# Lab 5 Report: Monte Carlo Localization Using Particle Filter

Team 15

Bilal Asmatullah
Jing Cao
Panagiotis Liampas
Megan Tseng
Michelle Wang

Robotics Science and Systems

April 12, 2025

## 1 Introduction

**Author(s): Michelle Wang**
In an autonomous system, localization, or knowledge of where the robot is in the world, is critical to informing downstream decisions in applications like navigation of complex environments, exploration of new locations, and multiagent systems. In a wheeled robot, the simplest localization method is achieved through integration of odometry measurements (typically linear and angular velocities) from wheel encoders. However, this method suffers from drift as errors in the sensor reading accumulate over time. To correct for drift, many systems utilize additional tools like GPS or beacons (RFID, ArUco tags, etc.) to provide highly reliable localization information in a global frame. Unfortunately, these methods are not useful in GPS-denied environments or in areas where beacons cannot be installed.

In this project, we investigate using a particle filter to accurately estimate robot pose from noisy sensor data in a pre-mapped environment. We first implement a motion model that updates the estimated poses of the particles using noised odometry readings and rigid body transformations. We then use a LiDAR sensor model to update the confidence in each of the particles. To ensure we have both a diverse spread of particles as well as a reasonable number of high confidence particles, we resample our particles at each time the sensor model updates. Finally, we take an average of all of our particles to estimate the robot pose.

Using our particle filter, we achieve accurate localization using the described particle filter in simulation and in the real world. Additionally, we show that our motion model is robust to odometry noise and investigate how to best model such uncertainty. We also investigate how different resampling methods affect particle filter performance.

# 2 Technical Approach

## 2.1 Motion Model

**Author(s): Panagiotis Liampas**
A key component of a particle filter, is keeping track of a set of particles that represent the robot's current estimate of where it is, with respect to a fixed reference frame.

A naive way to estimate the robot's pose is to iteratively compute the displacement of the robot using the linear and angular velocities measured by an Inertial Measurement Unit (IMU) or wheel/motor encoders attached to the robot, as follows: $(dx, dy, d\theta) = (\dot{x}dt, \dot{y}dt, \omega dt)$, and apply them as rigid body transformations:

$$T_{r_k}^W = T_{r_{k-1}}^W T_{r_k}^{r_{k-1}}, \text{ where}$$

$$T_{r_i}^W = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & x_i \\ \sin(\theta_i) & \cos(\theta_i) & y_i \\ 0 & 0 & 1 \end{bmatrix}, T_{r_k}^{r_{k-1}} = \begin{bmatrix} \cos(d\theta) & -\sin(d\theta) & dx \\ \sin(d\theta) & \cos(d\theta) & dy \\ 0 & 0 & 1 \end{bmatrix}$$

A result of this deterministic computation is shown in Figure 1.

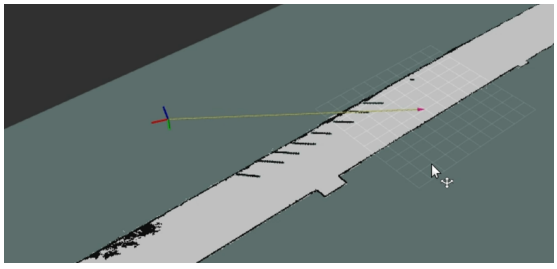


Figure 1: The deterministically computed robot pose estimate is significantly inaccurate in the real world.

However, due to odometry measurement noise, non-infinitesimal timesteps, and error-prone initial pose estimate, there is a large error in the resulting pose estimate after a few time-steps, and it accumulates over time, with no way to correct it using LiDAR scans.

