EECS498 Final Report: Experiments on Methods for Probabilistic Localization

Ruohua Li
ruohuali@umich.edu

Abstract

In this report I present an experimentation on two methods to solve the localization problem, namely, Kalman Filter [3] and Particle Filter [2]. I experimented these two method in the setting that the robot receives noisy GPS signals and control commands and tries to localize itself inside a room with obstacles while moving from one end to the other.

1. Introduction

1.1. Overview

The robotic localization problem refers to the problem that endeavors to localize a robot or robots with respect to an environment. In order to navigate any robotic system inside any environment effectively, the first step is always obtaining an accurate estimation of the current location of the robot, *i.e*. let the robot localize itself with the dynamics introduced both by itself and the environment.

To determine the current location of a robot, in reality, there are two references at our disposal, sensor-measured data as well as control commands. In most setting there is at least one kind of sensor such like camera and GPS to help the robot gain a relatively inaccurate location of itself. And in the settings where robots are actuated, the control or planning commands sent to the actuators is also helpful for localization when provided along with the robots' dynamic model.

Most methods that try to solve the localization problem take a probabilistic view of this world, which means that these methods assume that neither the sensor data nor the control commands are accurate entirely. This means that the states provided or extracted only reflect locations where the robots are more likely to be present. This thought lead to the development of Bayesian Filter method which employs Bayesian theorem.

In this report, I experimented two methods that are developed based on the idea of Bayesian Filter, namely, Kalman Filter and Particle Filter. I will briefly describe the basic idea of Bayesian Filter and then introduce Kalman Filter and Particle Filter in this section.

1.2. Bayesian Filter

Bayesian Filter stems from Bayes Theorem which states that the posterior probability of a random variable can be described by the likelihood, prior probability, and marginal probability.

$$P(X|Y) = \frac{L(Y|X)P(X)}{P(Y)}$$

Bayesian Filter is developed under the Markov assumption and Markov sensor assumption. It transforms the process into two steps, (1) prediction step and (2) update step. In prediction step, the dynamic model of the robot and the probabilities calculated in previous time steps are used to establish an estimation regarding the current location of the robot, which is the prior probability. In the following equations x_k, u_k, z_k represents the location, control command, and sensor data of robot at time step k.

$$P(x_t|z_{t,\dots,t-1}, u_{t-1}) = \sum_{\forall x_{t-1}} P(x_t|x_{t-1})P(x_{t-1}|z_{t-1,\dots,1}, u_{t-2})$$

And in the update step, the conditional probability provided by sensor model of the robot is used as the likelihood to refine the posterior probability of the current

location.

$$L(z_t|x_t) = P(z_t|x_t)$$

Afterwards the prior probability and posterior probability are combined in the manner provided by Bayes Theorem to give an optimal estimation of the current location.

1.3. Kalman Filter

Kalman Filter is derived from the assumption that the dynamic model and sensor model of the robot is linear, meaning these models can be described by linear stochastic difference equations. Also, it assumes that the noise introduced by both dynamic model and sensor model are additive and sampled from Gaussian distributions. It can be shown [1] that if the assumptions are met, the Kalman Filter will give the optimal estimation.

$$\begin{aligned} x_t &= Ax_{t-1} + Bu_t + \epsilon_t \\ \epsilon_t &\sim \mathcal{N}(\mu_\epsilon, \sigma_\epsilon) \\ z_t &= Cx_t + \delta_t \\ \delta_t &\sim \mathcal{N}(\mu_\delta, \sigma_\delta) \end{aligned}$$

The estimation of Kalman Filter is given in the form of a Gaussian distribution, in which locations where robot are calculated to be likely to be present currently have higher probabilities. At each time step t, as a function it takes in mean μ_{t-1} and covariance matrix σ_{t-1} which are the results of the previous step as parameters and returns the newly estimated distribution.

1.4. Particle Filter

Particle Filter is also known as sequential Monte Carlo estimation, which shows that it adopts a sampling-based estimation to perform localization. It makes no assumptions regarding the dynamic model and sensor model other than that they are consistent through out the sequence, which gives it more flexibility compared to Kalman Filter.

