Task 2 - Localisation

Jacob Johansson*, Mudar Ibrahim†
School of Innovation, Design and, Engineering
Mälardalens University, Västerås, Sweden
Email: *jjn20030@student.mdu.se, †mim20004@student.mdu.se

I. Extended Kalman Filter

A. Prediction equations

State prediction equation was done on a differential drive robot:

$$x(k+1) = x(k) + \left(\frac{\Delta s_r + \Delta s_l}{2} + v_d\right)\cos(\theta(k))$$
$$y(k+1) = y(k) + \left(\frac{\Delta s_r + \Delta s_l}{2} + v_d\right)\sin(\theta(k))$$
$$\theta(k+1) = \theta(k) + \frac{\Delta s_r - \Delta s_l}{l} + v_\theta$$

where $v_d \sim \mathcal{N}(0, \sigma_d^2 = 0.25)$, $v_\theta \sim \mathcal{N}(0, \sigma_\theta^2 = 0.03)$, and l = 0.1m.

State prediction uncertainty is calculated by the following: $\boldsymbol{P(k+1)} = \mathbf{F}_x \cdot \text{P(k)} \cdot \text{F}_x^T + \mathbf{F}_v \cdot \text{V} \cdot \text{F}_v^T$

where

$$\mathbf{F}_x = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\left(\frac{\Delta s_x + \Delta s_y}{2} + v_s\right)\sin(\theta(k)) \\ 0 & 1 & \left(\frac{\Delta s_x + \Delta s_y}{2} + v_s\right)\cos(\theta(k)) \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

and

$$\mathbf{F}_v = \begin{bmatrix} \frac{\partial f_1}{\partial v_d} & \frac{\partial f_1}{\partial v_\theta} \\ \frac{\partial f_2}{\partial v_d} & \frac{\partial f_2}{\partial v_\theta} \\ \frac{\partial f_3}{\partial v_d} & \frac{\partial f_3}{\partial v_\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta(k)) & 0 \\ \sin(\theta(k)) & 0 \\ 0 & 1 \end{bmatrix} \tag{2}$$

B. Correction Equations

The innovation equation is given by:

$$\mathbf{V} = \mathbf{Z}(k+1) - \mathbf{h}(k+1) \tag{3}$$

where $\mathbf{Z}(k+1)$ is a 12×1 matrix representing measured sensor values to landmarks, and $\mathbf{h}(k+1)$ is a 12×1 matrix representing predicted measured sensor values to landmarks, which is calculated as:

$$\mathbf{h}_i = \begin{bmatrix} \sqrt{(y_i - y)^2 + (x_i - x)^2} + w_r \\ \text{atan2}(y_i - y, x_i - x) - \theta(k+1) + w_\beta \end{bmatrix}$$

where $w_r \sim \mathcal{N}(0, \sigma_r^2 = 0.25)$, $w_\beta \sim \mathcal{N}(0, \sigma_\beta^2 = 0.03)$, i = 1..12 landmarks

The state prediction correction equation is:

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} + \mathbf{K} \cdot \mathbf{V} \tag{4}$$

where $\mathbf{K} = \mathbf{P}(k+1) \cdot \mathbf{H}_x^T \cdot (\mathbf{H}_x \cdot \mathbf{P}(k+1) \cdot \mathbf{H}_x^T + \mathbf{H}_w \cdot \mathbf{W} \cdot \mathbf{H}_x^T)^{-1}$.

$\mathbf{K}$ is the Kalman Gain with the following parameters:

$$\mathbf{H}_x = \frac{\partial h}{\partial x} = \begin{bmatrix} -\frac{x_i - x}{r_i} & -\frac{y_i - y}{r_i} & 0 \\ \frac{y_i - y}{r_i^2} & -\frac{x_i - x}{r_i^2} & -1 \end{bmatrix}$$

, where i = 1..12 landmarks and r_i is the distance between the estimated robot's position and landmark i.

$$\mathbf{H}_w = \frac{\partial h}{\partial w} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{is a } 12 \times 12 \text{ identity matrix.}$$

State prediction uncertainty correction equation is given by: $\boldsymbol{P(k+1)} = \boldsymbol{P(k+1)} - \boldsymbol{K} \cdot \boldsymbol{H_x} \cdot \boldsymbol{P(k+1)}$.