This is where the particles play a key role: instead of keeping track of a single pose estimate over time, we store a set of particles for each of which we independently sample noise variables $\epsilon_x \sim \mathcal{N}(0, \sigma_x^2), \epsilon_y \sim \mathcal{N}(0, \sigma_y^2), \epsilon_\theta \sim \mathcal{N}(0, \sigma_\theta^2)$, or uniformly around 0 for each component, as shown in Figure 2c, to compute the following transformation:

$$T_{r_k,noisy}^{r_{k-1}} = \begin{bmatrix} \cos(d\theta + \epsilon_\theta) & -\sin(d\theta + \epsilon_\theta) & dx + \epsilon_\theta \\ \sin(d\theta + \epsilon_\theta) & \cos(d\theta + \epsilon_\theta) & dy + \epsilon_\theta \\ 0 & 0 & 1 \end{bmatrix}$$

To construct the initial set of particles (before receiving any odometry measurements), we sample $m$ particles around an estimated pose: $x_{init} \sim \mathcal{N}(0, \sigma_{x,init}^2), y_{init} \sim \mathcal{N}(0, \sigma_{y,init}^2), \theta_{init} \sim \mathcal{N}(0, \sigma_{\theta,init}^2)$.

That way, we capture both the uncertainty in the initial pose estimate but also the noise of the odometry data. Varying the variance of the noise variables leads to different particle distributions over time, as shown in Figure 2. These variables are tuned based on the accuracy of the robot sensors and the characteristics of the environment the robot is navigating.

(a) Gaussian noise with moderate $\sigma_x$, small $\sigma_\theta$

(b) Gaussian noise with small $\sigma_x$, small $\sigma_\theta$

(c) Uniform noise

Figure 2: Gaussian noise motion models concentrate the particles closer to the mean while uniform noise ones spread them out equally, achieving greater variance with fewer particles.

## 2.2 Sensor Model

**Author(s): Megan Tseng**

In order to evaluate the likelihood of each of $N$ particles being an accurate estimate of the robot's pose, we implement a sensor model that compares the exteroceptive measurement to the map. For each particle, we perform ray casting to produce a set of LiDAR beams representing the robot's view of the map at the hypothesized position. As a result, we obtain a stack of $N$ LiDAR messages that contain measurements $d^{(i)}$.

At a timestep $k$, the robot receives a LiDAR message containing ranges $z_k^{(i)}$. Given a particle position $x_k$ in map $m$, we evaluate the probabilities of:
(1) Detecting a known obstacle,

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

where $\sigma$ is an intrinsic property of the sensor.
(2) A measurement falling short due to an unmapped obstacle or defects in the robot's field of view,

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

(3) A missed measurement,

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

(4) And a completely random measurement.

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

Combined, the sensor model calculates the probabilities of each LiDAR range:

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

where $\alpha_{hit}$, $\alpha_{short}$, $\alpha_{max}$, $\alpha_{rand}$ are tunable parameters
A particle's likelihood is the joint probability of all LiDAR ranges given the particle's hypothesized position.

As shown in Figure 3, these four cases are characterized respectively by (1) a Gaussian distribution about the "true" distance, (2) a downward sloping line, (3) a spike at maximum range of measurement, and (4) a small uniform value.

