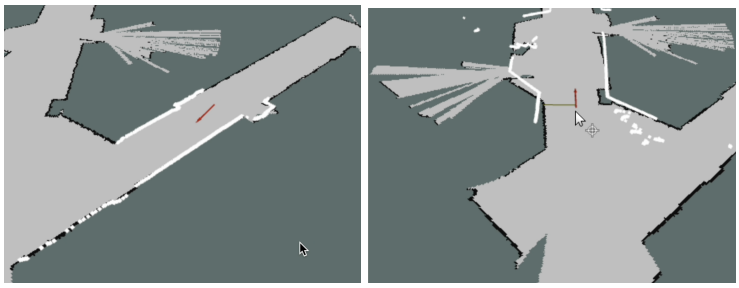


Figure 7: The error can be computed by using the measure tool in Rviz, represented by the yellow line between the wall and the axes representing the predicted end position (right). The measured distance in Rviz is compared with the measured distance in the basement and then used to calculate the mean error, both for a) driving in a straight line for 15m and b) taking a turn for 7m from a start pose (left)

The error in the predicted position was calculated across 3 runs for both the straight driving and turning as shown in Table II, showing a consistent and satisfactory performance with different magnitudes of the change in the odometry due to turning.

Table II: Mean errors across different path shapes

Path shape	Mean error (cm):
Straight	8.0
Turn	9.3

The video of the full basement loop can be found here or at the URL: `https://youtu.be/7KwP99XGfAk?si=eRJbWDbUIBhba4Pe`.

4 Conclusion

Author: Jialin Chen, Editor: Haris Imamovic

Our goal was to design a localization algorithm to determine the robot's pose in the real world on a 2D map. We used Monte Carlo Localization by implementing the motion and sensor model in a particle filter. This particle filter predicts and updates particles that represent beliefs of robot poses and ultimately converges to a location representing the highest likelihood of robot pose. By first making sure that our localization works in simulation, we then switched to the real racecar and fine tuned the parameters of x, y, and theta noises 0.1, 0.05, and 0.1 respectively with number of particles to be 50.

Our implementation initially faced challenges in not distributing different noise and not publishing points often enough when we received odometry and LIDAR data. This would work in simulation where odometry data and LIDAR scans are perfect, but it did not work on the real car where the odometry drifts and LIDAR scans may be noisy. Therefore, not everything in the simulation can translate to real implementation, as we had to do real-life testing and parameter tuning to find new insights not seen in simulation.

Looking ahead, we plan to use localization in the following lab for path planning by combining it with our Pure Pursuit algorithm for following a trajectory. To improve the performance of our localization on the real car, we can keep tuning parameters to reach higher accuracy. For instance, we can modify the α values in the sensor model to see whether it will improve localization. Finally, improving the accuracy of our localization will propagate better performance in future labs as well.

5 Lessons Learned

Alvin Banh

This lab was very technical by the nature of robot localization so I learned a lot more about how a robot can use odometry data and LIDAR scans to be able to cross-validate its location. A large part of the issues came from implementing a solution that would work on the physical racecar. Many confounding factors made it difficult to pinpoint one specific bug in the code due to many problems

arising at the same time. We were stuck on fixing issues with poses causing the racecar model in RVIZ to be completely erratic to static to moving in the circle. The fixes that worked included updating different Gaussian noise to each particle. I learned that how we come to these fixes were ablation tests and seeing the behavior of our localization algorithm if the motion or sensor model was turned off. This is significant for being able to be on the same page of issues and seeing where fixes can be made. In terms of collaboration, I believe we handled meeting times well and were flexible to new ideas. We handled communicating issues and fixes well so we are all on the same page to solve the next problem.

Jialin Chen

This lab was math heavy and I was able to better understand how the motion model and sensor model work together in the particle filter when implementing it. It was also interesting to compare how MCL worked in simulation versus in real life because what looked perfect in simulation didn't work well in real life. For instance, the simulation assumes perfect odometry data while in real life there is drift over time, also known as dead reckoning. This is also why incorporating noise in the motion model is necessary because we want to be able to adjust for situations where data is always perfect and correct it. It was also helpful to go over the math together as a team so that if we looked over one part, then another person can point it out. Collaboration was a important part of this lab and was really successful because we would build the foundation for the logic in code and debug it together. We are also good in communicating what times work best for meetings, what was accomplished, what errors are occurring, and solutions to bugs we run into.

Nevena Stojkovic

Throughout this lab, I gained valuable insights both about teamwork and robotics. On the teamwork side, I learned the importance of clear communication and task distribution. Initially, it was challenging to align the team's work and ensure that everyone was on the same page, but through regular discussions and careful planning, we were able to divide tasks efficiently and collaborate effectively. I also learned that troubleshooting and problem-solving are much easier when team members bring different perspectives, and that patience and flexibility are crucial when unexpected issues arise.
In terms of robotics, this lab deepened my understanding of how complex real-world data can affect localization algorithms. I learned that developing an accurate motion model requires not just mathematical equations but also an understanding of the noise in real-world sensor and control data. Simulating this noise in the particle filter was a key lesson in modeling uncertainty in

robotics. Additionally, working with sensor fusion and integrating LIDAR data with odometry data helped me realize the challenges in ensuring that all sensor information is correctly combined to improve localization accuracy. Finally, optimizing the particle filter to run in real-time and testing it in a simulated environment taught me the importance of efficiency in algorithms, especially when working with real-time systems.

Haris Imamovic

The math in this lab was a great overview of how one can use probability to model real-world behavior, as the different alpha coefficients and reasons why they are used are quite intuitive and seemingly make up a comprehensive model. It was slightly challenging to understand the different technical components of the sensor model implementation since the formatting and downsampling required careful indexing and a thorough understanding of the function parameters. Having a new teammate was a great addition to our team dynamic, as we gained another fresh perspective, and were quick to adapt to a change in flow. Learning how to reinstate team values and make sure everyone is on board with them was a great experience, as I am sure it will be applicable to any teamwork I encounter in the future.

References

[1] MIT RSS Teaching Staff, "Monte Carlo Localization (MCL)," GitHub, 2025. [Online]. Available: `https://github.com/mit-rss/localization`. [Accessed: Apr. 11, 2025].

[2] MIT RSS Teaching Staff, "Lecture 12: Monte Carlo Localization," Massachusetts Institute of Technology, 2025. [Unpublished lecture notes].