Lab 5 Report: Localization

Team 9

Zhenyang Chen
Kwadwo Yeboah-Asare Jr.
Meenakshi Singh

6.141/16.405 Robotics: Science and Systems

April 6, 2022

1 Introduction

Author: Meenakshi
Localization is a problem within robotics that focuses on determining the position of the robot within a known map. The goal of RSS Lab 5 was to understand and implement the Monte Carlo Localization algorithm in three parts: the motion model, the sensor model, and the particle filter.

1. The motion model portion of the lab involved taking the wheel odometry coming from integration of motor and steering commands and applying it to the current position of the robot.

2. The sensor model portion of the lab involves taking the position of the particles determined by the motion model and assigning the particles weights based on their likelihood. This step increases the chances of particles that are more likely to represent the robot position being sampled on successive iterations. In our approach to the sensor model, we constructed a probability look-up table where each column contained normalized Lidar measurements d_k and each row contains the distance z_k from raycasting at the particle. We did this in order to reduce the computational complexity of having to recalculate probabilities for each particle.

3. The particle filter takes the results of the motion model and sensor model. Using the odometry data and the motion model we update the particle positions. Then, we use the sensor model to compute the particle probabilities and resample the particles. For each update of the motion or sensor model, we calculate the average particle pose and publish that transform.

We were able to gain experience in all three goals by designing, testing, and implementing the wall follower and safety controller on the physical racecar system. In the following sections we detail our approach for the wall follower and safety controller.

2 Technical Approach

2.1 Motion Model

Author: Nana
For the lab we applied the method from the probabilistic robotics textbook to get the position the robot in the world frame given an odometry.
The method comes in three steps:

1. Rotate the car in the direction of the odometry vector.
2. Move the car the magnitude of the odometry vector in the current direction.
3. Rotate the car the remain degree to allow its total change in orientation to match the odometry.

The equations for theses various steps are outlined below

First rotation in direction of odometry vector

$$o_{rot1} = arctan2(dy, dx)$$

magnitude of movement in direction of odometry

$$o_{tran} = mag(dx, dy)$$

remaining rotation to get the correct orientation for the robot

$$o_{rot2} = d_o - o_{rot1}$$

To add noise to the model we simply add some gaussian noise with a suitable standard deviation to the d_{theta}, dy and dx values given by the odometry.
Here are images of the particles spreading out as the motion model progresses without resampling.

2.1.1 Sensor Model

Author: Zhenyang
Introducion
After updating the estimated state for each particle, we will create measurements for each particle. Theses measurements will be used to compare with the real measurement at time t, provided by the Lidar on the car and tell us which

particle has the largest possibility to represent the current robot state. These probabilities will be used in the resampling process when we build the particle filter.

Probability of Sensor Model

The probability model of sensor contains several parts: gaussian noise, higher probability of hitting objects closer to the robots, maximal range, and random noise. These noises can be represented by the following equations:

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

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

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

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

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

Precomputed Sensor Model

In our application, we are interested in finding a probability that represent how likely a Lidar measurement represents a real robot state. Where m is the given map, x_k is the real state and z_k is the measurement coming in.