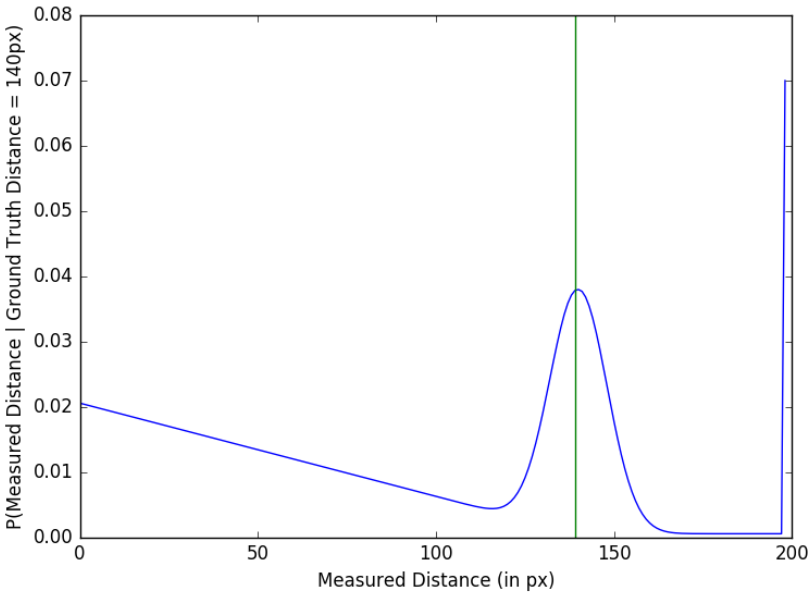



Figure 3: **Probability distribution of a LiDAR measurement.** The distance corresponding to peak probability is close to the green marker representing the ground truth distance (140 pixels) of an obstacle.

In order to reduce runtime when the car is moving quickly, we discretize inputs and precompute a lookup table. For each casted ray $d$ and LiDAR range $z$, we scale the distance to a pixel value using the map resolution. Upon initialization, the sensor model stores the measurement probabilities $p(z_k^{(i)}|x_k, m)$ in a 201x201 table where LiDAR range is represented by row and ray casting by column. As a result, our sensor model can efficiently look up the probability of any measurement up to 200 "pixels" in distance, and clips anything farther as a missed measurement.

## 2.3 Resampling

**Author(s): Michelle Wang**

Particle filters can suffer from degeneracy when there are few particles with high confidence and many particles with very low confidence–this can result in a very inaccurate overall pose estimate. However, having only high confidence particles results in little diversity. To balance these issues, we use resampling strategies to ensure we have a diverse spread with a reasonable amount of high confidence particles. We use the following 4 strategies on N particles with probabilities $p$ and evaluate their performance in simulation.

1. **Random choice sampling**: Particle $i$ has probability $p_i$ (from the sensor model) of being sampled with replacement, so higher probabilities are more likely to be selected. All particle probabilities can be raised to a power to saturate the distribution towards 1 (power $< 1$) or 0 (power $> 1$).

2. **Residual sampling**: We calculate scaled probabilities as $N \cdot p$. For each particle, we sample it $\lfloor N \cdot p_i \rfloor$ times, which fills up part of our collection of resampled particles with high probability particles. For the remaining particles, we use random choice sampling with residual probabilities $N \cdot p - \lfloor N \cdot p \rfloor$, which encourages a diverse spread.

3. **Stratified sampling**: We subdivide the cumulative distribution of the particles into $\frac{1}{N}$ evenly spaced intervals. In each interval, we sample a randomly generated position which corresponds to a particle that occupies that location in the cumulative distribution. This way, we ensure a diverse spread of particles while making it more likely to sample high probability particles.

4. **Systematic sampling**: This is a similar subdivision approach like stratified sampling, but we only generate one random position offset, and the rest of the particles are sampled at positions $\frac{1}{N}$ apart.

## 2.4 Particle Filter

**Author(s): Jing Cao**

The particle filter is the core probabilistic localization framework that integrates the motion model, sensor model, and resampling to maintain a belief distribution over the robot's pose. At a high level, the particle filter consists of five main steps shown in Figure 4.

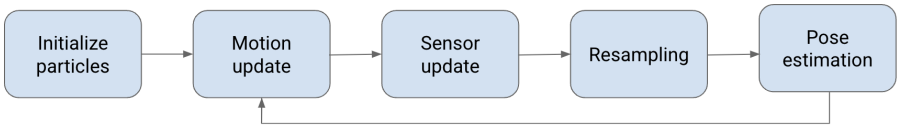



Figure 4: Overview of the particle filter algorithm.

**1. Initialization:** We first initialize $N$ particles by sampling from a Gaussian distribution centered around the robot's assumed starting location $(x_{init}, y_{init}, \theta_{init})$ with specified standard deviations $(\sigma_x, \sigma_y, \sigma_\theta)$. This captures the initial uncertainty in the robot's pose estimate.

