Monte Carlo Localization based Mapping

Qiong Wang
University of Pennsylvania
3401 Chestnut St., Philadelphia
qiong@seas.upenn.edu

Daniel Lee
University of Pennsylvania
3401 Chestnut St., Philadelphia
ddlee@seas.upenn.edu

Abstract

Simultaneous localization and mapping (SLAM) is a concept that combines localization, planning, navigation in a loop and usefully localizing the mobile robot when sensoring the information from the environment in the same time. In this paper, one Monte Carlo Localization (MCL) algorithm was developed to estimate the position and orientation of a robot as it moves and senses the environment.

1. Introduction

This project was accomplished for coursework in ESE 650: Learning in Robotics at University of Pennsylvania. The objective is to implement one Simultaneous localization and mapping (SLAM) algorithm to build up a map within an unknown environment while at the same time tracking the current location based on the data provided by Encoders, Inertial Measurement Unit (IMU) and laser scanning. As one way to localize the current position, Monte Carlo Localization (MCL) was used to estimate the current pose by applying particles each time.

Monte Carlo Localization (MCL) is also called particle filter localization which estimates the position and orientation of a robot as it moves and senses the environment when given a map of the environment. Here we use this algorithm as a particle filter to represent the distribution of likely states, with each particle representing a possible state. And then the correlation with the environment was computed based on the laser scan to find the best particle to describe the state and then update both the state and map.

In this project, a MCL algorithm was implemented to estimate the pose (x, y, θ) of the robot given the data from encoder, IMU and laser scan. The algorithm, the results and even the recognition precision table are presented in details below.

2. Wheel Odometry

In the wheel odometry part, the data from the encoders as the wheel sensor were used to update the position of the robot. If a robot starts from a position p, and the right and left wheels move respective distances as Δsr and Δsl, we can find the resulting new position p by modeling the change in angle $\Delta\theta$ and distance travelled Δs by the robot. Assuming the robot is travelling on a circular arc of constant radius, we can have as Figure 1 shown.

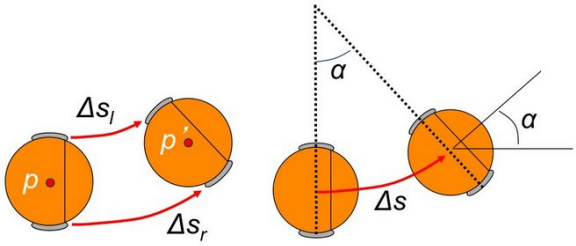

Figure 1. Wheel Odometry Model 1

$$\Delta s = \frac{\Delta sr + \Delta sl}{2}$$
$$\Delta\theta = \frac{\Delta sr - \Delta sl}{c \times B}$$

where the B is the robot width, which was also factorized by c in the code with consideration of the wheel slipping.

As shown the Figure 2 shown, we can also find the change in coordinates as

$$\Delta x = \Delta s \cos(\theta + \Delta\theta)$$
$$\Delta y = \Delta s \sin(\theta + \Delta\theta)$$

Here Δs is approximately considered as Δd since it is the small change between two time stamps [1].

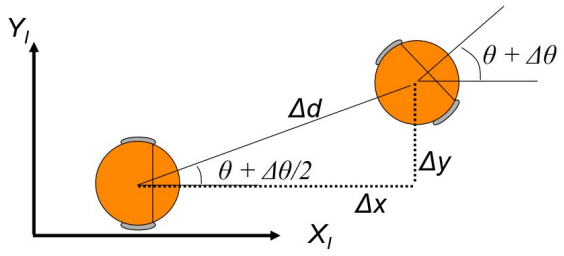

Figure 2. Wheel Odometry Model 2

3. Dead Reckoning

As we have the result from the encoders by modeling the motion for odometry [2]. We can have the dead reckoning by only using wheel sensors and heading sensor to update position. Here is the dead reckoning result of dataset 23 as Figure 3. We can find the path in the beginning should be flat and no big bearing while this bearing error was integrated in all the following path and the result was not as the actual building is.

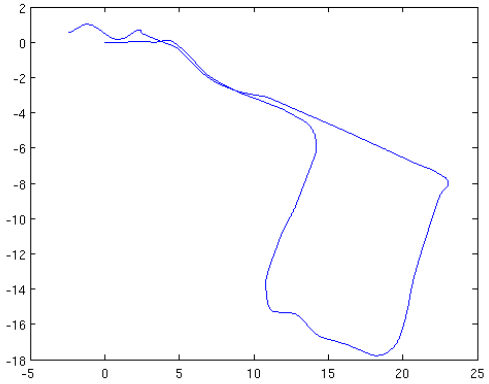

Figure 3. Dead Reckoning result of dataset 23

4. Particle Filter

A particle filter estimates the posterior distribution of the hidden states using the observation measurement process. Here is a single step of MCL particle filter [**?**]:

