State Estimation - Assignment 2

Haniyeh Altafi

August 17, 2023

1 Question 1 - Kalman Filter

Use pyGame, or any other similar libraries, to simulate a simplified 2D robot and perform state estimation using a Kalman Filter. Motion Model:

$$\dot{x} = \frac{r}{2}(u_r + u_l) + w_x \quad , \quad \dot{y} = \frac{r}{2}(u_r + u_l) + w_y$$

r = 0.1 m, is the radius of the wheel, u_r and u_l are control signals applied to the right and left wheels. $w_x = N(0, 0.1)$ and $w_y = N(0, 0.15)$

Simulate the system such that the robot is driven 1 m to the right. Assume the speed of each wheel is fixed and is 0.1 m/s

Use these initial values

$$x_0 = 0, y_0 = 0, P_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ (initial covariance matrix)}$$

and assume the motion model is computed 8 times a second. Assume every second a measurement is given:

$$z = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} r_x & 0 \\ 0 & r_y \end{bmatrix}$$

where $r_x = N(0, 0.05)$ and $r_x = N(0, 0.075)$

Answer :

Below is my explanation of the Kalman Filter and its associated algorithm:

The Kalman Filter is a mathematical tool used for estimating the state of a dynamic system based on a series of measurements. It is widely employed in various fields, including control systems, robotics, and signal processing.

The Kalman Filter algorithm consists of the following steps:

1. Initialization: At the start, the initial state of the system and its uncertainty are defined. This includes the system's position, velocity, and any other relevant variables. Additionally, the covariance matrix, representing the uncertainty in the initial state, is also initialized.

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \mu_0$$
$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$B = \begin{bmatrix} \frac{r}{2}\delta t & \frac{r}{2}\delta t \\ \frac{r}{2}\delta t & \frac{r}{2}\delta t \end{bmatrix}$$
$$\Sigma_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$
$$r = 0.1$$
$$\delta_t = \frac{1}{8}$$
$$U = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$
$$R = \begin{bmatrix} \omega_x \delta t & 0 \\ 0 & \omega_y \delta t \end{bmatrix} = \begin{bmatrix} 0.1\delta t & 0 \\ 0 & 0.15\delta t \end{bmatrix}$$
$$Q = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.075 \end{bmatrix}$$
$$z = C \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} r_x \\ r_y \end{bmatrix}$$

2. Prediction: In this step, the Kalman Filter predicts the next state of the system based on the previous state and the system dynamics. The prediction incorporates information about how the system is expected to evolve over time, such as its velocity and acceleration. The uncertainty in the prediction is also updated using the covariance matrix.

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t + R_t t$$
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R_t$$

3. Measurement Update: Once a measurement of the system is obtained, the Kalman Filter performs a measurement update. This step combines the predicted state with the actual measurement to refine the estimate of the system's state. It calculates the Kalman Gain, which determines the weight given to the predicted state and the measurement. The updated state and the reduced uncertainty in the estimation are then obtained.

$$K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t C_t) \Sigma_t$$

4. Repeat: Steps 2 and 3 are repeated iteratively for each new measurement received. The Kalman Filter continuously refines its estimate by incorporating new measurements and updating the state and uncertainty.

The Kalman Filter algorithm aims to provide the best estimate of the system's state by minimizing the mean squared error between the estimated state and the true state. It

takes into account both the prediction based on the system dynamics and the measurement information to optimize the estimation. [1]

Here, is the plotted trajectory of the robot.