**2. Motion Update:** Each particle is propagated forward based on the robot's measured control inputs using the motion model described in Section 2.1 and

motion equations

$$[dx, dy, d\theta] = [v_x \cdot dt, v_y \cdot dt, v_\theta \cdot dt].$$

Noise is added to account for odometry drift, causing the particle cloud to spread and reflect uncertainty in movement.

**3. Sensor Update:** Once particles have moved, we evaluate how likely each particle's pose is, given the latest LiDAR scan. Using our precomputed likelihoods described in Section 2.2, each particle is assigned a weight proportional to the probability of observing the LiDAR scan from that pose,resulting in a distribution where high-weight particles are more consistent with sensor data.

**4. Resampling:** We then resample from the particle set based on the computed weights, using random choice resampling with a power of 1. This favors high-likelihood particles while maintaining diversity.

**5. Pose Estimation:** To estimate the robot's current pose, we compute the weighted average of all particles. This final estimate is robust to outliers and can track the robot's true pose over time, even in the presence of non-Gaussian noise and sensor anomalies.

# 3 Experimental Evaluation

## 3.1 Particle Filter Working in Simulation

**Author(s): Jing Cao**
Initially, we tested the particle filter's performance in simulation. We collected ground truth pose data from the simulator and compared it with the estimated robot pose over time, computing the mean absolute error in the $x$ and $y$ positions and the orientation $\theta$.

In the noise-free simulation, odometry and LiDAR measurements are ideal, allowing the particle filter to maintain a tightly clustered and highly confident particle distribution. As shown in Figure 5, the pose estimation error remains consistently low across all three dimensions:

- $x$ error: $0.2642\,\mathrm{m}$
- $y$ error: $0.0522\,\mathrm{m}$
- $\theta$ error: $0.0127\,\mathrm{rad}$

This confirms that the particle filter performs well when sensor readings and motion data are accurate, and the filter converges quickly and stably to the true pose.

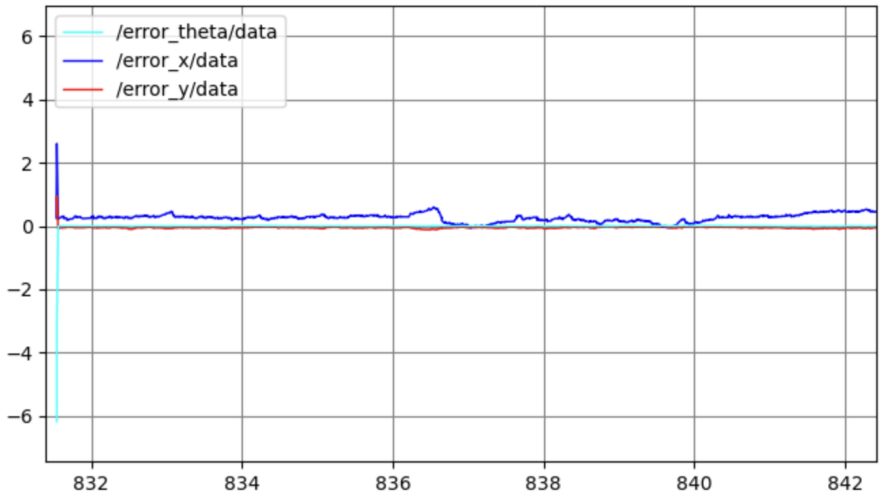



Figure 5: Particle filter working in simulation with no motion or sensor noise. Errors in pose remain very low and stable throughout the trajectory.

## 3.2 Noisy Odometry/Motion Model

**Author(s): Bilal**

Although our algorithm performed well in simulation, real-world odometry is subject to various sources of noise that are absent in an idealized simulator. In simulation, we can measure ground-truth robot poses and compare them to our estimated poses to quantify errors in $x$, $y$, and $\theta$. To gain confidence that our approach would remain accurate once transferred to real hardware, we introduced artificial noise into the simulator's odometry data and measured how our pose-estimation errors changed.

