AI IN CONTROL ENGINEERING

2nd exercise

Lecturer: Dr.Pham Viet Cuong

Group 1

Luong Huu Phu Loc	1511844
Pham Ngoc Khoi Nguyen	1512221
Bui Tan Phat	1512396
Nguyen Trong Phuc	1512534
Mai Thien Quang	1512640

1. Problem

- Drawing trajectories of robot base on data from XTRUE, XODO and value of 3 particles which include the max, medium and min using particle filter.
- Computing RMS error of sub-trajectory to the real.
- Using EKF, draw XTRUE, XODO and posterior estimated trajectory as well as compute RMS error toward the real.

2. Parameter

- Before implementing paricle filter or EKF, some figures need choosing:
  - Number of particle: 100
  - Wb = 4
- Other parameters are given before hand, such as deviation of velocity, steering angle, range, bearing, time step, control signal (VG) and landmark's location. From that, we can infer covariance matrix and initiate correct size of result matrixes.
- According to given control signal matrix and deviation of velocity, steering angle, we can easily create new control signal adding Gaussian noise. That becomes variable for process model.
- One more vital thing to consider is that the control frequency's faster than measurement frequency (40Hz and 5Hz). So the internal time between 2 control signal is Δt = 0.025.

3. Particle Filter

3.1. Workflow

- Prediction
  - First, initiate the beginning state of particles, in this work, we choose the pose of 100 particles at the same point XODO(:,1).
  - At every time step (from 2 to 625), we need to calculate predicting pose, base on corresponding velocity (v) and steering angle (s) in VG matrix, which were added Gaussian noise.

$$\begin{bmatrix} v \\ s \end{bmatrix} = \begin{bmatrix} VG(1,i) \\ VG(2,i) \end{bmatrix} + \begin{bmatrix} \sigma_v \, 0 \\ 0 \quad \sigma_s \end{bmatrix} \begin{bmatrix} rand(1) \\ rand(1) \end{bmatrix}$$

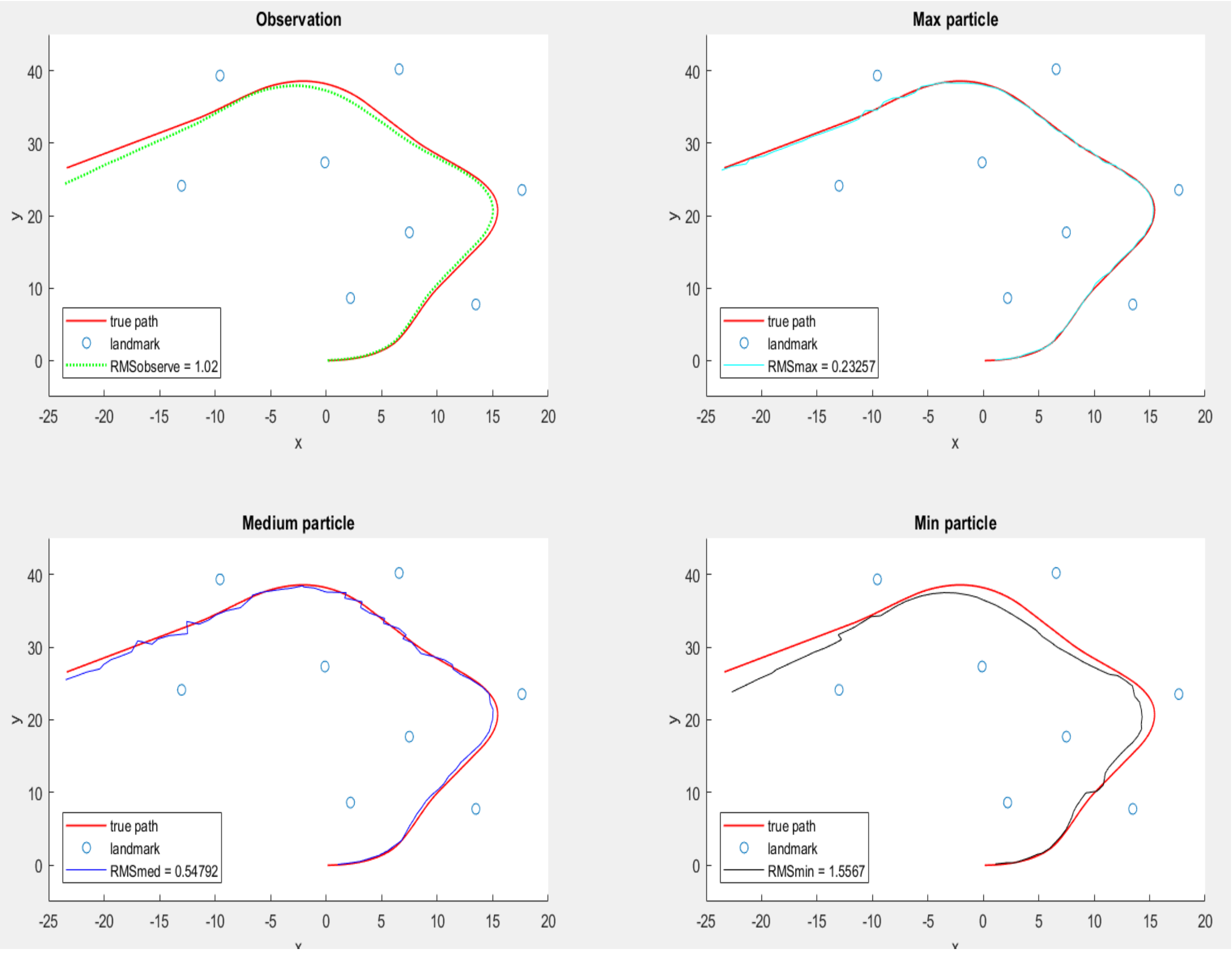
  - This methos is used to generate random pose of particles, a loop with 100 step can make 100 different control signal allowing distributing random particles according to the last position.
  - Second, check the signal from laser sensor whether it has or not. If there isn't, simply update the posterior pose of all particles. If there is, combine coordinate of robot from last step, position of landmarks in **lm** matrix to calculate range r_t and $b_{t.}$ Then compute the weight of each particle by following formula:

