Lab 5 Report: Localization

Team 4

Rohan Wagh
Oliver Rayner
Gabriel Jimenez
Kairo Morton

6.4200/16.405: Robotics Science and Systems

April 15, 2023

1 Introduction

In lab 5, we worked on solving robot localization in a known environment. Robot localization is a vital aspect of autonomous robotics and involves determining the precise location and orientation or pose of a robot in its surrounding environment. Being able to localize a robot's pose is essential for performing a wide range of tasks autonomously, including navigation, exploration, mapping, and object manipulation. Without accurate localization, a robot would be unable to determine its position relative to its surroundings and would be unable to make intelligent decisions about its movements.

Localization was achieved by taking the robots existing position and predicting possible next locations using odometry from the car. These possible next locations were then filtered using the input LIDAR readings and comparing the LIDAR readings to the possible locations' positions on a known map of the environment. This allowed the localization algorithm to find the most likely current new position as the position that best fit the odometry and scan inputs. This lab was divided into three modules: the robot model, the sensor model, and the particle filter which all combine to form a Monte Carlo localization algorithm.

2 Technical Approach

2.1 Motion Model

In order to accurately determine the position of the robot in a known map motion data must be incorporated. This data will allow the algorithm to up-

date the empirical distribution over the robot's pose represented by the set of particles. In order to update this distribution we treat the wheel odometry data as consisting of three random variables $\Delta x, \Delta y, \Delta\theta$. The robot provides the mean values for these random variables as well as a 3x3 covariance matrix to model measurement uncertainty. We assume the 2D odometry of the robot $(\Delta x, \Delta y)$ can be modeled by a 2-dimensional multivariate Gaussian distribution using the upper 2x2 corner of the 3x3 covariance matrix. We also assume the independence of $\Delta\theta$ as it relates to the other odometry data and as such we model $\Delta\theta$ using a VonMises distribution with a mean of the measured $\Delta\theta$ and a concentration, κ, equal to the inverse of the variance of $\Delta\theta$ retrieved from the 3x3 covariance matrix. The VonMises distribution was chosen due to its similarity to the Gaussian distribution while still being circular, thus, giving accurate probability values for all angles.

Using the definitions above the motion model is implemented simply by sampling a new $\Delta x', \Delta y', \Delta\theta'$ for each point from the distributions specified above. This new sampled odometry data, $\Delta\mathbf{x}'$, which accounts for uncertainty is then used to update that particle's using the following equation:

$$\mathbf{x}_{\text{new}} = \begin{bmatrix} \cos(-\theta_{\text{old}}) & -\sin(-\theta_{\text{old}}) & 0 \\ \sin(-\theta_{\text{old}}) & \cos(-\theta_{\text{old}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \Delta\mathbf{x}' + \mathbf{x}_{\text{old}}$$

The new particles represent the distribution over robot poses after accounting for the most recent motion of the robot and uncertainty.

2.2 Sensor Model

2.2.1 Implementation

The sensor model component makes up another critical part of the particle filter implementation. Its specified goal is to leverage the set of reported scan beams and evaluate the probability weights for an input set of particles. To compute the evaluation while maintaining a low time complexity, the sensor model precomputes probability modifiers over discretized pixel distances, because the operations can be computationally expensive if implemented one to one from the continuous distance model.

Formally, the sensor model defines $p(z_k|x_k, d_k, m)$, which is the probability of recording a sensor reading z_k from a position x_k given true distances d_k on known static map m. Moreover, $p(z_k|x_k, d_k, m)$ can be evaluated as the product of the probabilities for recording each range measurement in sensor reading z_k. Letting $p(z_k^{(i)}|x_k, d_k, m)$ be the probability for recording the i-th range measurement, and s a power less than 1 to mitigate filtering by probability peaks and improve performance:

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

To complete this evaluation we define the model for $p(z_k^{(i)}|x_k, d_k^{(i)}, m)$ as the weighted sum of 4 probabilities (Let z_{max} be the maximum valid range value):

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

1. p_{hit}: Probability of detecting a known obstacle in the map. Let η be a normalizing factor such that p_{hit} integrates to 1 over $[0, z_{max}]$.

$$p_{hit}(z_k^{(i)}|x_k, d_k^{(i)}, m) = \begin{cases} \eta \frac{1}{\sqrt{2\pi\sigma_{hit}^2}} \exp(-\frac{(z_k^{(i)} - d_k^{(i)})^2}{2\sigma_{hit}^2}) & \text{if } 0 \leq z_k^{(i)} \leq z_{max} \\ 0 & \text{else} \end{cases}$$

2. p_{short}: Probability of a short measurement.

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

3. p_{max}: Probability of a very large measurement.

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

4. p_{rand}: Probability of a random measurement.