1. For L=1,...,P draw samples from the proposal distribution
$$x_k^{(L)} \sim \pi(x_k | x_{0:k-1}^{(L)}, y_{0:k})$$
2. For L=1,...,P update the importance weights up to a normalizing constant:
$$\hat{w}_k^{(L)} = w_{k-1}^{(L)} p(y_k | x_k^{(L)})$$
3. For L=1,...,P compute the normalized importance weights:
$$w_k^{(L)} = \frac{\hat{w}_k^{(L)}}{\sum_{J=1}^{P} \hat{w}_k^{(J)}}$$
4. Compute an estimate of the effective number of particles as
$$\hat{N}_{eff} = \frac{1}{\sum_{L=1}^{P} \left(w_k^{(L)}\right)^2}$$
5. If the effective number of particles is less than a given threshold $\hat{N}_{eff} < N_{thr}$, then do resampling as extract P particles from the current particle set with probabilities proportional to their weights. Then replace the current particle set with this new one and reset the weights as $w_k^{(L)} = 1/P$.

The result of the MCL will be covered in the results part.

5. Ground Detection

In the ground detection part, after extracting data from the kinect, the data first converted as the format $(x, y, depth)$ and then computed the surface normal by using MATLAB built-in function $surfnorm$ and then clustering the depth as Figure 4. Sorry for no time to integrate the kinect data into the map bus just clustered ground as Figure 5.

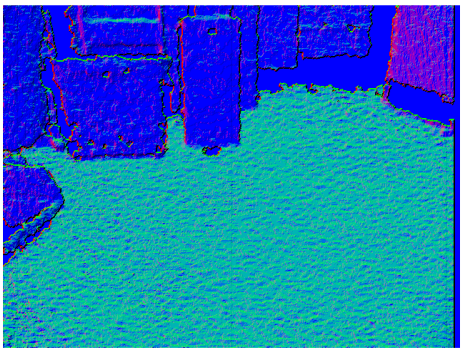

Figure 4. Surface Normal from disparity

6. Results

In this project, the result after MCL was not as good as imagined. Here is one of the example result. The reason for this might from the `map_correlation`. I have found some of the correlation seems not so good. I will try to rewrite the map matching function to see whether it works.

Here Figure 6 is from the testset 1 and Figure 7 is the result of testset 2. The first test set seems OK despite the

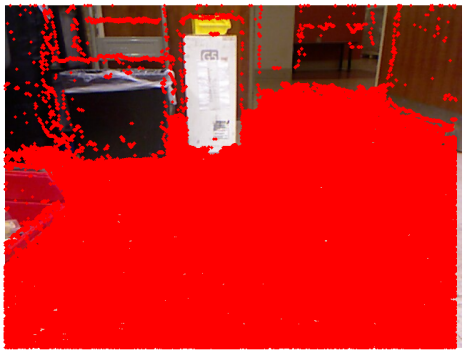

Figure 5. Detected Ground

uncorrected small rotation in the orientation changing. The second test set is bad since the jittering of the robot makes the correlation matching not so precise each time. So as to make the code efficent, I make the resolution of x and y smaller in the computation of correlation which might give worse correlation estimation. The third dataset is still running since the long running time.

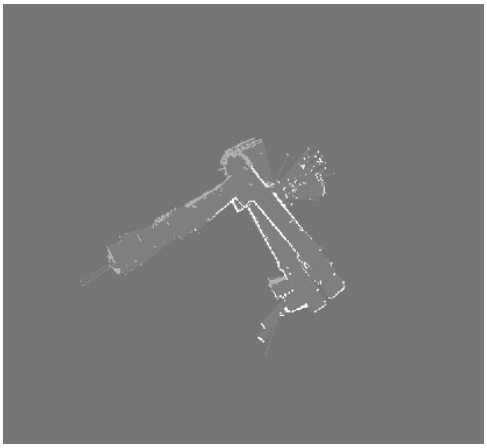

Figure 6. Map result of test dataset 1

7. Conclusion

In this project, one MCL algorithm was implemented to estimate the position and orientation of a robot as it moves and senses the environment. The result was not so great when there is huge jittering and the time was also insufficient for the kinect data integration. I will probably resubmit again for a better version of code later. Thank you for your understanding.

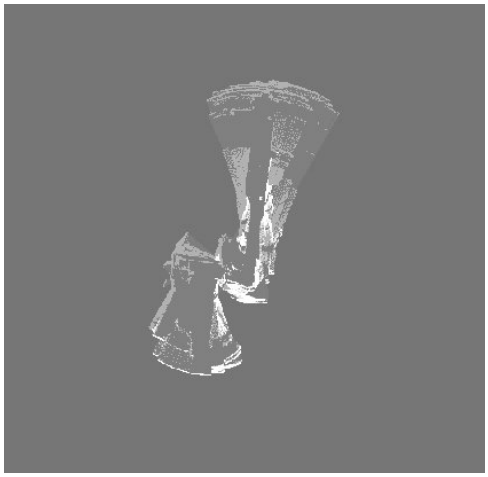

Figure 7. Map result of test dataset 2

8. Acknowledgement

Thanks to Professor Lee, Alex and Zhuo. Thank you so much to prepare this great project.

References

[1] C. Clark. Autonomous robot navigation, Fall 2011.
[2] D. Lee. Lecture notes for ese 650: Learning in robotics, 2014.