In order to stabilize the system, the *atan2* function was used. The result of the predicted robot's pose and the trajectory is shown in figure 1 and 2.

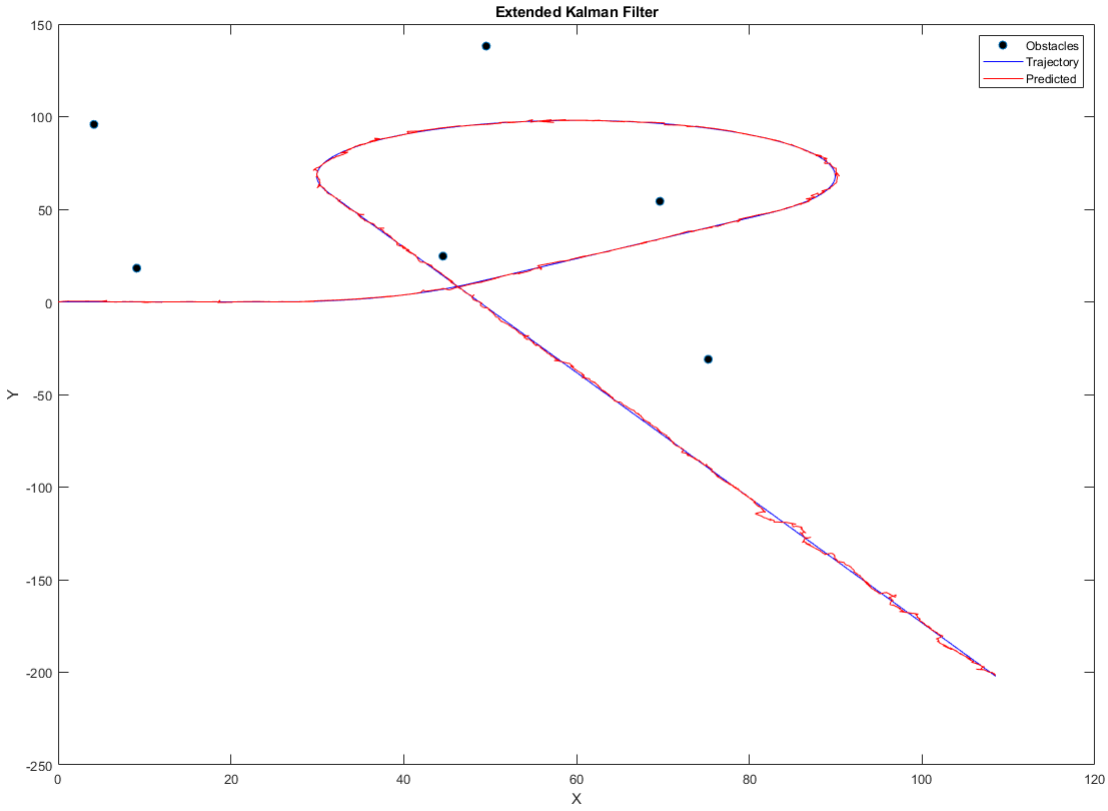

Figure 1. Image of the robot's x-y position vs. the trajectory.

