ESP3201

KALMAN FILTER MINI-ASSIGNMENT

Goh Kheng Xi, Jevan
A0199806L

Contents

1. Introduction

Kalman Filter is a recursive prediction algorithm for dynamic linear systems that estimates an internal immeasurable state of the system based on the measured data [1]. Common applications include navigation systems for autonomous robot, target tracking the assumptions that the error estimate obeys a gaussian distribution [2].

1.1 Aim

The problem statement given is to design a Kalman filter to track a target in a 2-dimensional space, given a set of measurement values and the measurement standard deviations. Ground truth values are given to compare and access the performance of the Kalman filter.

2. Designing of Kalman Filter

The design of the Kalman Filter for this problem statement are based on a few assumptions. The process(w) and measurement noise(v) are assumed to follow a gaussian distribution of mean 0 and their individual standard deviations, Q and R respectively. It is assumed that there are no control inputs to the system (such as throttle inputs, braking, turning, etc).

2.1 Mathematical Model

The algorithm for the Kalman filter involves the following set of 5 equations [3]:

$$\hat{X}_k^- = A\hat{X}_{k-1} + Bu_k \quad \text{(equation 1)}$$

$$P_k^- = AP_{k-1}A^T + Bu_k \quad \text{(equation 2)}$$

$$K_k = \frac{P_k^- C^T}{CP_k^- C^T + R} \quad \text{(equation 3)}$$

$$\hat{X}_k = \hat{X}_k^- + K_k(Z_k - C\hat{X}_k^-) \quad \text{(equation 4)}$$

$$P_k = (I - K_k C)P_k^- \quad \text{(equation 5)}$$

Where X represents the state matrix and P represents the covariance matrix. A hat notation denotes a prediction and a bar notation denotes a priori estimate.

The Kalman filter also requires a mathematical model of the linear system to calculate the innovation term and update the system's state.

$$x = x_0 + V_x \cdot dt + \frac{1}{2} \cdot a \cdot dt^2 \quad \text{(equation 6)}$$

Using the kinematics equation for displacement of an object in a 1-dimensional space (equation 6), the following recursive equation is derived:

$$x_k = x_{k-1} + V_x \cdot dt + \frac{1}{2} \cdot a \cdot dt^2 \quad \text{(equation 7)}$$

2.2 Matrices

$$X = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}$$

$$x: displacement\ of\ object(car)\ in\ the\ x - direction$$

$$y: displacement\ of\ object\ in\ the\ y - direction$$

$$\dot{x}: velocity\ of\ the\ object\ in\ the\ x - direction$$

$$\dot{y}: velocity\ of\ the\ object\ in\ the\ y - direction$$

The state matrix, X, is a 4 x 1 matrix that represents the internal state of the system.

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The state gain, A, is a 4 x 4 matrix obtained based on the kinematic model in equation 7. The time step, dt, corresponds to the measurement period given in the problem statement as 1 second.

$$Z = \begin{bmatrix} x_m \\ y_m \end{bmatrix}$$

The x_m and y_m variables in the 2 x 1 measurement matrix, Z, represent the measured x and y displacement of the target respectively.

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

The measurement matrix, C, represents the output gain of the linear system.

$$P = \begin{bmatrix} {\sigma_x}^2 & \sigma_x\sigma_y & \sigma_x\sigma_{v_x} & \sigma_x\sigma_{v_y} \\ \sigma_y\sigma_x & {\sigma_y}^2 & \sigma_y\sigma_{v_x} & \sigma_y\sigma_{v_y} \\ \sigma_{v_x}\sigma_x & \sigma_{v_x}\sigma_y & {\sigma_{v_x}}^2 & \sigma_{v_x}\sigma_{v_y} \\ \sigma_{v_y}\sigma_x & \sigma_{v_y}\sigma_y & \sigma_{v_y}\sigma_{v_x} & {\sigma_{v_y}}^2 \end{bmatrix}$$

P is a 4 x 4 covariance matrix of the system, where σ represents the standard deviation of the variable.

$$Q = Q_0 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Q is a 4 x 4 process noise covariance matrix where Q_0 is a hyperparameter to be tuned.

$$R = \begin{bmatrix} {\sigma_x}^2 & 0 \\ 0 & {\sigma_y}^2 \end{bmatrix} = \begin{bmatrix} 10^2 & 0 \\ 0 & 10^2 \end{bmatrix}$$

R is a 2 x 2 measurement noise covariance matrix where the standard deviation of the measured x-displacement of the target, σ_x, and the standard deviation of the measured y-displacement of the target, σ_y, are both given as 10.

