CMPE 245 Project 1

Leo Chen

June 11, 2018

1 Abstract

For this project, the task is to design an Extended Kalman Filter (EKF) and a Second Order Filter (SOF) to estimate the velocity and angle of direction of a moving robot.

2 Introduction

The first project consists of a robot moving in a given area. With the x- and y-coordinates and the orientation angle, we are supposed to design an 'Extended Kalman Filter' and a 'Second-Order Filter' to estimate the velocity and angle from a movie-clip provided for us so that we can have a comparison with the data provided. The dynamical model is provided, as well as measurements from the camera which will help me estimate the velocity and angle of the robot.

3 Methodology

3.1 Provided Data

The first thing that is provided for us is a movie clip that shows the movement of the robot across a plane. The movie clip can give a general idea of the movement of the robot for us later to compare. Through the movie, it is also possible to determine the noise that I will address later on in this report.

Another set of data provided for us is the file 'XY_Data' which provides the x- and y-coordinates in centimeters of the robot position in the first and second column respectively. The next data provided for us is the heading angle, which is only there as a reference to compare our results with.

The project also provides important models that are necessary to design the filter. First of all, the position of the robot is defined by the coordinates $[x, y, \Theta]^T$ in which x and y are the coordinates and Θ the orientation angle with $\Theta = 0$ being the x-axis. The state space form of the dynamical model for our robot is provided, as well:

$$X(t) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x \\ y \\ v \\ \Theta \end{bmatrix}$$

$$F(X(t)) = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} v(t)cos\Theta(t) \\ v(t)sin\Theta(t) \\ 0 \\ 0 \end{bmatrix}$$

$$\dot{X}(t) = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_3 cos x_4 \\ x_3 sin x_4 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \sigma_v(t) \\ \sigma_\Theta(t) \end{bmatrix}$$

It is also given that $\dot{v}$ and $\dot{\Theta}$ have zero-mean white noise and that the measurements are all absolute and with zero-mean Gaussian-distributed noise. The measurement model is

$$Z(t) = \begin{bmatrix} x_m(t) \\ y_m(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} X(t) + \begin{bmatrix} w_x \\ w_y \end{bmatrix}$$

The manual also mentions that the recorded data is every 10th frame of the movie while the movie was recorded with a rate of 30 fps which means that our $\Delta T = \frac{10}{30} = \frac{1}{3}$. Additional data that is provided is the image resolution of 320x240, and the size of the robot which is 70mm in its diameter. With this data, I will be able to measure the noise later on.

3.2 Derivation of model

The model is given to us in the manual is mentioned earlier. Let

$$\dot{x}_1 = dx, \dot{x}_2 = dy, \dot{x}_3 = dv, \dot{x}_4 = d\Theta$$

Knowing that

$$dx = vcos\Theta, dy = vsin\Theta, dv = \sigma_v dw_v, d\Theta = \sigma_\Theta dw_\Theta$$

which, by taking the antiderivative ultimately gives me

$$x_{k+1} = x_k + v_k cos(\Theta_k) * \Delta T$$
$$y_{k+1} = y_k + v_k sin(\Theta_k) * \Delta T$$
$$v_{k+1} = v_k + \sigma_v * \sqrt{\Delta T} w_v$$
$$\Theta_{k+1} = \Theta_k + \sigma_\Theta * \sqrt{\Delta T} w_\Theta$$

These equations will be necessary for both filters in order to compute the Kalman filter.

3.3 Mathematical Approach

3.3.1 Formulas for EKF

Equations that are important to remember are the ones in the prediction and update step in the discrete time model. Inside the prediction step the predicted state estimate is defined by

$$\hat{x}_{k+1}(-) = f_k(\hat{x}_k, 0)$$

while the predicted covariance estimate is

$$P_{k+1}(-) = \Phi_k P_k \Phi_k^T + \Gamma_k Q_k \Gamma_k^T$$

Inside the update step I have to determine the Kalman gain, the updated state and covariance estimate as well. The Kalman gain is defined by

$$K_{k+1} = P_{k+1}(-) H_{k+1}^T (H_{k+1} P_{k+1}(-) H_{k+1}^T + R_{k+1})^{-1}$$

while the updated state estimate and the updated covariance estimate are respectively defined by

$$\hat{x}_{k+1} = \hat{x}_{k+1}(-) + K_{k+1}(y_{k+1} - h_{k+1}(\hat{x}_{k+1}(-)))$$
$$P_{k+1} = (I - K_{k+1} H_{k+1}) P_{k+1}(-)$$

The state transition matrix Φ is defined by

$$\Phi_k = \frac{\delta f_k(\hat{x}_k, 0)}{\delta x_k}$$

Γ , which contains the noise is defined by

$$\Gamma_k = \frac{\delta f_k(\hat{x}_k, 0)}{\delta w_k}$$

and the observation matrix H is defined by

$$H_k = \frac{\delta h_k(\hat{x}_k(-), 0)}{\delta x_k}$$

After knowing all the equations, I had to take the derivatives of $\dot{X}(t)$ with respect to the corresponding parameter to plug it into the prediction and update step ultimately.