- (With N is number detected landmarks, η is normalization factor and Σ is covariance matrix of range and bearing)

$$w_i = P(z|X_i) = \eta \prod_{k=1}^{N} \frac{1}{\det(2\pi\Sigma)^{\frac{1}{2}}} \exp[-0.5(z-X_i)^T\Sigma^{-1}(z-X_i)]$$

- Resampling
  - Using Roulett method to resampling particles.
  - In that, implement a loop with 100 step, a cummulative sum of weight is calculated before normalizing them to interval [0 1] and generating random number in that interval. Choose the nearest value to random number with correlative index.
- Compute RMS and draw graph

3.2. Result



- Comment:
  - Only with process model, it's hard to get true position of robot. Despite only 625 time steps, too short to realize the error, we can easily see the mischasing of XODO path to real path.

- With particle filter, trajectory of the max particle is 5 times corrected than process model (rely on RMS). When decreasing the weight, RMS increases as the path of medium and min particle spread out the true path.

4. Extended Kalman Filter

- Since the process model is non-linear, but noise is normal distribution (Gauss), we can't use normal Kalman filter. So, Extended Kalman Filter (EKF) is some of good choices.

  4.1 Workflow

- From process model and observation model, it can be infer the linearzation matrix G and H.

$$\begin{cases} \chi_t = g(\chi_{t-1}, u_t) + w_t \\ z_t = h(\chi_t) + v_t \end{cases}$$

With:

$$g(\chi_{t-1}, u_t) = \begin{bmatrix} x_{t-1} + v_t.\Delta t.\cos(\varphi_{t-1} + \theta_t) \\ y_{t-1} + v_t.\Delta t.\sin(\varphi_{t-1} + \theta_t) \\ \varphi_{t-1} + \dfrac{v_t.\Delta t.\sin(\theta_t)}{wb} \end{bmatrix}$$

$$h(\chi_t) = \begin{bmatrix} \sqrt{(x_t - x_{lm})^2 + (y_t - y_{lm})^2} \\ atan2(y_t - y_{lm}, x_t - x_{lm}) - \varphi \end{bmatrix}$$

Infer:

$$G_t = \begin{bmatrix} 1 & 0 & -v.\Delta t.\sin(\varphi_{t-1} + \theta) \\ 0 & 1 & v.\Delta t.\cos(\varphi_{t-1} + \theta) \\ 0 & 0 & 1 \end{bmatrix}$$

$$H_t = \begin{bmatrix} \dfrac{x_t - x_{lm}}{\sqrt{(x_t - x_{lm})^2 + (y_t - y_{lm})^2}} & \dfrac{y_t - y_{lm}}{\sqrt{(x_t - x_{lm})^2 + (y_t - y_{lm})^2}} & 0 \\ \dfrac{-(y_t - y_{lm})}{(x_t - x_{lm})^2 + (y_t - y_{lm})^2} & \dfrac{x_t - x_{lm}}{(x_t - x_{lm})^2 + (y_t - y_{lm})^2} & -1 \end{bmatrix}$$

