Task 4 :- Tracking Multiple Objects

Srishty Gandhi, roll no. 20CS30052

***Abstract*— The task was to make and apply Kalman filter, and then find the trajectory of drone. Data from 6 stationary ground sensors was given and obviously the sensors weren't ideal, so the measurement recorded by them was noisy. Using these measurements we had to find the trajectory of the drone.**

Since all the ground frames are stationary, and we know that velocity remains same in different frames respect to ground, we can take the measured velocity to be average of difference of two consecutive measured data.

I. INTRODUCTION

The Kalman filter was invented in the 1950s by Rudolph Emil Kalman, as a technique for filtering and prediction in linear systems. The Kalman filter implements belief computation for continuous states. It is not applicable to discrete or hybrid state spaces.

The Kalman filter represents beliefs by the moments representation: At time t, the belief is represented by the the mean µt and the covariance t. Posteriors are Gaussian if the following three properties hold, in addition to the Markov assumptions of the Bayes filter.

II. PROBLEM STATEMENT

Kalman filter is an amazing tool in order to estimate predicted values.

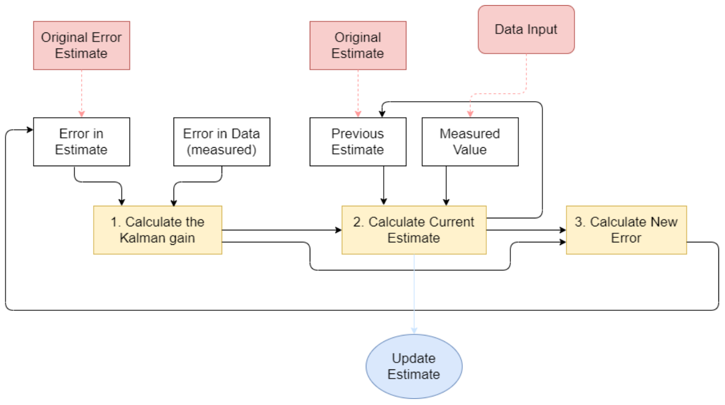
Kalman Filter is an iterative mathematical process that uses a set of equations and consecutive data inputs to quickly estimate the true value, position, velocity, etc of the object being measured, when the measured value contain unpredicted or random error, uncertainty, or variation.

Let's say we have a lot of data points, that come in one at a time, then if we find the average value and say that the average value is closer to true value in order to do that we need to have a whole bunch of inputs already. Kalman filter doesn't wait for whole bunch of inputs, it very quickly starts to narrow in to the true value by taking a few of those inputs and by understanding the variation or uncertainty of those inputs.

The data coming in is not the true value it's somewhere around the true value with a certain amount of uncertainty.

The following flowchart summarises all the processes that are going on in kalman filter, which is pretty self explanatory.

- yellow boxes – the mathematical calculations
- red boxes – the values we need only the first time
- blue box – output
- white box – the values needed/calculated in every iteration



Kalman Gain:-

One of the main portions of kalman filter is the kalman gain. Kalman gain is used to determine how much of the new measurements to use to update the new estimate. Kalman gain is calculated using the error in estimate, and the error in the data that we're reading.

$$KG = \frac{E_{EST}}{E_{EST} + E_{MEA}}$$

KG = Kalman gain

$E_{EST} = ErrorinEstimate$

$E_{MEA} = ErrorinMeasurement$

$$EST_t = EST_{t-1} + KG[MEA - EST_{t-1}]$$

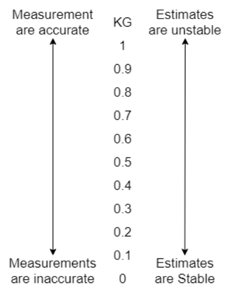
Current estimate = EST_t

Previous Estimate = EST_{t-1}

Measurement = MEA

If the Kalman gain is high (close to 1) That means that the measurements we're getting are fairly accurate and error in measurement is very small, which means the measured value will have more contribution as compared to estimated value.

On the other hand if Kalman gain is low(close to zero). That means the measurements that we are getting have high uncertainty in it, therefore the measured value will now have less contribution as compared to estimated value



Kalman Filter

There are three main equations equations that we use in kalman filter repeatedly. Each time we get a new data point we recalculate

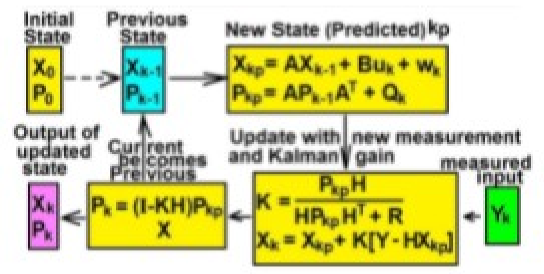
- *Kalman Gain*
- *The new estimate*
- *the new error*

$$E_{EST_t} = \frac{(E_{MEA})(E_{EST_{t-1}})}{E_{MEA} + E_{EST_{t-1}}}$$

$$\downarrow$$

$$E_{EST_t} = [1 - KG](E_{EST_{t-1}})$$

For this task, we have to use kalman filter with help of matrices.