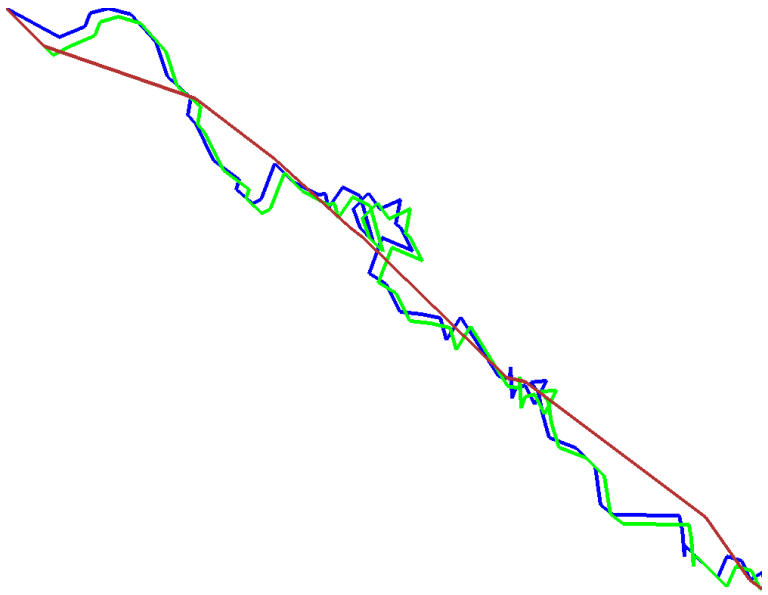

Figure 1: Trajectory of the Robot with KF

2 Question 2 - Extended Kalman Filter

Repeat the previous assignment, this time with a classic motion model and range observations made from a landmark located at $M = [10, 10]$. L is the distance between the wheel, known as wheelbase, and is 0.3 m.

$$\dot{x} = \frac{r}{2}(u_r + u_l)\cos(\theta) + w_x \quad , \quad \dot{y} = \frac{r}{2}(u_r + u_l)\sin(\theta) + w_y, \quad \dot{\theta} = \frac{r}{L}(u_r - u_l).$$

Assume

$$u_\omega = \frac{1}{2}(u_r + u_l), \quad u_\psi = (u_r - u_l)$$

Then the equations become:

$$\dot{x} = ru_\omega \cos(\theta) + w_\omega \quad , \quad \dot{y} = ru_\omega \sin(\theta) + w_\omega, \quad \dot{\theta} = \frac{r}{L}u_\psi + w_\psi$$

$w_\psi = N(0, 0.01)$ and $w_\omega = N(0, 0.1)$. Program the robot such that it loops around point M.

(a) Compute the EKF with the linear measurement model in the previous assignment.

Answer: To rotate the car around a landmark, control signals must be applied to its wheels. For left rotations, only the left wheel signal should be applied. The rest of the formulation remains unchanged. By incorporating observations, the state covariance decreases, and the estimated positions move more smoothly.

1. Initialization:

Initialize the initial state estimate, error covariance matrix, process noise covariance, and measurement noise covariance.

$$\begin{bmatrix} \hat{x}_{k+1} \\ \hat{y}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \hat{y}_k \\ \hat{\theta}_k \end{bmatrix} + \begin{bmatrix} r\Delta t \cos(\theta) & 0 \\ r\Delta t \sin(\theta) & 0 \\ 0 & \frac{r}{l}\Delta t \end{bmatrix} \begin{bmatrix} u_r \\ u_l \end{bmatrix} + w_n \Delta t$$

The covariance matrix will be:

$$P_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

2. Prediction Step:

Predict the state estimate using the nonlinear system dynamics function and linearize it using the Jacobian matrix. Update the error covariance based on the linearized dynamics and process noise.

$$\bar{\mu}_t = f(\hat{x}_{k-1}, v_k, 0)$$
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R_t$$

3. Update Step:

Obtain a measurement and compute the measurement residual. Linearize the measurement model using the Jacobian matrix. Compute the Kalman gain, update the state estimate, and update the error covariance.

$$K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t(z_t - g(\bar{\mu}_t))$$
$$\Sigma_t = (I - K_t C_t)\Sigma_t$$

4. Repeat Steps 2 and 3 for each subsequent time step.

Figure 3 is the plotted trajectory of the robot.

