#Junseo Kim

Assignment 2

Summer 2023

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

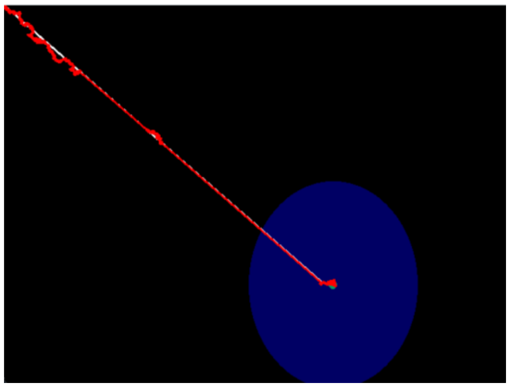


Figure 1: Kalman Filter Implementation with randomness 0.01

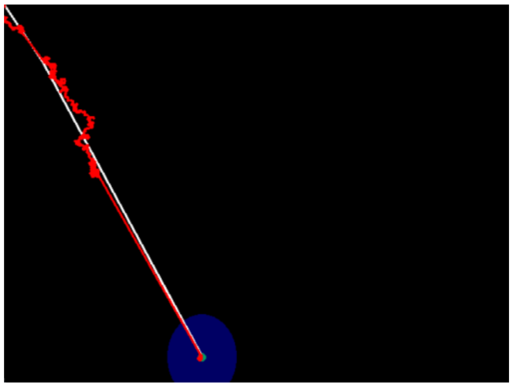
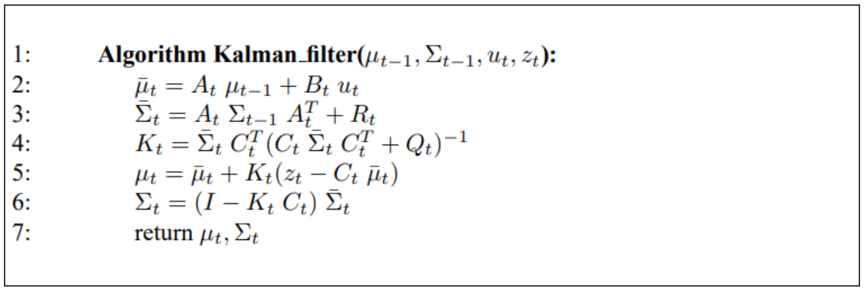


Figure 2: Kalman Filter Implementation with randomness 0.02

Red lines show the path with noise, white line shows the path with the Kalman filter

Blue Circle represents the covariance ellipse



```
**Algorithm Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
    $\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$
    $\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$
    $K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$
    $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\ \bar{\mu}_t)$
    $\Sigma_t = (I - K_t\ C_t)\ \bar{\Sigma}_t$
    return $\mu_t, \Sigma_t$
```

Figure 3: Kalman Filter Algorithm [1]

With the application of the Kalman filter Algorithm, the prediction step involves predicting the next state with the covariance with motion model.

The state prediction can be determined with the following equation

$$P = P + Q$$

where P is the state covariance and Q is the noise covariance set in the requirement

Measurement model is given by the equation

$$z = Hx + r$$

where z is the measurement, H is the measurement matrix, x is the state, and r is the measurement noise

As for the correction step, the state is updated with the Kalman gain

$$x = x + K(y - Hx)$$

y is the actual measurement. Covariance is also updated with new equation

$$P = (I - KH)P$$

Detailed Codes can be found at
https://github.com/JunseoKim19/State_estimation/blob/main/Kalman_Filter_Example.py

Full simulation can be found at
https://youtu.be/Jcn_a-hLRNk

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 0.3m.

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

Assume

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

Then the equations become:

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

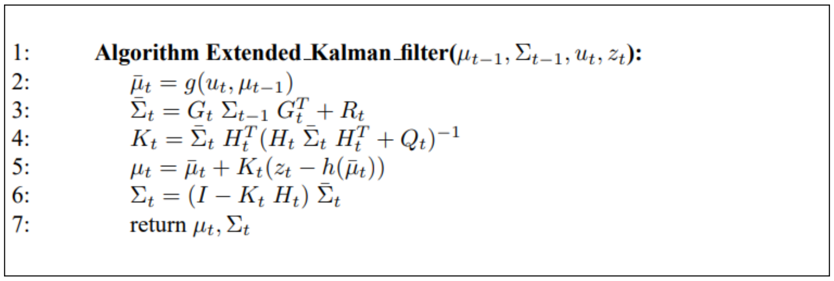
$w_\psi = N(0, 0.01)$ and $w_\omega = N(0, 0.1)$. Program the robot such that it loops around point M.

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

(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.