B is a matrix that represents the input gain, while u represents the input. However, as the acceleration of the target is assumed to be 0 throughout the entire tracking process, the input term is 0 and hence there is no need for the B and u term.

2.3 MATLAB code

```
%*****************************************************************
%                        Kalman Filter Code
%*****************************************************************

%======================== import of data ============================
opts = detectImportOptions('MeasGT.csv');
opts.SelectedVariableNames = (2:5);
measured_data = readmatrix('MeasGT.csv', opts);

%---------- measured vales ----------------
measured_x = measured_data(1:31,1);
measured_y = measured_data(1:31,2);

% ----------- Ground truths ---------------
groundT_x = measured_data(1:31,3);
groundT_y = measured_data(1:31,4);

%====================== end of import ============================

% --------------- initialising of variables ----------------------
x = 500;       % initial predicted x
y = -200;    % initial predicted y
v_x = 10;    % initial predicted x velocity
v_y = 10;     % initial predicted y velocity
v_max = 12.5;    % estimate of maximum velocity for an initial covariance
matrix prediction
Q0 = 0.01;    % small process noise variance
std_measured_x = 10;     % standard deviation for measured x values
std_measured_y = 10;     % standard deviation for measured y values

X = [x; y; v_x; v_y];    % state matrix
```

```
A = [1 0 1 0; 0 1 0 1; 0 0 1 0; 0 0 0 1]; % A matrix corresponding to the
state matrix used
C = [1 0 0 0; 0 1 0 0]; % transformation matrix to retain only the measured
values
Z = [measured_x measured_y];     % Measurement matrix
P = [10^2 0 0 0; 0 10^2 0 0; 0 0 (v_max/3)^2 0; 0 0 0 (v_max/3)^2]; %
initialised covariance matrix
Q = Q0 * [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];  % process noise variance
matrix
R = [std_measured_x^2 0; 0 std_measured_y^2];   % measurement noise matrix

%creating result array
res_x = zeros(32,1);
res_y = zeros(32,1);

%storing the initial prediction values into the result
res_x(1) = X(1,1);
res_y(1) = X(2,1);

%============================== start of algorithm
==============================
for i = 1:31

    % ------- prediction (priori estimate) -------
    X_priori = A*X; % predicted state BEFORE measurement using mathematical
model
    P_predict = A*P*transpose(A)+Q; % predicted covariance matrix

    %----- update (posteriori estimate) -------
    %caluculating Kalman gain
    K_denominator = C*P_predict*transpose(C) + R; % denominator of Kalman
gain
    K = P_predict*transpose(C)*pinv(K_denominator); %Kalman gain

    %updating state and covariance
    E = [Z(i,1); Z(i,2)] - (C*X_priori);
    %E = [measured_x(i); measured_y(i)] - (C*X_priori); % Innovation
residual(i.e. error term)
    X = X_priori + K*E; % update posterior estimate for state matrix
    P = (eye(4) - K*C)*P_predict; % update covariance matrix

    %storing of results
    res_x(i+1) = X(1,1);
    res_y(i+1) = X(2,1);

end
%============================== end of algorithm
==============================
%Plotting of graph
plot(groundT_x, groundT_y, 'k',res_x, res_y, 'r- .');
```

