1 Introduction

In this example we introduce the particle filter which is a very powerful filter as it holds superior amongst a general set of conditions. In tutorial 1 it had already been shown that the Kalman Filter produces optimal estimated for unimodal linear systems with Gaussian noise, which is an ideal case and almost non-existent. The EKF has it's own disadvantages such that it does not work for multimodal problems which involve tracking more than 1 object [1]. We begin by taking an alternative route from the Bayes filter, where instead of approximating our probability density functions as Gaussian, we approximate our PDFs using a large number of random samples and perform Monte Carlo integrations as opposed to linearizing the motions and observation models, and evaluating the integrals in closed form.

1.1 Monte Carlo Method

The Monte Carlo method is an alternative to transforming a PDF through a nonlinearity, and is also known as the "brute force" approach [2].The advantages of this method are that it works with and PDF, not just Gaussian, we do not need to know the mathematical form of the nonlinear function, and it handles any type of nonlinearity. As we know a larger amount of samples will allow us to converge to the correct answer due to the law of large number, this contributes to the main disadvantage of the method: Requires alot of memory, and can be inefficient in higher dimensions.

1.2 Particle Filter

The particle filter is one of the only practical techniques able to handle non-Gaussian noise and nonlinear observation and motion models. This method does enables us to solve problems without requiring analytical expressions for f(·) and g(·), nor for their derivatives. Particle filters have many versions, the method we introduce in this tutorial is also known as the sample importance resampling technique. This method involves us to find the desired posterior distribution by propagating the prior PDFs (initial weights) forward using the motion model and latest motion measurements. The first step requires us to draw M samples from the joint density comprising the prior and the motion noise.

$$\begin{bmatrix} \hat{\mathbf{x}}_{k-1,m} \\ \mathbf{w}_{k,m} \end{bmatrix} \leftarrow p\left(\mathbf{x}_{k-1} \mid \check{\mathbf{x}}_0, \mathbf{v}_{1:k-1}, \mathbf{y}_{1:k-1}\right) p\left(\mathbf{w}_k\right) \tag{1}$$

After this we generate the prediction of the posterior PDF by using:

$$\check{\mathbf{x}}_{k,m} = \mathbf{f}\left(\hat{\mathbf{x}}_{k-1,m}, \mathbf{v}_k, \mathbf{w}_{k,m}\right) \tag{2}$$

This equations produces the posterior belief of the current state by including the most recent process input, but no measurements yet.

Then we use Bayes theorem such that when a new measurement comes in we multiply the current probability of the current state (the current normalized

weights) which in our case is the prior belief of each particle, by the likelihood that the measurement matched that location:

$$P(\check{\mathrm{x}}_{k,m} \mid \mathbf{y}_k) = \frac{P(\mathbf{y}_k \mid \check{\mathrm{x}}_{k,m}) P(\check{\mathrm{x}}_{k,m})}{P(\mathbf{y}_k)} \tag{3}$$

We then resample using the systematic resample technique. Particles with very small weights do not meaningfully describe the probability distribution of the robot. The resampling algorithm discards particles with very low probability and replaces them with new particles with higher probability. Particles with relatively high probability are duplicated.

$$\hat{\mathbf{x}}_{k,m} \overset{\text{resample}}{\longleftrightarrow} \{\check{\mathbf{x}}_{k,m}, w_{k,m}\} \tag{4}$$

We also compute the estimate mean as the sum of the weighted values of the particles.

$$\mu = \frac{1}{N} \sum_{i=1}^{M} w^i x^i \tag{5}$$

It should be noted that there are several different ways to resample and the systematic technique is just one of many. We don't sample at every iteration, we use a indicator also known as the effective M. If the effective M value is below a certain threshold, we can deduce that the number of particles which meaningfully contribute are low and thus it is time for a resampling to occur. The equation for this is:

$$\hat{M}_{\text{eff}} = \frac{1}{\sum w^2} \tag{6}$$

2 Implementation and Results

In this section our robot is assumed to move throughout the trajectory, based on the nonlinear bicycle model approach introduced in the Extended Kalman Filter example. A thousand particles were generated with each component of their pose (x,y, and θ), being normally distributed with mean $[10, 0, 0]$ and standard deviation of $[5, 5, \pi/4]$. This initial normal pose distribution can be expressed by the green points in Figure 1. The characteristics of the robot in this example includes: wheelbase of 0.5 m, steering angle of 0.0499 radians, velocity of 5 m/s, standard deviation of velocity as 0.05, standard deviation of the steering angle as 0.2, the total standard deviation of the sensor is 0.1. In this example we also include landmarks located at $[10, 10], [5, 10], [12, 14]$. After approximately 12.57 seconds our robot will our robot will move thorough one full circle with radius of 10.01, given the fact that it's velocity is 5 m/s, and it's radius given as $R = \frac{w}{\tan(\alpha)}$, where w is the wheelbase and α is the steering angle as mentioned earlier. Our implementation simulated the robots trajectory for 12.5 seconds, where an prediction takes place 8 times per second and the update incorporating the measurement model takes place 1 time per second.