Particle Filter keeps a set of samples which is a subset of all possible locations the robot can be present called "particles", $\{p_1, ..., p_N\}$. The number of particles N is a hyperparameter for the model, and later in the experiments I chose $N = 100$. Each particle is associated with a normalized weight w which indicates the probability for the robot to actually be at the location. Since the Particle Filter gives a set of particles and their corresponding weights, a natural way to do inference based on them would be to use the weighted average location of the particles.

$$x^* = \sum_{i=1}^{N} w_i p_i$$

2. Implementation

In this section I will describe the settings based on which my implementations will be applied. Furthermore, I will depict the ideas and specifics of my implementation of the two algorithms and explain the reasons for my own design choices.

2.1. Environment Settings

The experiment setting for my implementation to work in is a room with obstacles, and the robot will try to localize itself while it moves from one end to the other. Naturally, the localization problem in this setting is to determine a 2D coordinate $(x, y)_t$ at time step t which describes the location of the robot.

To make the problem interesting enough, the control command and sensor data are noisy. The control commands are exposed to the localization algorithms in the form of 2D relative offsets, *e.g.*, $(4, 5)$ means the robot moved 4 meters in the direction of x axis and 5 in the direction of y axis. The sensor data is given as 2D coordinates which is an inaccurate measurement of the robot's location, *e.g.* $(10, 20)$ means robot is measured to be at 10 meters away from the origin in the direction of x axis and 20 in the direction of y axis.

2.2. Kalman Filter

As mentioned, the Kalman Filter at each step will make estimation in the form of a Gaussian distribution which is derived from the Gaussian distribution provided by the previous step. Specifically, in the prediction step, the Kalman Filter will move the given Gaussian distribution using the linear dynamic model. Then in the update step, it will calculate the "Kalman gain" based on the sensor data and correct the distribution obtained in the prediction step. In the following nota-

tions R is the covarinace matrix for the control Gaussian noise and Q is the covarinace matrix for the sensor Gaussian noise.

Algorithm 1 Kalman Filter

Input: $\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$
$\hat{\mu_t} \leftarrow A\mu_{t-1} + Bu_t$
$\hat{\Sigma_t} \leftarrow A\Sigma_{t-1}A^T + R$
$K_t \leftarrow \hat{\Sigma_t}C_t^T(C_t\hat{\Sigma_t}C_t + Q_t)^{-1}$
$\mu_t \leftarrow \hat{\mu_t} + K_t(z_t - C_t\hat{\mu_t})$
$\Sigma_t \leftarrow (I - K_tC_t)\hat{\Sigma_t}$
return μ_t, Σ_t

Given the form of coordinate data and dynamic model I chose the transfer matrices A and B to be 2-by-2 identity matrices I_2. And covariance matrices Q and R are set to

$$R = \begin{bmatrix} 9.24 \times 10^{-4} & 1.44 \times 10^{-6} \\ 1.44 \times 10^{-6} & 9.24 \times 10^{-4} \end{bmatrix}$$
$$Q = \begin{bmatrix} 1.78 \times 10^{-2} & 7.10 \times 10^{-4} \\ 7.10 \times 10^{-4} & 3.6 \times 10^{-1} \end{bmatrix}$$

At each time step, the estimation is given by the mean of the Gaussian distribution.

2.3. Particle Filter

An instance of the Particle Filter contains a set of particles, in which each particle is a location that the robot maybe present. There is also a set of normalized weights that associate with the particles representing each particle's probability of being the actual localization result.

Similar to Bayesian Filter, Particle Filter can also be divided into prediction step and update step. In the prediction step, it moves all of its particles following the dynamic model and control command. But in order to counter the inaccuracy induced by the control noise, it adds a probabilistic variable to the control command to make every particle move in a way that is a little bit different than the input control command. I chose to add a uniform distributed noise $\epsilon \sim \mathcal{U}$ with a small radius to achieve this effect. Then in the update step, it adjusts the weights of each particle according to the sensor data and then normalize the weights to make them acts like probabilities. I chose to represent the likelihood provided by the sensor data as inverse of Euclidean distance from the sensor provided location.