3.3.2 Calculations for EKF

To find Φ_k I need to take the partial derivative of f_k (rows) with respect to x_k (columns) and ultimately gives me a matrix of

$$\Phi_k = \Phi_k = \frac{\delta f_k(\hat{x}_k, 0)}{\delta x_k} = \begin{bmatrix} 1 & 0 & \Delta T * cos(x_4) & -\Delta T * sin(x_4) \\ 0 & 1 & \Delta T * sin(x_4) & \Delta T * cos(x_4) * x_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly to find Γ_k I need to take the partial derivative of f_k (rows) with respect to w_k (columns) and ultimately gives me a matrix of

$$\Gamma_k = \Gamma_k = \frac{\delta f_k(\hat{x}_k, 0)}{\delta w_k} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \sigma_v * \sqrt{\Delta T} & 0 \\ 0 & \sigma_\Theta * \sqrt{\Delta T} \end{bmatrix}$$

σ_v and σ_Θ have to be adjusted at the end of the project to find the best parameters for a good filter. The matrix H is already given to us in the manual which is

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Lastly, Q is a simple 2x2 identity matrix while R contains the light noise and looks as follows

$$R = \begin{bmatrix} 0.77 & 0 \\ 0 & 0.77 \end{bmatrix}$$

The light noise can be determined by finding the equivalence of the length of the robot (70mm) in pixels. Through that, we will be also able to determine how much noise is coming from the LEDs since the lens causes noise from the LED. The image below will give a good understanding of how much noise we are dealing with.

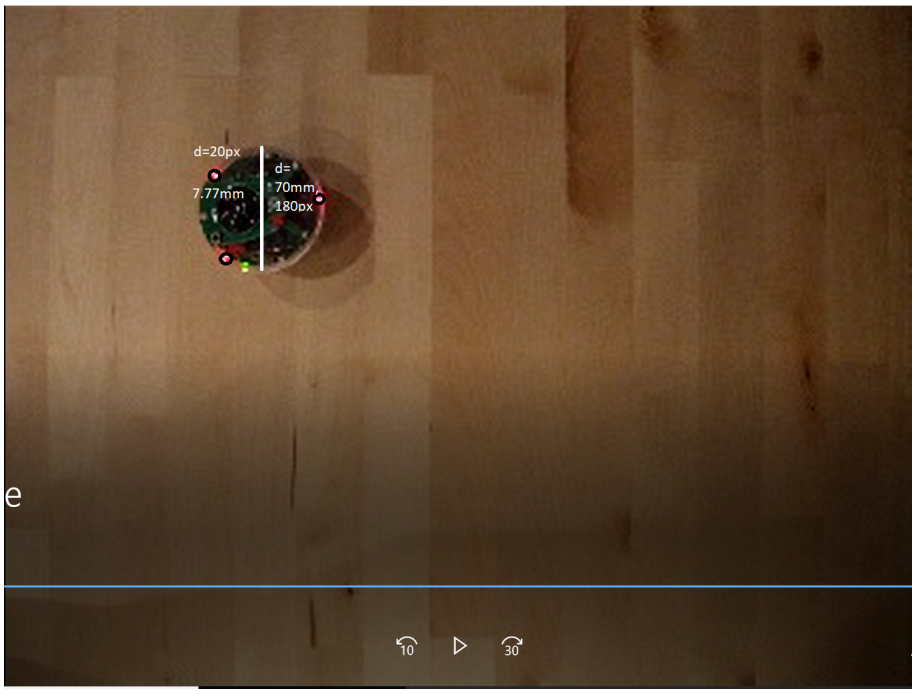

Figure 1: Noise Calculation

With all the matrices figured out, I can start implementing my Matlab code.

3.3.3 Formulas for the Second Order Filter

For the equations of the second order filter I know that for the prediction step there are three equations required. The first one is to determine predicted state estimate which is

$$\hat{x}_{k+1}(-) = f_k(\hat{x}_k, 0) + b_k$$

while similarly to the EKF the predicted covariance estimate is

$$P_{k+1}(-) = \Phi_k P_k \Phi_k^T + \Gamma_k Q k \Gamma_k^T$$

Another new thing to add for the second order filter is b_k which is

$$b_k = \frac{1}{2}\sum_{i=1}^{n} e_i (tr[F_k^i P_k] + tr[G_k^i Q_k])$$

For the update step there are different equations than for the EKF, as well

$$[S_{k+1}]_{ij} = \frac{1}{2} tr[M_{k+1}^i P_{k+1} M_{k+1}^j P_{k+1}(-)]$$

$$s_{k+1} = -\frac{1}{2} K_{k+1} \sum_{i=1}^{r} e_i tr[M_{k+1}^i P_{k+1}(-)]$$

As well as the updated state estimate

$$\hat{x}_{k+1} = \hat{x}_{k+1}(-) + K_{k+1}(y_{k+1} - h_{k+1}(\hat{x}_{k+1}(-))$$