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

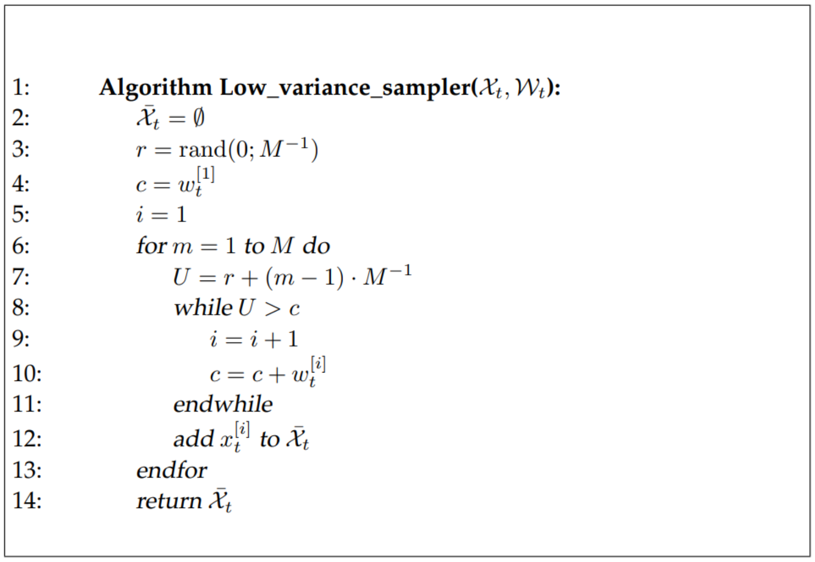
When using the particle filter, we need to calculate this probability every time step which involves in calculating n (number of the particles) probability. To reduce the computational complexity, we will build a probability look up table where the column is the distance from Lidar measurement d_k and the row is the "real" distance z_k given by the particles. We built up a table with 201*201 dimension, start from $d_k = 0$ and $z_k = 0$. To make the probability reasonable, we also normalized the Gaussian distribution and probability for each column. And the figure below shows the what look up table looks like.

Evaluate function:

For the evaluate function, we need to return probability value of each particle using equation (1), given the estimated motion state and the Lidar measurement. We first rescaled the state of particles and measurements of the Lidar, turning them from meters to pixel. And then to make the probability reading process more efficient, we utilized the slicing techniques provided by the numpy.array. We tiled up the rescaled Lidar measurements data and used it as a column indices and did the same for the particle states.

2.1.2 Particle Filter

After building the motion model and sensor, we combined them together to make the whole particle filter. The idea is simple. Everytime we receives odometry data, we update the states of all particles using motion model. When we receive Lidar data, we do the ray-casting and evaluate particles, which return a probability weights for us.
Low variance resampling:
Author: Zhenyang
After testing our first version of



```
Algorithm Low_variance_sampler($\mathcal{X}_t, \mathcal{W}_t$):
    $\bar{\mathcal{X}}_t = \emptyset$
    $r = \text{rand}(0; M^{-1})$
    $c = w_t^{[1]}$
    $i = 1$
    for $m = 1$ to $M$ do
        $U = r + (m-1) \cdot M^{-1}$
        while $U > c$
            $i = i + 1$
            $c = c + w_t^{[i]}$
        endwhile
        add $x_t^{[i]}$ to $\bar{\mathcal{X}}_t$
    endfor
    return $\bar{\mathcal{X}}_t$
```

Figure 1: Low Vari

Simulation experiment:

Real car experiment:

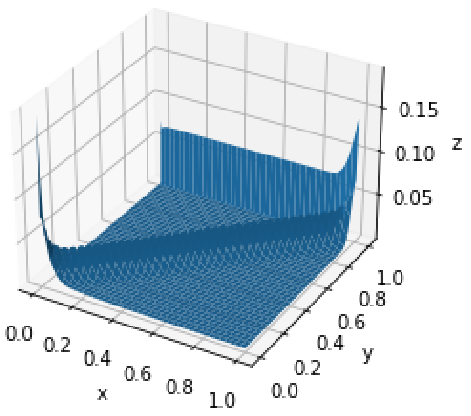


Figure 2: Precomputed Sensor Model

3 Experimental Evaluation

As a result of various hardware and technical issues, we were not able to conduct experimental trials. Instead we describe our methodology and set up for the empirical tests we conducted of the wall follower and safety controller.

3.0.1 Wall Cases

We ran several tests to verify the accuracy of the wall follower in different physical testing situations and different velocities.

- Inner corner at 1 m/s - The car did very well at following the pure pursuit trajectory stably
- Outer corner at 1 m/s - Just like with the inner corner, with nice enough geometry
- Inner corner at 2 m/s - The car went smoothly around the inner corners, thought it would get very close to the wall
- Outer corner at 2 m/s - Going around the corner at higher speeds typically left the car going very wide and once it had turned the corner the path would be unstable for a long time. It would oscillate around the pure pursuit trajectory for a bit instead of smoothly approaching it
- Narrow corridor at 1 m/s - At 1 m/s the car was always able to navigate the corner

- Narrow corridor at 2 m/s - At 2 m/s the car was only able to successfully complete the turn at certain orientations entering the turn. Otherwise it would get too close the wall and activate the safety controller.