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

For our chosen weights and constants, we keep the recommended values as we observed satisfactory performance:

$$\alpha_{hit} = 0.74, \alpha_{short} = 0.07, \alpha_{max} = 0.07, \alpha_{rand} = 0.12, \sigma_{hit} = 8.0, s = \frac{1}{2.2}$$

Input distance measurements are converted from meters to the nearest pixel and bounded to the valid range by an intermediary operation:

$$\text{rounded\_pixel\_distance} = \lfloor \frac{meter\_distance}{map\_resolution} \cdot lidar\_to\_map\_scale \rceil$$

$$\text{bounded\_pixel\_distance} = \begin{cases} \text{rounded\_pixel\_distance} & \text{if } 0 \leq \text{rounded\_pixel\_distance} \leq z_{max} \\ z_{max} & \text{if rounded\_pixel\_distance} > z_{max} \\ 0 & \text{else} \end{cases}$$

2.2.2 Observations

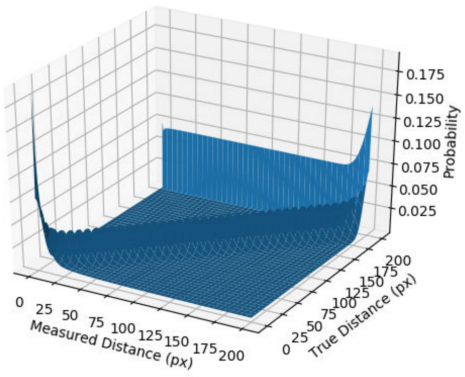


Figure 1: Distribution of probability weighting over measured and true distances

Our sensor model implementation passed the unit tests with the precomputed weights distribution shown on the above image. Additional experimenting with the weights and constants showed that the α_{hit} weight and s value played the largest role in determining the sensor model's impact on particle filter performance. If we increased the weight share of α_{hit} relative to the other weights, then we create more probability peaks with particles that have more scans near true distances. Additionally, with s, it proved a balancing act between not over fitting probability peaks and thus filtering particles that could be considered good enough, while also not completely dissolving useful probability peaks.

2.3 MCL through Particle Filtering

In this section we tie everything together in what's called Monte-Carlo Localization (MCL). MCL is a form of particle filtering from a family of recursive Bayesian algorithms. In MCL, instead of representing the robot's current pose as a singular point, MCL represents the pose as a collection of particles, each with different weights representing the probability of it being the robot's location. MCL then recursively updates the location of each particle based on sensor measurements and motion updates. In our case, the sensor measurements are

from a LIDAR, the motion from the odometry read from the IMU of the robot. This process is done by first

1. initializing the particles with random variations in pose. This is done in rviz through the 2D pose estimate tool.
2. Taking odometry data from the robot and updating the particle locations.
3. Updating the probabilities of each particle using the sensor data.
4. Resampling the particles given the new probabilities found using the sensor data.

This process is repeated indefinitely. In practice and simulation, the particles should coalesce around the robot. To test our algorithm, we utilized the racecar_simulator package to export simulated LIDAR data from the Stata Basement map to the /scan topic. The results from this testing are briefly summarized in the subsequent section.

2.4 Evaluation

To evaluate the efficacy of the localization algorithms, the robot was driven in simulation along a variety of paths in the Stata basement map. The car was driven straight near a consistent wall feature, around turning features, and along an oscillating path in a complex feature environment. These paths were designed to test the limits of the localization algorithm. The first path is shown below:

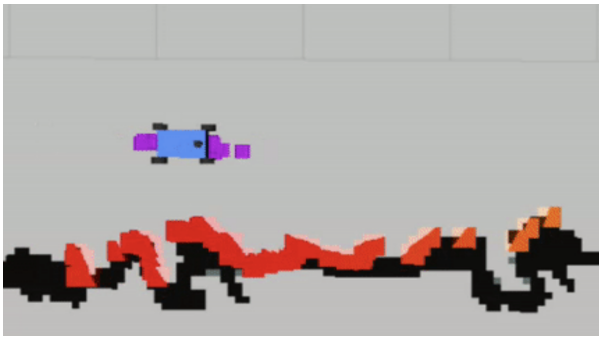


Figure 2: Testing path 1, straight line alongside wall

This path tested the error that the localization would report in cases where the LIDAR data has trouble eliminating many points. In this situation, the possible locations in front of or behind the ground truth location are expected to fit similar LIDAR scan profiles. This resulted in the purple point cloud extending in front or and behind the car as well. When testing the error in simulation (measured as the distance between the ground truth pose and the predicted pose from localization), the component of error in the x direction was

consistently around 0.2 meters and composed of nearly all the reported error as seen below:

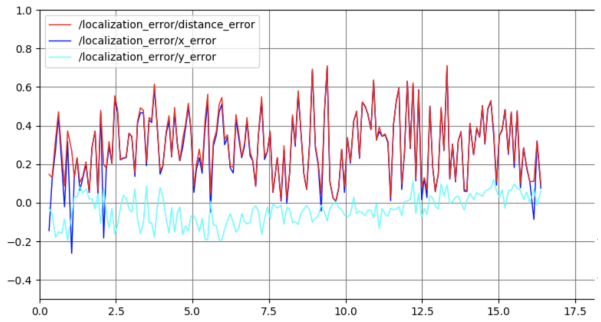


Figure 3: Error over time from localization along testing path 1

In addition, the error along the y direction was minimal, around 0 meters. This showed promise, as the model was successfully able to find and fit points that fit the LIDAR scan and could result from the odometry with noise.

The next two paths testing the robot localization in more complex scenarios. These paths included turning and driving randomly in areas without a consistent feature (like a wall):

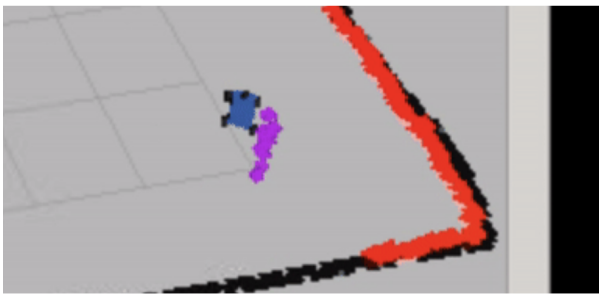
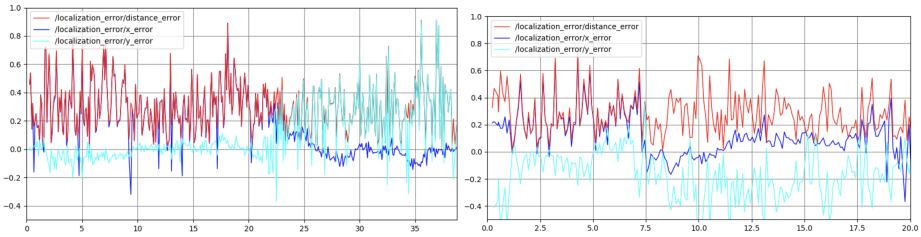


Figure 4: Testing path 2, turning alongside wall

These two other paths posed a larger challenge for the localization, and corresponded to a higher average error, closer to 0.3 meters on average. This makes sense as the odometry is more complicated, involving changing directions and speeds. This could result in the predicted particle locations drifting, especially as the car took sharp turns. Below are the error recordings from these paths:

Overall, the localization algorithm was able to predict robot pose within 0.6 meters at worse. This was deemed a success for the team as the error averages around 0.3 meters, which was roughly the size of the car itself. From a qualitative perspective, the localization algorithm maintained a point cloud centered around the car and was able to recover from drift observed in the turning cases.



(a) Extreme inside corner (b) Extreme outside corner

Figure 5: Examples of extreme turn angles

3 Conclusion

Through Lab 5, Group 4 managed to complete all deliverables by successfully designing a Monte Carlo localization algorithm and estimating robot pose within reasonable error. This lab was challenging due to the integration of the different components and overcoming debugging more complicated issues with the algorithm and car infrastructure. From a communications standpoint, this lab was a resounding success. Car handoffs were done smoothly and resources were shared in an effective and equitable manner. In addition, the group was able to parallelize work across different modules and come together on a consistent timeline, which is a far improvement from previous labs.

4 Individual Reflections

4.1 Kairo Morton

From my perspective, the lab 5 assignment was a success. I found my section, which involved writing the motion model, to be straightforward, and I was able to complete it with ease. Additionally, I enjoyed assisting Gabriel in debugging the sensor model, which allowed me to contribute to the team's success beyond my assigned tasks. Finally, clear communication between team members played a critical role in making the integration of our work mostly seamless. This experience has continue to show me the importance of effective communication in any collaborative project, particularly those that involve complex robotics tasks such as localization with many components. Overall, I am proud of what our team accomplished and believe we worked well together to achieve our objectives.

4.2 Rohan Wagh

This lab was more challenging due to the integration components. A lot of our time was spent debugging and incorporating code rather than designing and writing code. This switch was challenging to time and we had to spend some

time properly planning out our workflows ahead of time. This was something I was not as used to doing and a great learning experience for proper time management in group settings.

4.3 Gabriel Jimenez

As for communication and collaboration lessons, I would say that my experience during Lab 3 has validated my approach of direct and open communication, with clear central documentation of outstanding tasks and assignments. We had little friction in terms of dividing the required technical and communication work amongst the team members.