The update step involves redistributing the weights of each particle according to which how well they match the measurements obtained at that instance and are computed used Bayes theorem. as mentioned in the previous section. The mean and the variance of the x and y poses are also computed for each prediction and update. A snippet of the animation is shown in Figure 4. The solid red line trajectory represents the predicted trajectory without the measurements, and dotted green line represents the corrected trajectory of the robot. As illustrated in Figure 2, the spread of the particles increase up until the measurement update is included, and the cycle continues. The measurement update occurs once every 8 times as mentioned before, from which we can see in Figure 2, after 8 red squares (estimates), the update occurs and the particle are now more closer together. Therefore, it is safe to assume that the variance decreases once the measurement is included which was also illustrated by the ellipses in the EKF and KF examples. For the animation in Figure 4, the landmarks are included, and the positions are scaled to fit the pixel frame (700,500).

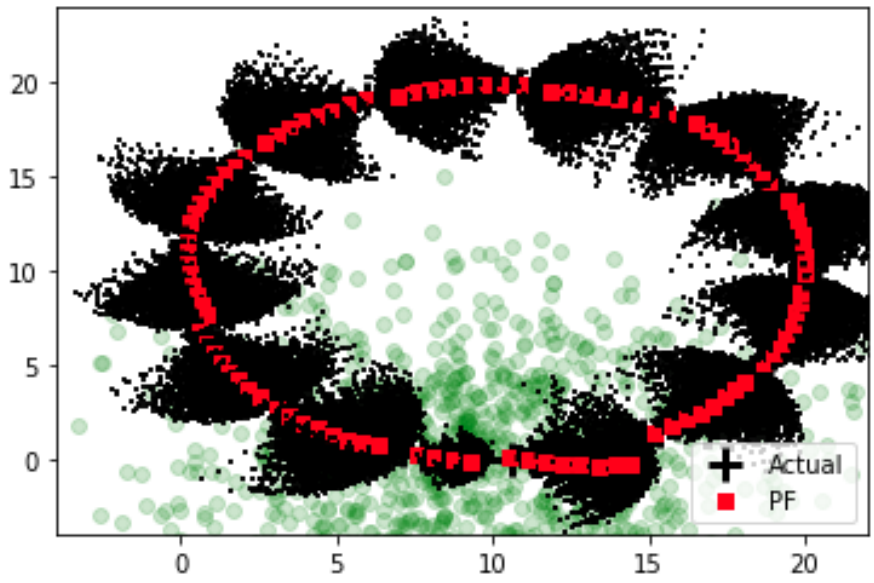

Figure 1: Particle Filter Graphical Trajectory

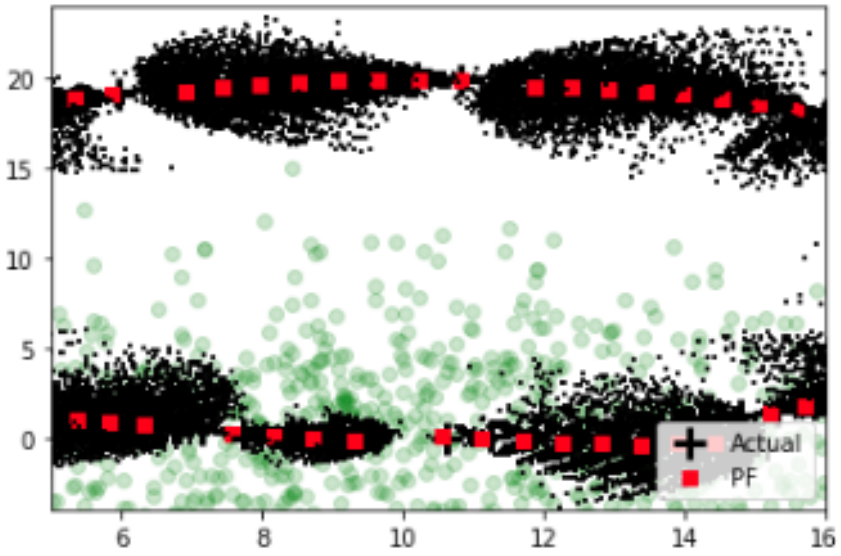

Figure 2: Zoom in on the Graphical Trajectory

```
final position error, variance:
        [-0.294794  -0.1085584] [0.06597266 0.33598771]
```