- About the noise w_t and v_t, they are both Gaussian noise with relative covariance:

$$Q_t = G_u \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} G_u^T = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\varphi^2 \end{bmatrix}$$

$$R_t = \begin{bmatrix} \sigma_{range}^2 & 0 \\ 0 & \sigma_{bearing}^2 \end{bmatrix}$$

With:

$$G_u = \frac{\partial g}{\partial u} = \begin{bmatrix} T.\cos(\varphi_{t-1}+\theta_t) & -v_{t.}T.\sin(\varphi_{t-1}+\theta_t) \\ T.\sin(\varphi_{t-1}+\theta_t) & v_{t.}T.\cos(\varphi_{t-1}+\theta_t) \\ \dfrac{T.\sin(\theta_t)}{wb} & \dfrac{v_{t.}T.\cos(\theta_t)}{wb} \end{bmatrix}$$

That means, at every time step, the covariace matrix Q_t change upon the change of v_t and θ_t.

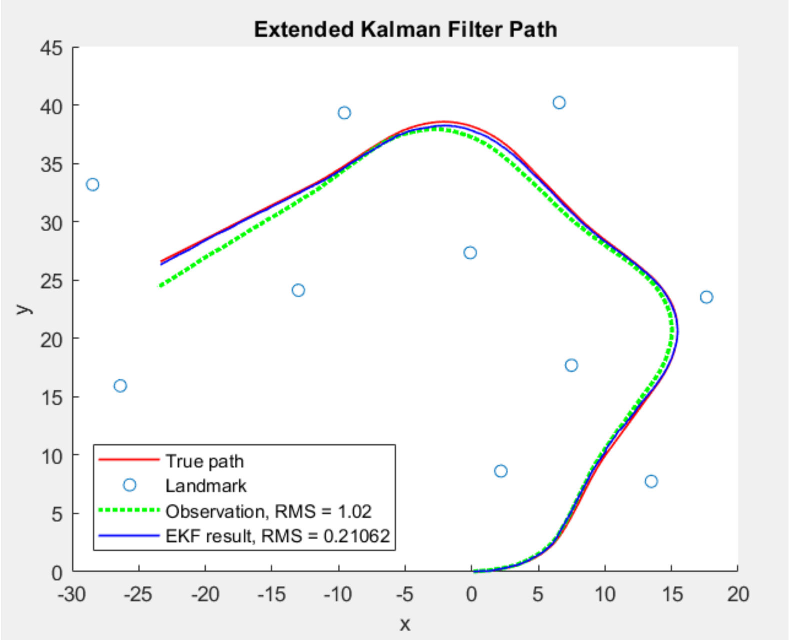
- Just like particle filter, next pose of robot is calculated base on process model and noise control signal VG.
- Apply the discrete model of EKF with average method

$$\begin{cases} \bar{\chi}_t = predict\_model(v_t, \theta_t, \Delta t, \chi_{t-1}) \\ \bar{P}_t = G_t P_{t-1} G_t^T + Q_t \\ K_t = \bar{P}_t.\left[\dfrac{1}{N}\displaystyle\sum_{k=1}^{N} H_{k,t}^T \left(H_{k,t}\bar{P}_t H_{k,t}^T + R_t\right)^{-1}\right] \\ \chi_t = \bar{\chi}_t + K_t.\dfrac{1}{N}\displaystyle\sum_{k=1}^{N}\left[z_{k,t} - h_k(\bar{\chi}_t)\right] \\ P_t = \left[\dfrac{1}{N}\displaystyle\sum_{k=1}^{N}\left(I - K_t H_{k,t}\right)\right].\bar{P}_t \end{cases}$$

The reason is data from sensor is not always same size (range 0 -> 5), so we take average the Kalman gain and error between true measurement and calculating measurement.

- Compute RMS and draw graph

4.2 Result



- Comment: EKF performs quite well with RMS is low (lower than particle filter). At straight path, EKF’s path follows closely the real, but at some corner or round path we still see mischasing error.

5. Summary

- Although Particle Filter is built for non-linear system and non-Gaussian noise, it’s still applied for robot’s position prediction with high performance. However, max particle doesn’t have 100% weight, which mean the real RMS is higher.
- Extended Kalman Filter in this case is better than Particle Filter (it’s specified for this situation).
- Nervertheless, random variable makes both filter result change a little bit everytime we run.