$$L((x, y)|(x_s, y_s)) \propto \frac{1}{\sqrt{(x^2 - x_s^2) + (y^2 - y_s^2)}}$$

However there exists a potential problem that may affect the performance of Particle Filter negatively: after several iterations, most particles may be assigned small weights while handful particles have high weights, which means that the number of particles contributing to localization inference is extremely small. To resolve this issue, we need to resample the particles when the number of particles with relatively high weights are small. I chose to resample the particles in such a way that if the number of particles with a relatively high weights is smaller than a threshold, sample N particles from the existing particles with the probabilities of being chosen same as their own weights. The threshold is determined by the effective sample size of particles which is defined as

$$n = \frac{1}{\sum_N^{i=1} w_i^2}$$

When $n < \frac{N}{3}$ the algorithm will resample all particles.

Algorithm 2 Particle Filter

Input: $p_{t-1}, w_{t-1}, u_t, z_t$
$p_t \leftarrow p_{t-1} + u_t + \epsilon$
$w_t \leftarrow \frac{1}{\eta} w_{t-1}(\text{ dist}(z_t, p_t))^{-1}$
$n \leftarrow \frac{1}{\sum_N^{i=1} w_i^2}$
if $n < N/3$ **then**
  $p_t \leftarrow$ resample(w_t, N)
end if
return p_t, w_t

And as mentioned, the inference is done by taking the weighted average of all particles.

3. Results

3.1. Experiment Settings

I experimented on four sets of conditions to evaluate the performance of these two models. For every setting, I evaluated the models based on the Mean Square Error (MSE), Mean Absolute Error (MAE), and Final Error (FE).

For visualization, I will provide (1) a plot show-

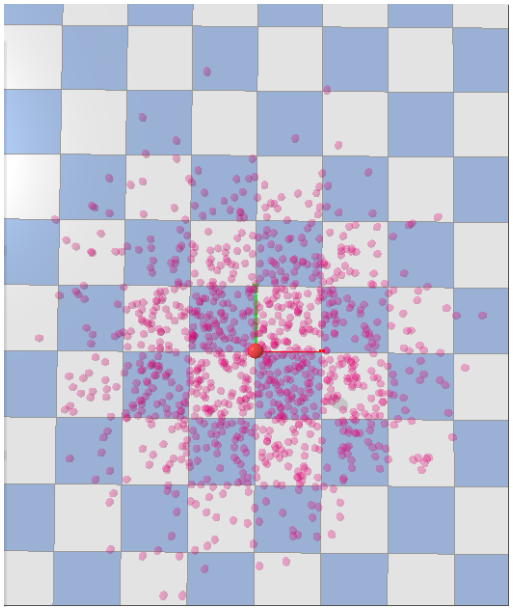

Figure 1. The noise distribution for experiment 1

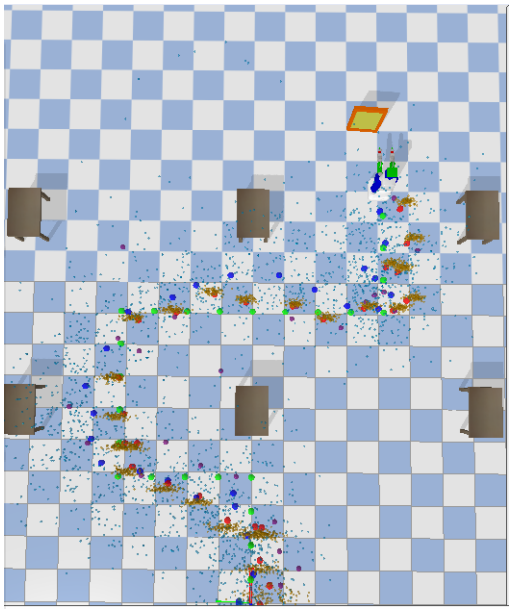

Figure 2. The trajectory for experiment 1

ing the sensor noise distribution in which the orange sphere shows the center of the distribution and pink dots show samples of the distribution, (2) a plot showing the trajectory of localization results in which green spheres shows ground truth for each time step, purple spheres show the noisy sensor data, blue spheres show the estimations of Particle Filter, lightblue dots shows the particles, red spheres show the estimations of Kalman Filter, and orange dots show samples of the Gaussian distribution given by Kalman Filter.