3.0.2 Safety Controller

The experiments designed to test the safety controller's effectiveness were based on how the controller performs under different stopping conditions. We divided the test situations into:

- Static situations, driving into an existing obstacle/wall at 1 m/s and 2 m/s. We also planned to drive towards the wall starting from different initial distances, measuring the total stopping time and the distance it took to stop at different velocities.
  - In static situations, as long as the car was going slow enough to detect the wall as it entered it's field of detection in time, the safety controller worked as expected.
- Dynamic situations, such as a person walking in front throwing a box in front of the robot.
  - As in the static simulations, if the car controller was given enough time to break and the Lidar read distance was high enough it did stop successfully.

Although we did not get to analyze the data, we hope to continue working on this to improve our wall follower and safety controller.

4 Conclusion

Author: Meenakshi
Over the course of lab 5, we learned about Monte Carlo Localization. We first understood the mathematical theory behind the concepts of the motion model, sensor model, and particle filter. Then, we implemented the models in simulation and refined them until we were satisfied with their accuracy. Finally, we tested our particle filter on the racecar in Stata basement and refined it. Through this process, we were able to build up the parts of the particle filter throughout the lab. It also demonstrated the discrepancy between simulated testing and real-world application as our team had to spend

5 Lessons Learned

Presents individually authored self-reflections on technical, communication, and collaboration lessons you have learned in the course of this lab.

5.1 Meenakshi

Due to personal circumstances, I was unable to help with many of the technical aspects of the lab early on. However, despite the beginning circumstances, I was able to gain experience with collecting and processing data from recorded rosbags. I also gained a better understanding of the challenges that come with localization problems. Although I have a theoretical understanding of the model utilized, I am not as familiar with the problems that came with testing it on the robot. Another thing to improve moving forward is to document our progress along the way in a clear manner because we had methodical approaches to improve the performance of the particle filter based on testing, but the iterative changes were not documented so looking at the code between versions can feel like large gaps in knowledge.
My biggest takeaways from this lab is about the importance of communication within the team during the face of external factors. I am very apologetic that my situation contributed to the team falling behind. Even though I could not control those external factors, I could have been more responsible by actively communicating with my teammates so that the division of work was more fair to them.

5.2 Zhenyang

The biggest lessons I learned from this lab is to see the large gap between theory and engineering practice and see the importance of communication and collaboration in an engineering challenge. Particle filter is a kind of Bayesian filter using Mente-Carlo method to sample motion state and estimate the real robot states. Though the idea seems simple, the mathemacical derivation looks messy and straight forward. When it comes to implementation, we need to deal with more realistic cases. For example, to reduce computation complexity, we pre-compute the sensor model, and to mimic the real motion state of the robot, we add noise in motion model. This details and tricks are not necessarily included in the formulas, but it is definitely worth noting and thinking, and they result in the correct performance in real application.
This week is challenging for me and for the team. For the last two weeks, I lost connection with my teammates. One of them feels not well, the other one decided to drop the class before briefing. This leads to a lag in progress and makes me realize everyone is important and responsible in an engineering team. I believe with coming back of my teammate and the join of new teammate, we can do much better in the Lab6. Keep working, keep updating with others.

5.3 Kwadwo

The team was always ready to lend a hand in making sure the lab turned out great. I feel we can do much better with fewer hardware failures and a better division of labour as it seemed like half the team was working on things in such a way the other half could not start working if they didn't finish. We also need to make a greater effort to record runs of the lab since we had many instances were we had run tests but collected no data on them.

For the technical aspects I learnt more about car control than I ever wished to know. I have experience with tuning models for the robot and what variables change what behaviour and how to trade things for different performances. I am more familiar with the robots Linux base, transferring files with SCP, editing parameters during testing and generally navigating a file system from terminal. I need to get better with GitHub though.