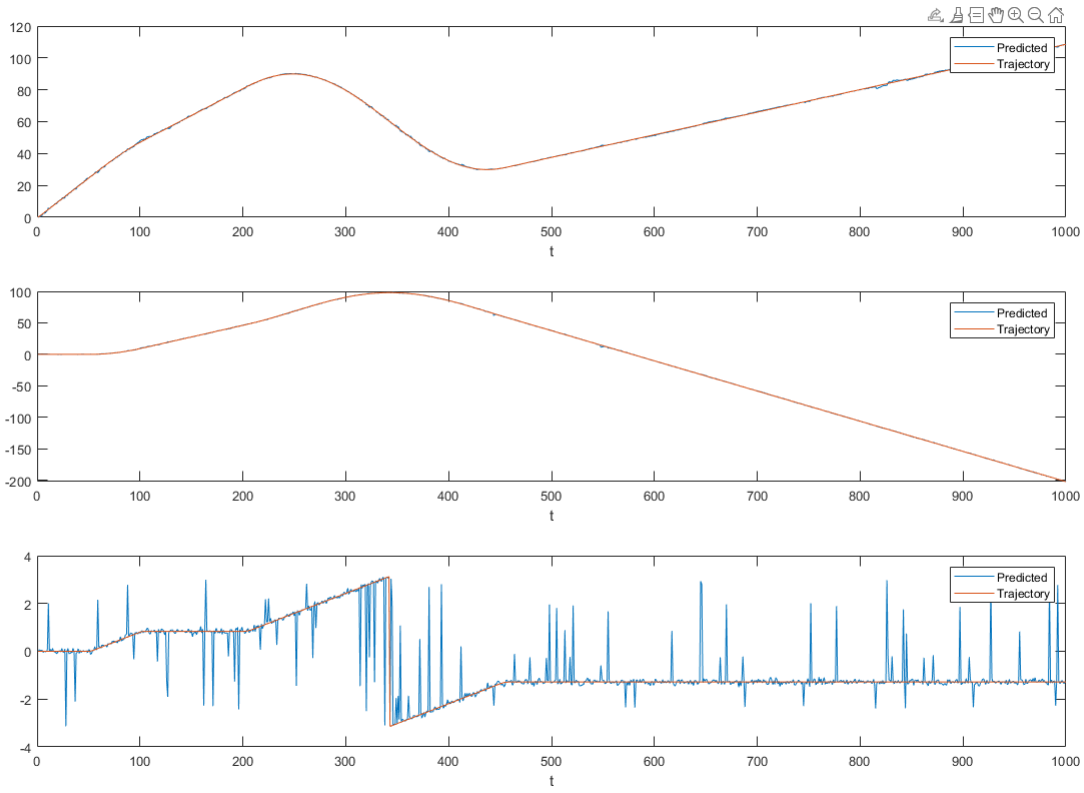

Figure 2. Image of the robot's pose vs. the trajectory.

II. Particle Filter

The implementation of the particle filter is done in *6* steps:

Step 1. Initialization of Particles and the Robot

Due to the robot's pose being unknown initially, all the particles are uniformly distributed across the map, with $x_{\text{min}} = 0$, $x_{\text{max}} = 120$, $y_{\text{min}} = -250$, $y_{\text{max}} = 150$, $\theta_{\text{min}} = -\pi$, and $\theta_{\text{max}} = \pi$. The weight for each particle is initialized as:

$$w_i = \frac{1}{N}$$

where N is the total number of particles.

Step 2. Prediction of the pose of each particle

For predicting the pose of each particle, the following motion model was used:

$$\begin{aligned} x(k+1) &= x(k) + (\delta_d + r_d) \cdot \cos\theta(k) \\ y(k+1) &= y(k) + (\delta_d + r_d) \cdot \sin\theta(k) \\ \theta(k+1) &= \theta(k) + (\delta_\theta + r_\theta) \end{aligned}$$

where $r_d \sim \mathcal{U}(-1, 3)$ and $r_\theta \sim \text{Logistic}(\mu_\theta = 0.5236, s_\theta = 0.7854)$.

Step 3. Update Weights of the particles

The measurement model $\mathbf{h}$ is calculated as before, but with $w_d \sim \text{Logistic}(0, s_d = 1)$ and $w_\beta \sim \text{Logistic}(0, s_\beta = 0.349)$. Due to the measurement noise being distributed logistically, the weight for each particle was calculated as follows:

$$w_i = \left[\prod_{i=1}^{N} \text{L}(\boldsymbol{z} - \boldsymbol{h}, \mu, s_d) \cdot \text{L}(\boldsymbol{z} - \boldsymbol{h}, \mu, s_\beta) \right]$$

where L is the logistic probability distribution function, $\boldsymbol{z}$ is the actual measured distance and angle between the robot and all the landmarks, $\boldsymbol{h}$ is the distance and angle between each particle and all the landmarks, w_i is the calculated weight for particle i, and N is the total number of particles. The parameters are $\mu = 0$, $s_d = 1$ m, and $s_\beta = 0.349$ rad.