To ensure consistency across different noise levels, we recorded a single ROS bag of the robot's motion in simulation. We then replayed this bag four times, each time injecting different noise profiles into the odometry:

- **No Noise (Baseline)**
- **Gaussian Noise (1 m/s, 0.5 rad/s)**
- **Gaussian Noise (2 m/s, 1 rad/s)**
- **Gaussian + Uniform Noise (1 m/s, 0.5 rad/s + additional uniform noise 1 m/s, 0.5 rad/s)**

We recorded the differences between the ground-truth pose (available from the simulator) and our estimated pose for each condition as shown in Figure 6.

- **From Baseline to 1 m/s Gaussian Noise**

  - $x$-error change: 0%
  - $y$-error change: $\approx 4.3\%$
  - $\theta$-error change: $\approx 16.7\%$

- **From Baseline to 2 m/s Gaussian Noise**
  - $x$-error change: $\approx 14.3\%$
  - $y$-error change: $\approx 4.3\%$
  - $\theta$-error change: $\approx 33.3\%$

- **From Baseline to Gaussian + Uniform Noise (1 m/s, 0.5 rad/s)**
  - $x$-error change: $\approx 23.8\%$
  - $y$-error change: $\approx 17.4\%$
  - $\theta$-error change: $\approx 83.3\%$

Although these percentages may look large in some cases (especially for $\theta$), the absolute differences in $x$ and $y$ remain modest. The overall error increases provide a reasonable worst-case estimate of how our method might behave on physical hardware, giving us confidence that our algorithm is robust to moderate levels of odometry noise.

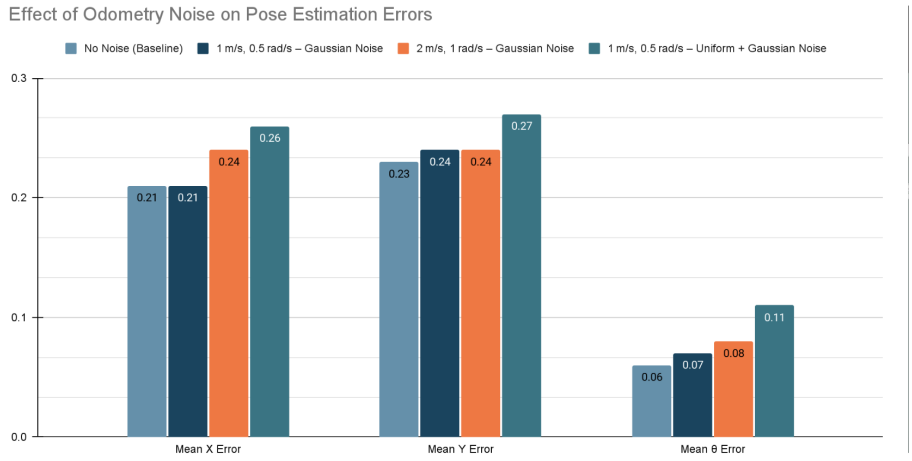



Figure 6: Figure summarizes the mean errors in $x$, $y$, and $\theta$. As expected, the baseline (no added noise) yields the smallest errors. When noise is added, errors do increase, but generally remain within acceptable bounds.