2.4 Algorithm

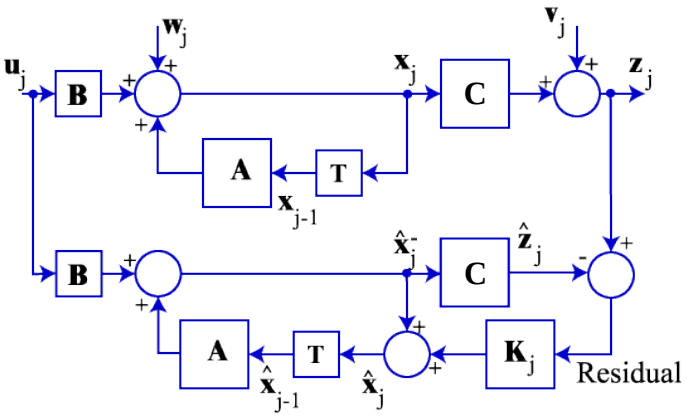

Figure 1. Block diagram of Kalman Filter

Figure 1 shows the block diagram of the Kalman filter, where w and v represent the gaussian errors for the process input and the measurement output respectively.

The start of the algorithm occurs at the start of the for loop as annotated clearly in the code. The algorithm is split into 2 predominant steps, the prediction where a priori estimate is calculated based on the mathematical model and the updating step where the posteriori estimate is calculated based on the priori estimate and the measured values.

In the prediction step, a priori estimate of the state and the covariance matrices are calculated based on the state and the covariance estimated in the previous loop. For the initial loop, the "previous" state and covariance matrices are initialised based on rough estimation.

In the updating step, the Kalman gain is a 4 x 2 matrix, calculated based on the covariance matrix predicted in the prediction step and the measurement covariance matrix. Next, the error residual ($Z_k - C \cdot \hat{x}_k^-$) is calculated based on the difference between the measured values and the priori estimate, and a posteriori estimate of the system's state is obtained by adding the dot product of the Kalman gain and the error residual to the priori state estimate. Finally, the updated values of the covariance matrix are calculated based on the Kalman gain and the predicted covariance matrix in the prediction step. The updating algorithm is formulated such that the posteriori estimate is based on the proportion of the predicted and measurement covariance. The larger the predicted covariance, the more weightage the posterior estimates places on the measurement values, while the larger the measurement covariance, the more weightage the posteriori estimates places on the predicted values.

The x and y displacement in the posterior state estimate of the loop are then stored in a result matrix for tabulation and plotting of the results. Once the state and covariance matrices are updated and the results are stored, the algorithm repeats, using the new updated state and covariance as the basis for the calculation of the next priori state and covariance matrices, thus completing the algorithm.

3. Testing and tuning

In the designed Kalman filter, there are a few hyperparameters that can be tuned to improve the performance of the Kalman filter. These hyperparameters are Q_0, initial X (consisting of x, y, V_x , V_x) and P estimate. As the initial state and covariance matrices will converge eventually in the algorithm, it is more important to tune the Q_0 value. For the testing and tuning of Q_0, the initial

$x, y, \dot{x}, \dot{y}$ are initialised to 500, -200, 15, 15 which allows the convergence and the smoothness of the tracking to be observed. P is set to be $\begin{bmatrix} {\sigma_x}^2 & 0 & 0 & 0 \\ 0 & {\sigma_y}^2 & 0 & 0 \\ 0 & 0 & \left(\frac{V_{max}}{3}\right)^2 & 0 \\ 0 & 0 & 0 & \left(\frac{V_{max}}{3}\right)^2 \end{bmatrix}$, where σ_x = 10, σ_y = 10, and V_{max} = 12.5.