Step 4. Normalizing the weights

The normalization of the weights is done by the equation:

$$w_i = \frac{w_i}{\sum_{n=1}^{N} w_i}$$

Step 5. Estimate Robot's Pose

Estimating the robot's position is done by calculating the weighted average of all the positions of the particles, while the robot's angle is calculated by finding the weighted average of the particle angles on a unit circle:

$$\begin{aligned} \hat{\boldsymbol{X}} &= \sum_{n=1}^{N} w_i \cdot x_i \\ \hat{\boldsymbol{Y}} &= \sum_{n=1}^{N} w_i \cdot y_i \\ \hat{\boldsymbol{\theta}} &= \text{atan2}\left(\sum_{n=1}^{N} w_i \cdot \sin\theta_i, \sum_{n=1}^{N} w_i \cdot \cos\theta_i \right) \end{aligned}$$

Step 6. Resampling the Best Particles

The low variance resampling algorithm was used to resample the best particles, as shown in Figure 3.

Algorithm 1 Low variance re-sampling $(\mathbf{x}, \mathbf{w})$

$\mathbf{x}_{k+1} = \emptyset$
$r = rand(0; N^{-1})$, $c = w_1$, $j = 1$
for $i = 1$ to N **do do**
  $U = r + (i-1) \cdot N^{-1}$
  while $U > c$ **do**
    $j = j + 1$
    $c = c + w_j$
  end while
  add x_j to $\mathbf{x}_{k+1}$
end for
Return $\mathbf{x}_{k+1}$

Figure 3. The algorithm used for the re-sampling.

A comparison between 100, 500, 1000, and 3000 particles was done. The result is shown in Figure 8 and Figure 9. The execution time is shown in Figure 4, Figure 5, Figure 6, and Figure 7. The result clearly demonstrate that the more particles, the better estimation of the position. But after 1000 particles, the increased execution time may not be worth the minimal improvement.

```
Elapsed time is 1.424194 seconds.
```

Figure 4. Execution time with 100 particles.

```
Elapsed time is 1.520810 seconds.
```

Figure 5. Execution time with 500 particles.

```
Elapsed time is 1.789023 seconds.
```

Figure 6. Execution time with 1000 particles.

```
Elapsed time is 2.507152 seconds.
```

Figure 7. Execution time with 3000 particles.