## 3.3 Resampling Study

**Author(s): Bilal**

To evaluate the robustness of different resampling strategies under extreme conditions, we injected Gaussian noise with $\sigma = 1.5$ m in both $x$ and $y$ directions and 0.4 rad in $\theta$ at every motion model update. We compared six resampling methods using mean absolute errors in $x$, $y$, and $\theta$ as our evaluation metrics. Our tests showed that the `random.choice` method with particle probabilities raised to 1 (before resampling) consistently achieved the lowest mean errors, making it our preferred choice for the final implementation. Notably, systematic resampling performed poorly and lost the robot. The complete results are summarized below in Table 1.

| **Method** | **Mean X Error** | **Mean Y Error** | **Mean $\theta$ Error** |
|---|---|---|---|
| random.choice, power = 1/3 | 0.70 | 0.21 | 0.05 |
| **random.choice, power = 1** | **0.64** | **0.21** | **0.04** |
| random.choice, power = 2 | 0.68 | 0.21 | 0.05 |
| Stratified | 0.91 | 0.22 | 0.05 |
| Residual | 0.90 | 0.22 | 0.05 |
| Systematic (lost robot) | 14.0 | 0.98 | 0.13 |

Table 1: Mean error performance of resampling strategies under high noise

## 3.4 LiDAR-Map Alignment for Real-World Evaluation

**Author(s): Panagiotis Liampas**

After we verified the accuracy of the particle filter in simulation and made it work in the real world after minor modifications and tuning, we needed to evaluate its performance in those high-variance conditions where the sensor (especially odometry) noise was much larger.

As shown in Figure 7, we transform the LiDAR scan by our average particle pose and visually evaluate its alignment with the map. We calculate the average negative log probability of the LiDAR rays detecting an object in the correct position based on the map:

$$\frac{-\log(prob_{total})}{\text{num beams}} = \frac{-\log(\prod prob_{beam})}{\text{num beams}} = \frac{\sum -\log(prob_{beam})}{\text{num beams}} = \overline{-\log(prob_{beam})}.$$

where a lower value indicates that our system works well in the real world.

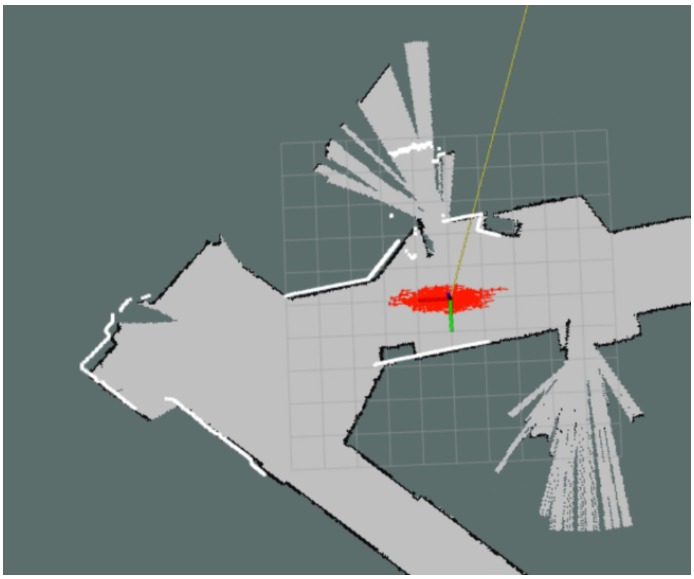
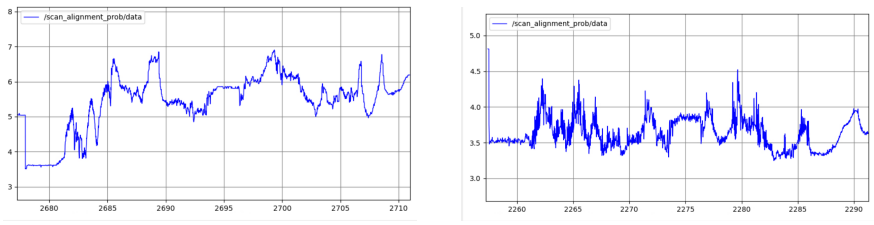


Figure 7: The LiDAR scan is seemingly well aligned with the map walls, indicating that the position estimate of the robot is relatively accurate

This is closely related to the average distance between the detected position and the actual position of a point intersected by the LiDAR beam, since the sensor model is really similar to a Gaussian distribution.