3.1 Results

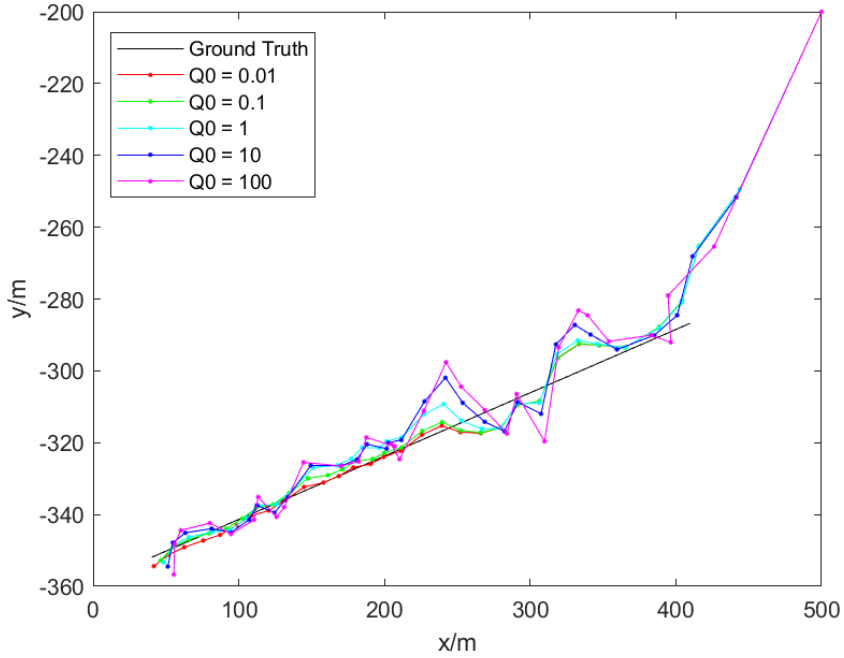

Figure 2. Graph of varying Q0 values

By keeping the other variables constant and varying Q_0, results in Figure 2 shows that the higher the Q_0, the faster the convergence, but the more unstable the tracking is after the convergence where more oscillations are observed.

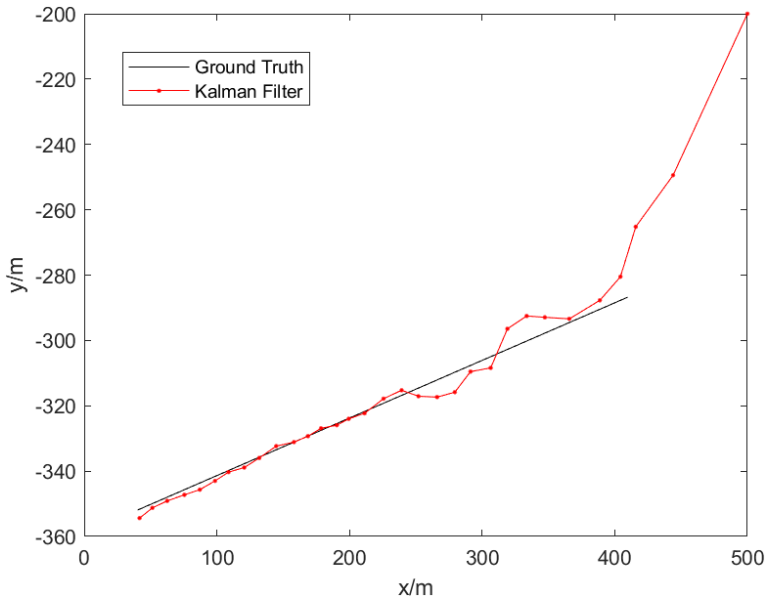

Figure 3. Graph of finalised model

Settling the value Q_0 on 0.001, Figure 3 shows the final model for the Kalman Filter. It converges quickly around the fourth iteration (inclusive of the initial estimate) and the remaining estimated points are relatively smooth and accurate.

4. References

[1] S. Srinivasan, "The Kalman Filter: An algorithm for making sense of fused sensor insight," Towards Data Science Inc., 19 April 2018. [Online]. Available: https://towardsdatascience.com/kalman-filter-an-algorithm-for-making-sense-from-the-insights-of-various-sensors-fused-together-ddf67597f35e. [Accessed 27 August 2021].

[2] H. Lim Chot, O. Lee Yeng, L. Tien Sze and K. Voon Chet, "Kalman Filtering and Its Real-Time Applications," in *Real-time Systems*, IntechOpen, 2016.

[3] K. Youngjoo and B. Hyochoong, Introduction to Kalman Filter and Its Applications, IntechOpen, 2018.