3.2. Experiment 1

In the first experiment I used a Gaussian noise $\mathcal{N}(0, 1)$ as the sensor data 1. It is clear that both estimations provided by Kalman Filter and Particle Filter follows the ground truth trajectory 2 closely since the setting meets the assumption of Kalman Filter and is stable enough for the Particle Filter to follow.

–	Particle Filter	Kalman Filter
MAE	0.90	1.09
MSE	0.70	0.88
FE	0.86	0.94

3.3. Experiment 2

In the second experiment I used a Uniform noise $\mathcal{U}(-1.5, 1.5)$ as the sensor data 3. Since this distribution is similar to Gaussian distribution in the way that they are both symmetric about the origin, both Kalman Filter and Particle Filter give good localization accuracy along the trajectory 4.

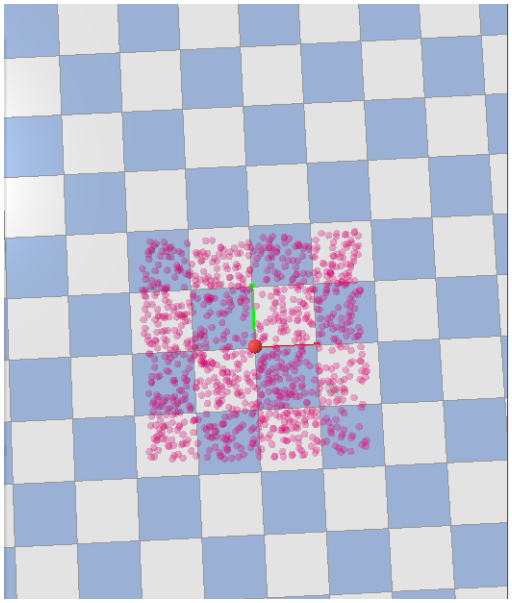

Figure 3. The noise distribution for experiment 2

–	Particle Filter	Kalman Filter
MAE	0.83	0.92
MSE	0.67	0.76
FE	0.80	0.29

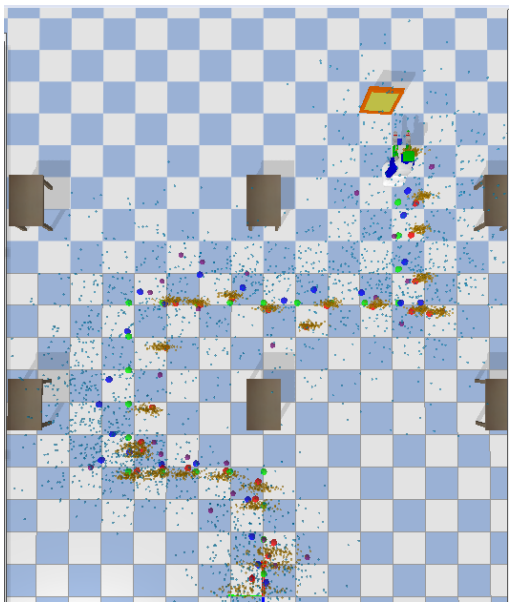

Figure 4. The trajectory for experiment 2

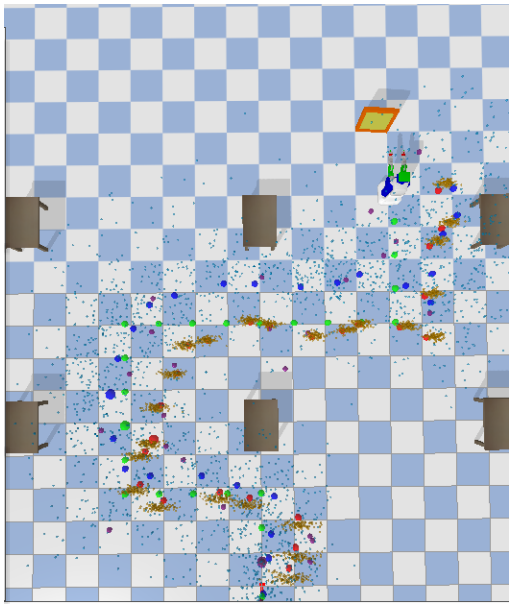

Figure 5. The trajectory for experiment 3

3.4. Experiment 3

To test both algorithms' ability to cope with sudden changes in sensor noise, I designed an experiment with the same noise as Experiement 2 3, but in step 10, I deliberately enlarged the noise (corresponding sphere markers are enlarged in the plot 5). It is clear that Kalman Filter is influenced towards the direction of enlarged noise, but Particle Filter, by comparison, does not show much greater accuracy either.