Figure 8 shows that, quantitatively, our system works better in the real world, especially in terms of handling the set of particles and updating it, using both odometry and LiDAR, to align the 2D scan with the walls that are on the map, getting closer to an accurate estimate of its position.




(a) Deterministic model using odometry (b) Particle Filter

Figure 8: A comparison of the LiDAR-map alignment between the deterministic motion model and the noisy one, in the real world, using the metric described above.

# 4 Conclusion

**Author(s): Megan**
In this lab, we successfully designed a particle filter to perform probabilistic localization on our robot. In simulation and real world testing, we were able to show the advantages of our particle filter over "dead reckoning" with only the robot odometry.

We experimentally tuned a Gaussian noise distribution for our motion model that allowed us to evaluate a range of positions as opposed to a single deterministic value, which was subject to factors like time lag and drivetrain drift. By experimenting with noise models and robot settings in simulation, we were able to make our particle filter robust to odometry noise and quantify the effect of dramatic real-world imperfections on our motion model predictions. Additionally, we investigated four resampling methods for updating the particle distribution after sensor model evaluation. We selected random choice resampling, achieving small average x-direction, y-direction, and angular position errors of 0.64 m, 0.21 m, and 0.04 rad in the presence of large velocity noise. To quantify the success of our pose estimate over an entire run, we showed that our particle filter produced much better LiDAR-to-map alignment than the deterministic model.

In the future, we aim to further investigate our sensor model in the real world. Testing with more complex environments or a range of speeds may allow us to systematically tune our sensor model parameters for various situations. Additionally, optimizing our code for higher drive speeds and larger numbers of particles will further improve localization precision. Although we did a significant amount of qualitative analysis on our robot in the real world, looking ahead, we would develop better methods for utilizing ground truth position.

# 5 Lessons Learned

**Michelle Wang**: For technical skills, this lab taught me how to implement a particle filter from end-to-end. I had read about particle filters before, but had never had an opportunity to try it out, since having access to a mapped location is pretty rare. This lab also lined up really well with my probability class's unit on Bayesian updating, so it was very enlightening to see how the ideas in the particle filter connected to the pure math I was learning in another class–this also enabled me to take a stab at the extra credit. Generally, when I work with complex algorithms in robotics, I feel as if the math is over my head, but I really appreciated understanding the particle filter at a theoretical level. In the future, I hope to be able to achieve this kind of understanding more often. Also maybe we're doing something wrong, but this lab once again reminded me how non-user-friendly rosbags are: going forward, validate each rosbag before we finish data collection! (...or simply don't use them...?)

This lab taught me the importance of trust, communication, and being responsible for taking on a fair share of work. I knew I would be out of town the weekend before the briefing, so I communicated this with my team 2 weeks before spring break. I believe this saved everyone from being blindsided, and in the future, I will make sure to give well in advance notice whenever possible. In an effort to keep the division of labor even, I implemented a functional simulated particle filter by 3/30. This allowed us to have the code working on the real racecar before I left town, which gave the team enough time to collect real world data. I believe taking on this responsibility helped spread the work out evenly and allowed me to take the trip without worrying as much about the project. Finally, throughout it all, I kept in communication and contributed to discussions as needed, so I was still in the loop when I returned to campus. I was happy to see that everything pretty much worked out, so I hope to continue to apply these principles of fairness, communication, and trust in future projects.

**Megan Tseng**: In this lab, I gained an understanding of the challenge of robot localization and different methods of approaching it. Without prior knowledge, it didn't seem like a particularly difficult problem at first, especially given that we were able to use a map of Stata basement. However, I later understood how imperfections in the real world contributed to this localization approach; modeling motion as an evolving collection of particles allowed the robot to incorporate external measurements and account for uncertainty. Additionally, learning more about the different resampling methods we investigated was very insightful because I don't have much prior knowledge in probability. Being able to see the benefits of how some resampling methods selected low-weighted particles to preserve in the particle distribution gave me a better understanding of the advantages of probabilistic localization.

