ECE5242 Project3: Particle filter based SLAM

Sijia Gao

April 3, 2019

Abstract

The aim of this project is to utilize encoder and lidar recordings to simultaneously construct surrounding map and localize the robot. To achieve this, we implement a particle filtering algorithm that consists of three steps. First, we sample particles from the encoder. Second, particle weights are updated using correlation-based matching algorithm with the current map. Finally, a occupancy grid mapping algorithm is implemented to refresh the map with the best particle. The proposed method nicely integrates simultaneous map construction and robot localization which contributes to high accuracy.

1 Introduction

Simultaneous localization and mapping (SLAM) describes a problem of updating a map of an unknown environment while simultaneously keeping track of an agent's location within it. It generalizes a way for the machine to perceive its surrounding environment and localize itself at the same time. SLAM algorithms are tailed to various cases such as self-driving cars and unmanned aerial vehicles. Popular SLAM methods include particle filter (used in this paper) and extended Kalman filter.

The organization of this report is as follows: Sec.2 describes the project setup and the pipeline of the proposed solution; Sec.3 illustrates each part in the particle based SLAM algorithm in detail. Sec.4 illustrates performance of the proposed method on the test set. Sec.5 concludes the whole report.

2 Problem formulation

In this project, we use wheel encoder and 2D lidar recordings from a moving robot to simultaneous construct the surrounding grid map and localize the robot. Each grid of the map has two states: "occupied" and "free" indicating if there are obstacles in the grid.

3 Particle filter based SLAM method

The pipeline of the proposed method include three steps:

(1) Sample particles using encoder

(2) Particle weight is updated using correlation-based matching with the current map

(3) Map is refreshed using occupancy grid algorithm (Log-odd addup)

Now we move onto describing the above three steps in more details.

3.1 Sample particle

The encoder records the distance traveled by the 4 wheels of the robot. Denote d_r, d_l the traveled distance of right and left wheel, respectively. State of the robot at discrete time t is a three-tuple $s_t = [x_t, y_t, \theta_t]$ that consists of position in the x, y axis and the heading (in radian). State dynamic is illustrated as follows:

$$\begin{aligned} \Delta\theta &= \frac{d_r - d_l}{Bb} \\ \theta_t &= \theta_{t-1} + \Delta\theta \\ x_t &= x_{t-1} + \frac{d_r + d_l}{2}\cos(\theta_t) + w_x \\ y_t &= y_{t-1} + \frac{d_r + d_l}{2}\sin(\theta_t) + w_y \end{aligned} \tag{1}$$

Here, $B = 1.85$ denotes the scale factor for slippery. b is the width of robot. w_x and w_y are Gaussian noise with zero mean and standard deviation$=\frac{|d_r|+|d_l|}{2}$.

3.2 Particle weight update: correlation based map matching

Denote x_t^i the ith particle at discrete time t and its weight w_t^i. In this part, we discuss how to update w_t^i that verifies if the particle accommodates the current map. In our work, w_t^i is computed by

$$w_t^i = w_{t-1}^i \times \text{corr}(x_t^i, \text{map}) \tag{2}$$

$\text{corr}(x_k^i, \text{map})$ provides the effectiveness of a particle accomodating the current map and is computed as the sum of values of all cells hitted by the particle's laser. It describes if laser emitted from this particle really hits obstacles on the map. 50 Particles are initialized with $[0, 0, 0]$. Weights at each iteration are normalized and resampling technique is implemented when fraction of number of effective particles is less than 0.25.

3.3 Map update

Each grid on the map is represented by a value called Log-odd which is updated using the following equation:

$$log\frac{p(m_{x,y}=1|z)}{p(m_{x,y}=0|z)} = log\frac{p(z|m_{x,y}=1)}{p(z|m_{x,y}=0)} + log\frac{p(m_{x,y}=1)}{p(m_{x,y}=0)} \tag{3}$$
$$\text{Log odd} = \text{Log LH ratio} + \text{Log prior ratio}$$

Here, z={1,-1} represents if the cell is hitted by the laser. In our work, we set Log LH ratio (z=1)=3 and Log LH ratio (z=-1)=0. From equation(3), we can see that log odd value of each cell is updated by simply adding Log LH ratio at each discrete time. In practice, we only update occupied cells detected by the best particle. Log odd of each cell is bounded within [0,127].