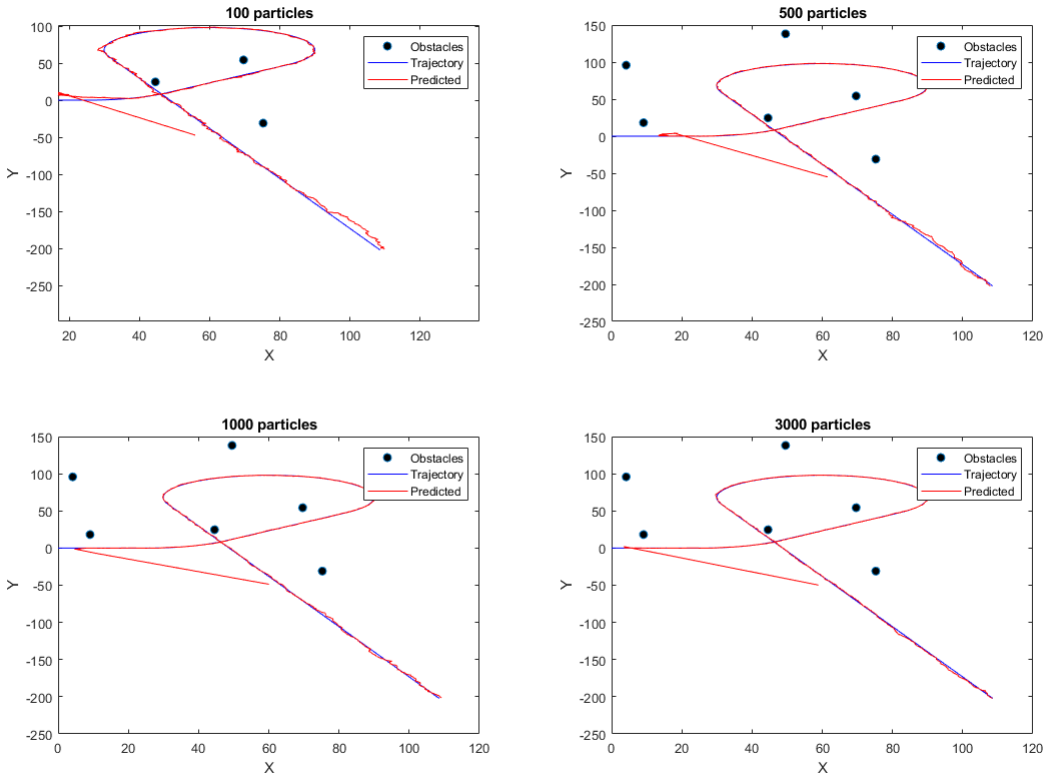

Figure 8. Plot of the Particle filter vs. Trajectory.

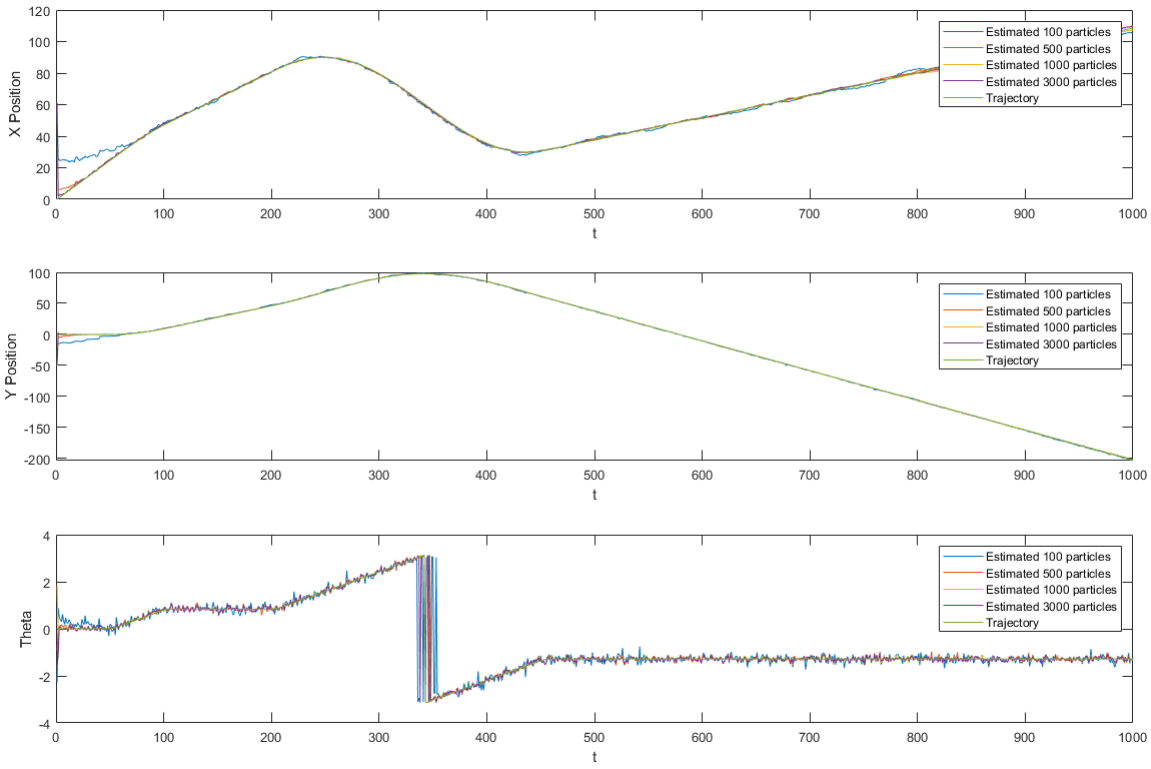

Figure 9. Plot of the Particle filter vs. Trajectory, each state component individually.

REFERENCES