with the updated covariance estimate

$$P_{k+1} = P_{k+1}(-) - P_{k+1}(-) H_(^T k+1)[H_{k+1} P_{k+1}(-) H_{k+1}^T + R_{k+1} + S_{k+1}]^{-1} H_{k+1} P_{k+1}(-)$$

The parameters such as the state transition matrix

$$\Phi_k = \frac{\delta f_k(\hat{x}_k, 0)}{\delta x_k}$$

and Γ , which contains the noise is defined by

$$\Gamma_k = \frac{\delta f_k(\hat{x}_k, 0)}{\delta w_k}$$

and the observation matrix H is defined by

$$H_k = \frac{\delta h_k(\hat{x}_k(-), 0)}{\delta x_k}$$

The newer matrices require the second order partial derivative such as

$$F_k^i = \frac{\delta^2 f_k^i(\hat{x}_k, 0)}{\delta x_k^2}$$

$$G_k^i = \frac{\delta^2 f_k^i(\hat{x}_k, 0)}{\delta w_k^2}$$

$$M_k^i = \frac{\delta^2 h_k^i(\hat{x}_k(-), 0)}{\delta x_k^2}$$

3.3.4 Calculations for the Second Order Filter

To find the different matrices, it is necessary to compute the second order partial derivatives. Only Φ, H, R and Γ remain the same as for the EKF. To calculate F_k^i there are 4 matrices for each $f_1 - f_4$. The matrices are calculated by taking partial derivatives of x_1 to x_4 for both row and columns which means that the transpose of it is the same. Therefore,

$$F_1 = \frac{\delta^2 f_k^1(\hat{x}_k, 0)}{\delta x_k^2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\Delta T sinx_4 \\ 0 & 0 & -\Delta T sinx_4 & -\Delta T x_3 cosx_4 \end{bmatrix}$$

$$F_2 = \frac{\delta^2 f_k^2(\hat{x}_k, 0)}{\delta x_k^2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Delta T cosx_4 \\ 0 & 0 & \Delta T cosx_4 & -\Delta T sinx_4 \end{bmatrix}$$

$$F_3 = F_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Taking the partial derivatives for M_k^i and G_k^i results to zero with the same concept of using the derivatives which therefore cancels out $[S_{k+1}]_{ij}$ and s_{k+1}. Now that everything is known, I am ready to code it in Matlab.

3.4 Matlab Implementation

(Note: The Matlab code is submitted along with the e-mail)

I started off by loading all the data provided into the folder. Then I declared all the variables and matrices. The initial value for P is

$$P_0 = \begin{bmatrix} 0.77 & 0 & 0 & 0 \\ 0 & 0.77 & 0 & 0 \\ 0 & 0 & \frac{0.77}{\Delta T^2} & \\ 0 & 0 & 0 & \frac{\pi}{180} \end{bmatrix}$$

with P_{11}=P_{22}=0.77 being the noise and P_{33} being the noise divided by ΔT and P_{44} being the starting angle. The initial x- and y-coordinates are the initial x- and y-coordinates from the data provided for us. The initial velocity can be calculated by subtracting the first data point from the second data point (both from the x- and y-coordinates) and dividing it with ΔT which gives me the initial velocity in the x- and y-direction

$$v_x = \frac{x_2 - x_1}{\Delta T}, v_y = \frac{y_2 - y_1}{\Delta T}$$

Ultimately, to find the overall velocity, I used

$$v = \sqrt{v_x^2 + v_y^2}$$

For the initial angle, I used the first data point from the provided heading angle data, as well. After I declared everything, I am using a for loop that runs as many times as the amount of data points we have. Inside, I have the functions

$$x_{k+1} = x_k + v_k cos(\Theta_k) * \Delta T$$

$$y_{k+1} = y_k + v_k sin(\Theta_k) * \Delta T$$

$$v_{k+1} = v_k + \sigma_v * \sqrt{\Delta T} W_v$$

$$\Theta_{k+1} = \Theta_k + \sigma_\Theta * \sqrt{\Delta T} W_\Theta$$

but since it is a zero-mean white noise, I only used

$$x_{k+1} = x_k + v_k cos(\Theta_k) * \Delta T$$

$$y_{k+1} = y_k + v_k sin(\Theta_k) * \Delta T$$

$$v_{k+1} = v_k$$

$$\Theta_{k+1} = \Theta_k$$

Lastly, I coded the prediction and update steps from the formulas I mentioned above and plotted the results.