Data collection was particularly challenging for this lab because we had trouble finding a way to evaluate our particle filter in real life. While our particle filter was very successful in simulation, transferring to real world required a lot of tuning that we didn't have a quantitative basis for. As a result, I learned a lot about statistical analysis from my teammates, who were really creative with showing how confident and accurate our robot pose estimate was. To do this, we simulated our robot localization with large odometry noise, similar to what it might experience in real world, and also evaluated LiDAR-to-map alignment at every timestep of a map. Our collection of real-world data (which resulted in some difficult-to-play rosbags) also taught me the importance of keeping things simple and verifying that we have the data we need.

From a teamwork standpoint, I learned the importance of front-loading the planning and being able to delegate work early. We had some unexpected hardware issues come up during the crucial part of our weekend, which three of our team members had to handle since two of us were out of town (myself included). We ended up having to take a lot of data at the end of Sunday, which was a lot

for everyone involved. I think we definitely could have organized better or delegated in-sim work to certain members while others worked on debugging the hardware. However, we pulled together at the end really well and I learned a lot from how everyone contributed to each other's parts. Even though we split up the different sections, we each provided input on other sections that made our analysis more meaningful.

**Jing Cao**:
In this lab, I gained a deeper understanding of the components behind particle filters and how localization works. Although I've used localization in previous projects, implementing it myself gave me a new appreciation for the abstraction layers and how noisy data can be managed to build accurate environmental models.

The most challenging part was data collection and evaluation. Because localization depends on an initial pose estimate, and that estimate was hard to pinpoint in the simulated map, we had to explore alternative ways to evaluate our filter's accuracy.

Despite the challenges, working with my teammates made the process more manageable. We split up tasks and collaborated closely on debugging and improving our localization model.

One major hurdle was dealing with hardware issues during data collection when office hours weren't available. To prevent this in future labs, we plan to collect data earlier so we have time to troubleshoot with TA support if needed.

**Bilal Asmatullah**:

This was one of the most fun and intuitive labs. I got a nice overview of theoretical basis for particle filters from lectures and practice of their implementation on race car. The lab was also effective at introducing real world challenges such as noise and was an excellent demonstration of how a well distributed noise combined with averaging can cut through real world noisy data. The lab was also unique in the sense that it made clear the difference in computing power of our personal computers vs the limited computing power available on race car. It was pretty cool to see the difference in frequency of published topics due to above differences. Aside from meeting the objectives of the lab itself, it was a great experience working with my teammates. Everyone brings their own unique strengths and insights to the table. Some are good at planning, some are good at coding, some good in analysis, some good in coming up with cool ideas but it is great that we come together nicely as a team to bring out the best combination. Moreover, during moments when things get challenging and quite some work is piled up at the very last day, it is good that even in heated moments and middle of disagreements everybody gets a chance to share their

view, as well as respect each other's opinions. I think times like these, while maybe frustrating for the time being, strengthen and test the team spirit for the long run.

**Panagiotis Liampas**: I learned a lot about particle filters and how they work, both by understanding the theoretical foundations behind them, but also by experimenting with certain aspects of them (mostly the motion model) and making it work in the racecar by fine-tuning its parameters. That way I learned a lot about the various issues that might occur in localization, and specifically particle filters, like large/small amounts of noise (and maybe using different types of it), places that look similar in the LiDAR scan, odometry noise that plays a big role in the robot's predictions (and how to deal with it), and an insufficient number of particles, all of which can compromise a particle filter's performance.

I also got familiar with more ways to gather and analyze data on the performance of the robot in real-world environments, and how to quantify intuitive observations even when it's not exactly obvious, and when gathering ground-truth data takes a lot of effort. Coming up with ideas like that can be really useful in later work (in the class or elsewhere) as well since getting data to compare the robot's performance with is not always easy, so workarounds might be more practical and still capture similar information.