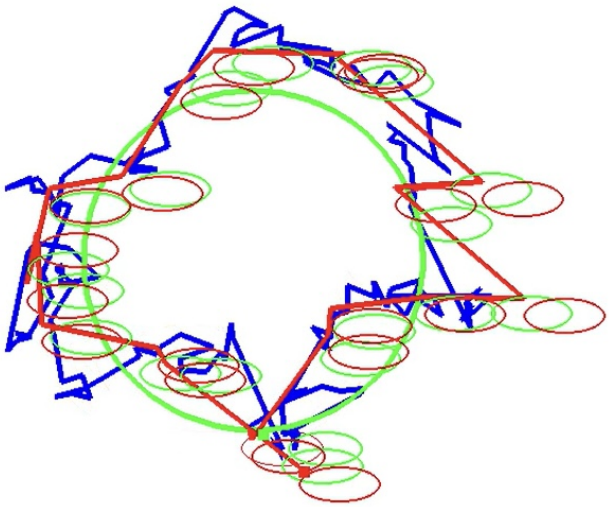

Figure 2: Trajectory of the Robot with EKF

(b) Compute the EKF with range/bearing measurements of point M. Assume range noise is $N(0, 0.1)$ and bearing noise is $N(0, 0.01)$. Range is in meters, and bearing is in radians. Visualize the measurements as well.

Answer: For this part, the measurement probability changes nonlinearly. Polar coordinates are determined by the robot's distance and angle relative to a landmark as follows:

$$X = \begin{bmatrix} \sqrt{(c_x - x)^2 + (c_y - y)^2} \\ \arctan(c_y - y) / (c_x - x) - \theta \end{bmatrix}$$

Jacobian of vector X is:

$$H = \begin{bmatrix} \frac{-c_x + x}{\sqrt{x^2+y^2}} & \frac{-c_y + y}{\sqrt{x^2+y^2}} & 0 \\ \frac{y - c_y}{\sqrt{(c_x-x)^2+(c_y-y)^2}} & \frac{x - c_x}{\sqrt{(c_x-x)^2+(c_y-y)^2}} & -1 \end{bmatrix}.$$

In EKF, H substitutes C. This process is the same as part a, but we use Jacobian H instead of C and update the Kalman gain using polar coordinates. R is also defined as follows:

$$R = \begin{bmatrix} \sigma^2_{\text{range}} & 0 \\ 0 & \sigma^2_{\text{bearing}} \end{bmatrix}.$$

According to Figure 3, the measurement probability is closer to reality. Using the Kalman filter, the predicted pose sometimes deviates from the reference. It depends heavily on the amount of noise and the speed of the robot. Figure 3 shows the results for $u_r = u_l = 0.1$.

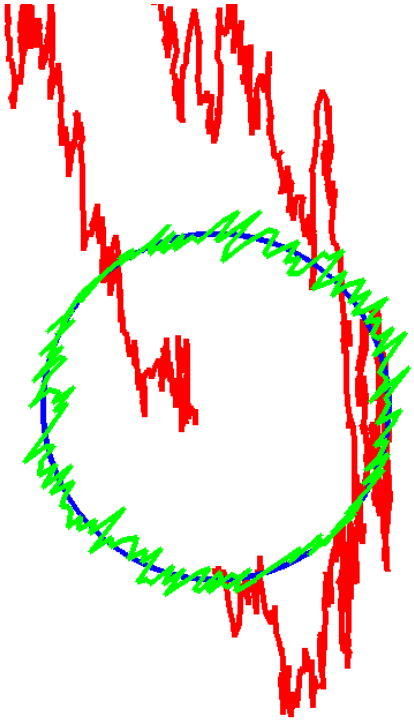

Figure 3: Trajectory of the Robot with non-linear EKF

Over time, the predicted pose may differ from the actual reference pose. Often, this happens if the measurements have too much noise, or if the system is moving too quickly for the filter to track it accurately.

References

[1] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.