4 Results

The results are provided in the figures 2 and 3.

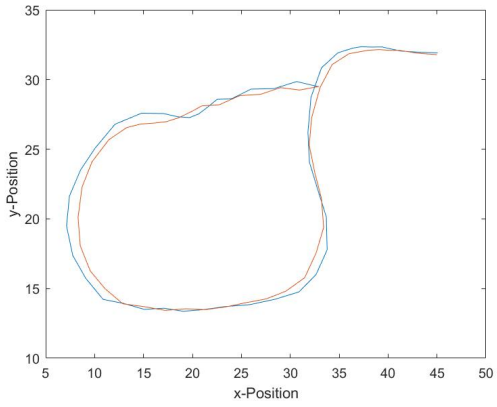

Figure 2: Plot of EKF, (blue: own results, red: data provided)

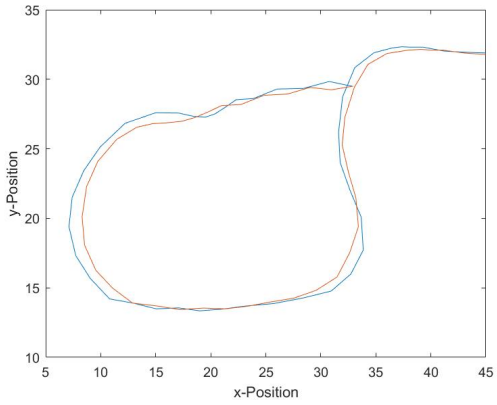

Figure 3: Plot of SOF, (blue: own results, red: data provided)

As you can see, there is only a slight improvement in the second-order filter because I am using the second partial derivative of the functions. This makes sense since for the EKF there is only one order of derivative required which gives us more room for residuals. Since both are very similar, I will discuss furthermore on how to adjust the noise σ_v and σ_Θ. To find out the perfect noise value, I used really big and really small values first. I noticed that the bigger values would give me a result that is very off from the data that we can use to compare with. Results are below with σ_v=15 and σ_v=12.

I also noticed that having a larger noise in the angle will affect the results a lot more than having a larger noise in the velocity. After running the code multiple times with different values for the noise I came to the conclusion that the most fitting values for the noise in my EKF would ultimately be σ_v=1.5 and σ_Θ=1.2. The results are decent. My results are slightly off from the data that was provided for us, especially when it is on the left hand side of the plane. This could result from an offset in the noise parameter I used from the pixel calculation. Since the measurements

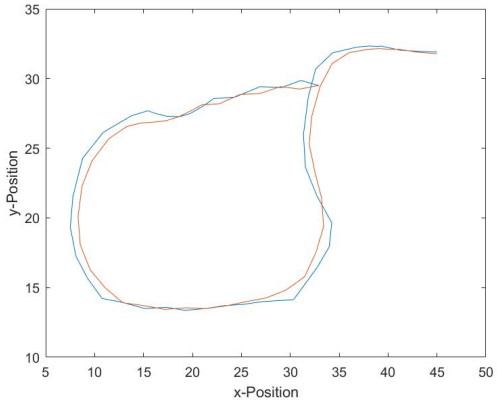

Figure 4: Testing with large σ_v, (blue: own results, red: data provided)

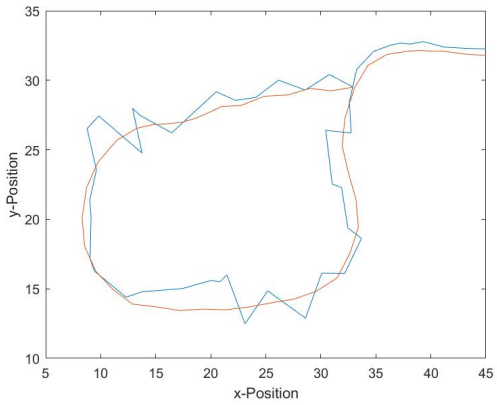

Figure 5: Testing with large σ_Θ, (blue: own results, red: data provided)

of noise might not be exactly right within the R-matrix in my code, there might be a lot of room for error in the filter. Therefore, I would say that there can be more room for improvement in my filter but still this is a decent result in which my filter results still somehow match with the data in a visible way.

5 Conclusion

All in all, this project was really helpful to understand the concept of a Kalman filter. It was a real life application of using provided data from an actual robot and then applying my knowledge to an actual Kalman filter. Both filters provided a better comparison of what similarities and differences they provide. The Matlab coding went smooth and understanding how to find the matrices was confusing at the beginning but makes a lot of sense now.