–	Particle Filter	Kalman Filter
MAE	1.32	1.38
MSE	1.01	1.13
FE	1,77	1.32

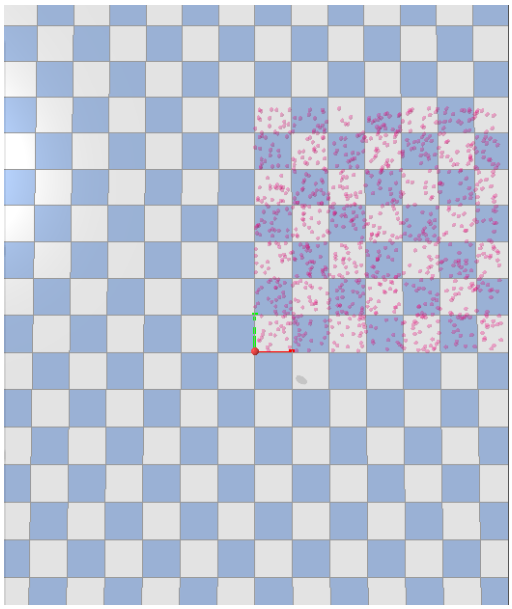

Figure 6. The noise distribution for experiment 4

3.5. Experiment 4

In Experiment 4 I introduced a noise that is not Gaussian and does not resemble the center-symmetric property of Gaussian distribution, which is an Uniform distribution $6\,\mathcal{U}(0,6)$ centered at $(3,3)$ (note that the plot shows the noise distribution is taken by a camera pose rotated from the one taking the trajectory plot). It is indicated from this experiment 7 that Kalman Filter is more vulnerable to skewed noises compared to Particle Filter. The reason of such vulnerability of Kalman Filter comes from its heuristics to "average" the sensor data measurement and prediction. In the contrast, Particle Filter only reweight its particles in events of drastic change of observation relative to its prediction, so it is more probable to make reasonable choice when it encounters such an uneven noise distribution.

–	Particle Filter	Kalman Filter
MAE	1.28	5.15
MSE	0.97	3.76
FE	3.87	3.41

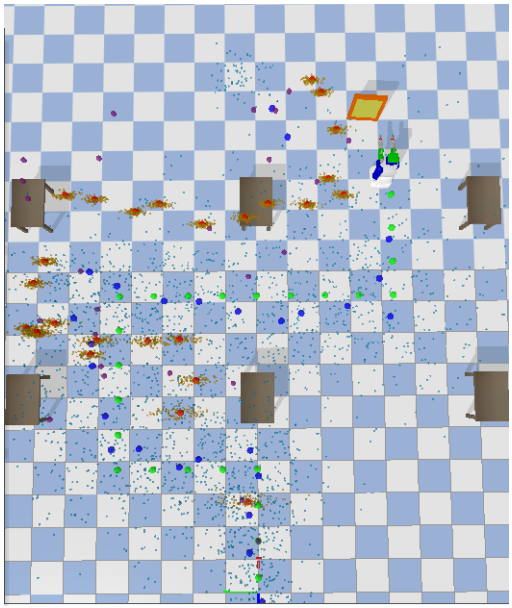

Figure 7. The trajectory for experiment 4

References

[1] R. Faragher. Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]. *IEEE Signal Processing Magazine*, 29(5):128–132, 2012.

[2] S. Godsill. Particle filtering: the first 25 years and beyond. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7760–7764, 2019.

[3] Y. Pei, S. Biswas, D. S. Fussell, and K. Pingali. An elementary introduction to kalman filtering, 2019.