The problem statement was to implement the above explained kalman filter, on a given data set. A drone flying in an area,where, there are 6 ground and on seeing the drone they provide you the coordinates of the drone with respect to them. Since they are all stationary frame of reference, which means that the velocity of the drone measured by each of the stations would be same. Hence we can consider the velocity of the drone as difference of the two consecutive data points

III. Related Work

Other filters such as histogram filter, particle filter etc can be used to deal with the similar problem

IV. INITIAL ATTEMPTS

Initial attempt was a simple one it did not include any matrices. The loop did 2 things repeatedly

1) *Measurement Update*
2) *Prediction*

```
def update(mean2, var1, mean2, var2):
    new_mean=(var2*mean1+var1*mean2)/
    (var1+var2)
    new_var = 1/ (1/ var1 + 1/ var2)
    return [new_mean, new_var]
```

```
def predict(mean1, var1, mean2, var2):
    new_mean = mean1 + mean2
    new_var = var1 + var2
    return [new_mean, new_var]
```

V. Final Approach

Step 1:- *Calculate* $X_{k(predicted)}$
X = State Matrix
U = Control Variable Matrix
W = Noise in the process

$$X_{k(predicted)} = AX_{k-1} + BU_k + W_k$$

$$X = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \quad A = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad U = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix}$$

If we rearrange the above equation we will finally get the equation of kinematics

$$x = x_o + \dot{x}\Delta t + \ddot{x}\frac{1}{2}\Delta t^2$$

$$y = y_o + \dot{y}\Delta t + \ddot{y}\frac{1}{2}\Delta t^2$$

$$z = z_o + \dot{z}\Delta t + \ddot{z}\frac{1}{2}\Delta t^2$$

$$\dot{x} = \dot{x}_o + a_x\Delta t$$

$$\dot{y} = \dot{y}_o + a_y\Delta t$$

$$\dot{z} = \dot{z}_o + a_z\Delta t$$

Step 2 :- *Calculate initial process co-variance matrix (we have to do this only once)*

$$P_{k-1} = \begin{bmatrix} \delta_x^2 & \delta_x\delta_y & \delta_x\delta_z & \delta_x\delta_{\dot{x}} & \delta_x\delta_{\dot{y}} & \delta_x\delta_{\dot{z}} \\ \delta_y\delta_x & \delta_y^2 & \delta_y\delta_z & \delta_y\delta_{\dot{x}} & \delta_y\delta_{\dot{y}} & \delta_y\delta_{\dot{z}} \\ \delta_z\delta_x & \delta_z\delta_y & \delta_z^2 & \delta_z\delta_{\dot{x}} & \delta_z\delta_{\dot{y}} & \delta_z\delta_{\dot{z}} \\ \delta_{\dot{x}\delta_x} & \delta_{\dot{x}\delta_y} & \delta_{\dot{x}\delta_z} & \delta_{\dot{x}}^2 & \delta_{\dot{x}}\delta_{\dot{y}} & \delta_{\dot{x}}\delta_{\dot{z}} \\ \delta_{\dot{y}\delta_x} & \delta_{\dot{y}\delta_y} & \delta_{\dot{y}\delta_z} & \delta_{\dot{y}}\delta_{\dot{x}} & \delta_{\dot{y}}^2 & \delta_{\dot{y}}\delta_{\dot{z}} \\ \delta_{\dot{z}\delta_x} & \delta_{\dot{z}\delta_y} & \delta_{\dot{z}\delta_z} & \delta_{\dot{z}}\delta_{\dot{x}} & \delta_{\dot{z}}\delta_{\dot{y}} & \delta_{\dot{z}}^2 \end{bmatrix}$$

Step 3:- *Calculate Predicted process covariance matrix*

$$P_k = AP_{k-1}A^T + Q$$

Step 4:- *Calculate kalman gain*

$$KG = \frac{P_kH}{HP_KH^t + R}$$

Step 5 :- *Read new observation*

$$Y_k = CY_{k_m} + Z_k$$

Step 6:- *Calculate the current state*

$$X_k = X_{k_p} + KG[Y_k - HX_{k_p}]$$

Step 7:- *Update the process covariance matrix*

$$P_k = [I - (KG)H]P_{k_p}$$

Step 8:- *Continue the cycle*

$$X_{k-1} = X_k$$

$$P_{k-1} = P_k$$

VI. Future Work

Some advanced version of kalman filter can be used such as extended kalman filter, particle filter etc.

CONCLUSION

Usefulness to ARK :- Kalman Filter as applied in this task to localise the drone itself is very useful for the drones in ark. Suppose once you planted a tracker which tracks your drone, and sends its location to you via Bluetooth , but due some logistics issues(such as storm) it got lost. Now you know the where that tracker is but you don't know its exact location. An easy solution to find it would be to place another tracker whose location is known and the fly the drone in known trajectory. After comparing the data from these two trackers we can deduce the exact location of the tracker.

References

[1] *PROBABILISTIC ROBOTICS by Sebastian THRUN, Wolfram BURGARD and Dieter FOX*

[2] *https://classroom.udacity.com/courses/cs373/lessons/48723604/concepts/486709880923*

[3] *https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python*