Part a)

As Extended Kalman Filter (EKF) is applied for the same linear measurement used in part a, the result is very similar.



```
1:  Algorithm Extended_Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2:      $\bar{\mu}_t = g(u_t, \mu_{t-1})$
3:      $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
4:      $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
5:      $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6:      $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$
7:      return $\mu_t, \Sigma_t$
```

Figure 4: Extended Kalman Filter Algorithm [1]

In the prediction step,

$$x = f(x,u) + w$$

The state transition function, with the state and control input, and the process noise were simplied as the following:

$$x = prev_x + \frac{r}{2}(ur + ul) * np.array([1,1]) * T + noise$$

The covariance equation is simplied to

$$P = FPF + Q$$

as F is assume to be the identity matrix since linear model is applied to the system.

For the correction step,

Jacobian matrix is also considered as an identity matrix because the measurement function H is

assumed to be a linear function.
The updated state estimation is also simplified due to the linear model

$$x = x + K(z - H) \rightarrow x = x + Kx$$

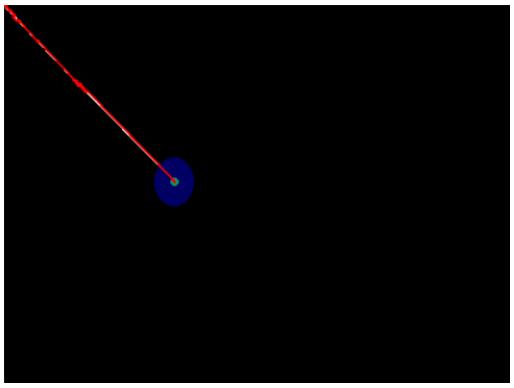


Figure 5: Extended Kalman Filter with linear model

Detailed Codes can be found at
https://github.com/JunseoKim19/State_estimation/blob/main/Extended_Kalman_Filter_a.py

Full simulation can be found at
https://youtu.be/O7Jk0CAinwY

Part b)

For part b, new landmark is applied and pyplot was utilized instead of pygame for better visualization.

As new motion model and observation model are applied for the system, Jacobian for each model is determined for the extended Kalman filter estimation.

The motion model of the robot is given by:

$$x_t = x_{t-1} + DT * r * u_w * \cos(\varphi_{t-1})$$

$$y_t = y_{t-1} + DT * r * u_w * \sin(\varphi_{t-1})$$

$$\varphi_t = \varphi_{t-1} + DT * r * \frac{u_\varphi}{L}$$

u_w is the average of the control signals to the wheels and u_φ is the difference of the control signals for better computation.
Based on the motion model, Jacobian of the motion can be determined:

$$jF = \begin{bmatrix} 1 & 0 & -DT * r * u_w * \sin(\varphi_{t-1}) \\ 0 & 1 & DT * r * u_w * \cos(\varphi_{t-1}) \\ 0 & 0 & 1 \end{bmatrix}$$

Next, the observation model of the robot is given by:

$$d = \sqrt{(x_l - x_t)^2 + (y_l - y_t)^2}$$

$$\theta = atan2(y_l - y_t, x_l - x_t) - \varphi_t$$

Based on the observation model, Jacobian of the observation model can be determined:

$$H = \begin{bmatrix} -\frac{x_l - x_t}{d} & -\frac{y_l - y_t}{d} & 0 \\ \frac{y_l - y_t}{d^2} & -\frac{x_l - x_t}{d^2} & -1 \end{bmatrix}$$

Then, extended Kalman filter is applied to the system according to the Figure 4.

In the prediction step, motion model and the Jacobian of the motion model is determined. Then, prediction for the covariance is made based on the predicted Jacobian and the process noise covariance.

In the update step, Jacobian of the observation model is determined with the measurement prediction. Then, the difference between the actual measurement and the predicted measurement will be determined. With the Kalman gain, new state estimation and covariance estimation is found.

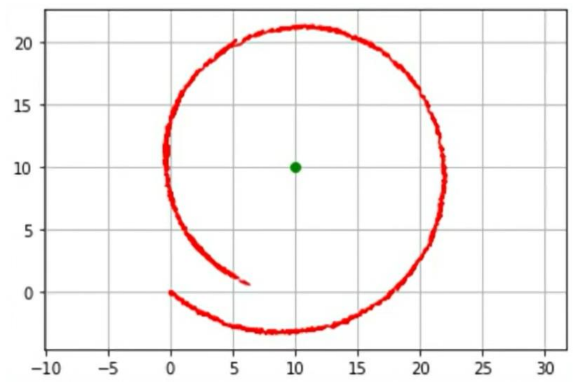


Figure 6: Extended Kalman Filter with non-linear model

Detailed Codes can be found at
https://github.com/JunseoKim19/State_estimation/blob/main/Extended_Kalman_Filter_b.py

Full simulation can be found at
https://youtu.be/vFin5vsXAR0

Reference

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2010.

[2] T. D. Barfoot, *State Estimation for Robotics*. Cambridge, United Kingdom: Cambridge University Press, 2017.