4 Simulation results

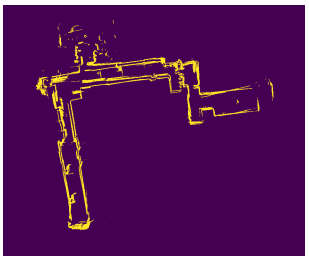
In the initial code submission, we do not implement the particle filter but rather focus on encoder itself to give the target's position and heading. Map construction is based on lidar and encoder pose estimation. Cells hitted by the laser are updated using log-odds while free cells (cells located on the segment connecting robot and hit cells) are ignored. Results for our initial submission are illustrated in Fig.1, we can see that our algorithm can nicely detect the obstacled boundaries. To plot map, cell values above 60 are set to 1, otherwise they are set to zero and then we use matplotlib in python to plot.

We then try implementing a particle filter based SLAM algorithm. It operates in the way that simultaneously update robotic position and headings which are accommodated with the current map. Simulation results with both refreshment on occupied and free cells are shown in Fig.2. Results for the particle filter based SLAM illustrated in this report are illustrated in Fig.3.

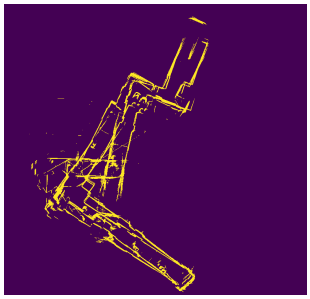
5 Conclusion

In this project, we use encoder and lidar to construct the surrounding map and localize the robot. Our method can nicely show the obstacle boundaries.

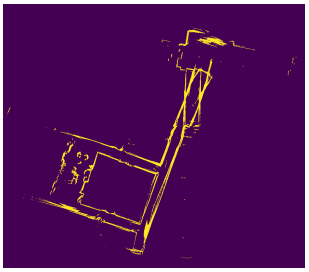
The key part in this project is to give accurate estimation of the heading θ_t because the accumulated error of θ_t can result in large betray from the groundtruth trajectory. Our work prefers to give clean and sharp boundary detection. To achieve this, we enrich particles by adding noise only to position and log odd update is implemented only to the occupied cells. However, we also notice that if a temporary obstacle exists for quite a long time during robot's forward path and disappears when robot returns, this fake obstacle is expected to be removed. Future solution should include log-odd refreshment on free cells to tackle this condition.



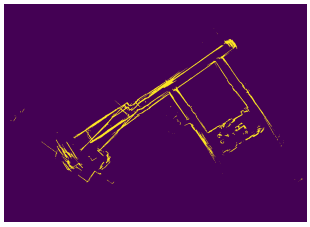
(a) dataset20



(b) dataset22

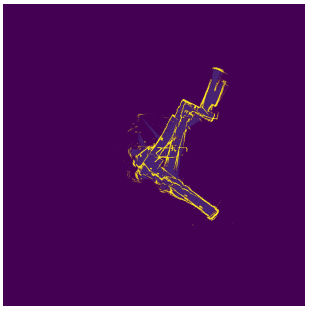


(c) dataset23

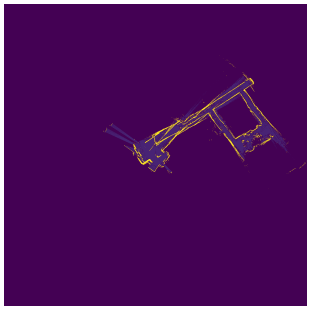


(d) dataset24

Figure 1: Results for our initial code submission.

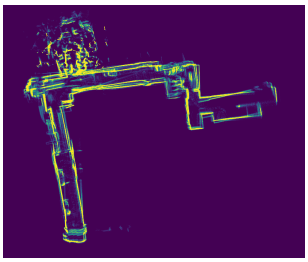


(a) dataset20

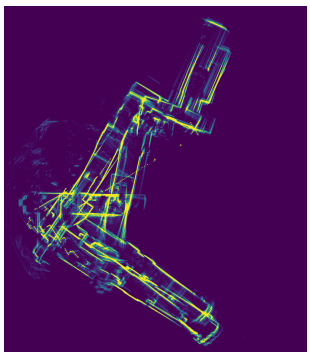


(b) dataset22

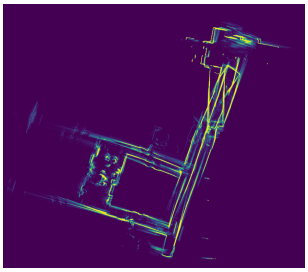
Figure 2: Results for particle filter based SLAM (freecell considered).



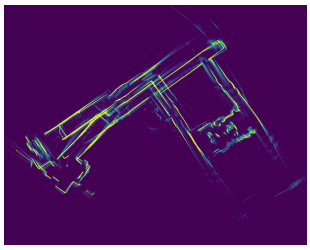
(a) dataset20



(b) dataset22



(c) dataset23



(d) dataset24

Figure 3: Results for particle filter based SLAM illustrated in this report.