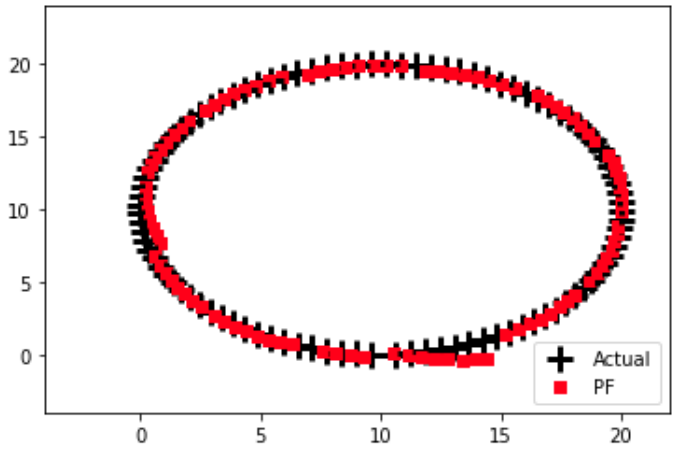

Figure 3: Particles are hidden. Plot of ground truth and mean positions

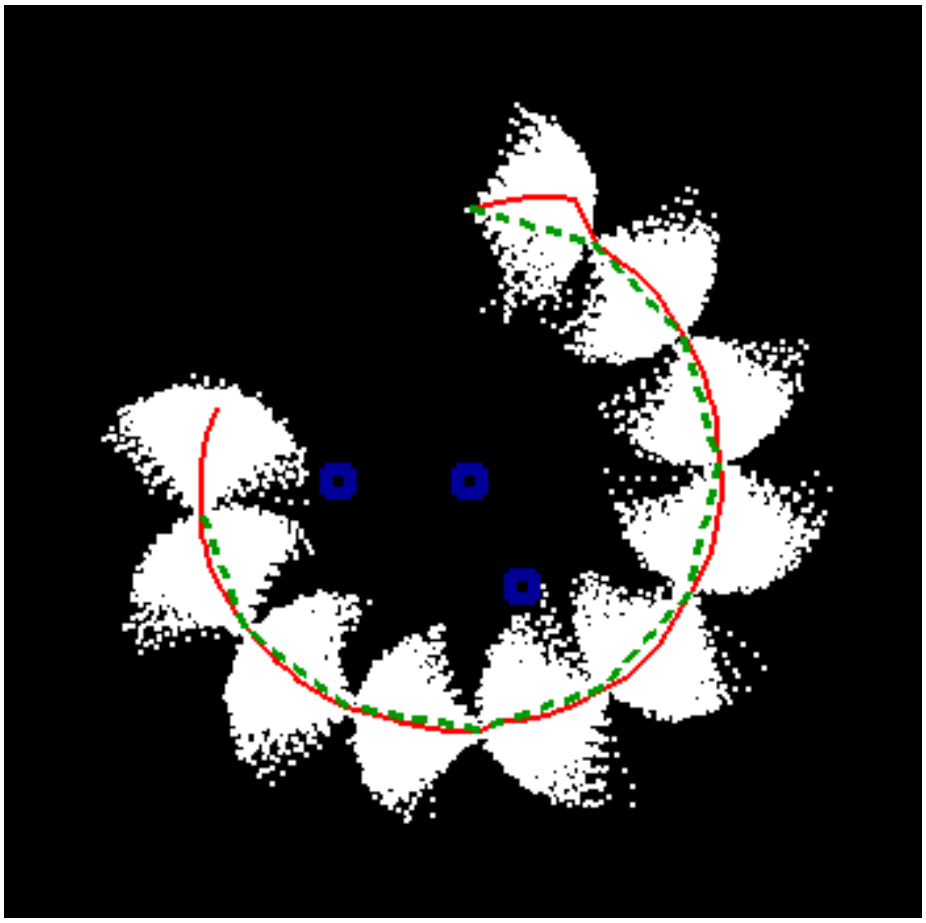

Figure 4: Snippet Of Animation

References

[1] R. R. L. Jr, *Kalman and Bayesian Filters in Python*, 2020.

[2] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2020.