Understanding and Applying Kalman Filters

By...

K.Surya Prakash : EE18BTECH11026
Ritwik Sahani : EE18BTECH11038

Brief Summary :

- Understand the theory behind KF .
- Implementing KF (from scratch using Python)
- Analysing Pros and Cons of Linear KF
- Implementing Extended Kalman Filter
  (to deal with non-linearity from scratch using Python)
- Sensor fusion :
    - How can multiple sensors be used
    - Improving predictions using all the information
    - An intuitive study
    - Where to put to use?

Definitions :

Prediction Model :

$$x_k = A_{k-1} x_{k-1} + B_{k-1} u_{k-1} + q_{k-1} ; q_{k-1} \sim \mathcal{N}(0, Q_{k-1})$$

Measurement Model :

$$y_k = H_k x_k + r_k ; r_k \sim \mathcal{N}(0, R_k)$$

A_k : State Transition Matrix

x_k : State vector

u_k : Control Input

q_k : Prediction Nois

r_k : Measurement N

Algorithm :

Predict Step :

$$\hat{x}_{k \mid k-1} = A_{k-1}\hat{x}_{k-1 \mid k-1} . + B_{k-1}u_{k-1}$$

$$P_{k \mid k-1} = A_{k-1} . P_{k \mid k-1} . A_{k-1}^T + Q_{k-1}$$

Update step :

$$K_k = P_{k \mid k-1} . H_k^T (H_k . P_{k \mid k-1} H_k^T + R_k)^{-1}$$

$$\hat{x}_{k \mid k} = \hat{x}_{k \mid k-1} + K_k (y_k - H_k . \hat{x}_{k \mid k})$$

$$P_{k \mid k} = (1 - K_k . H_k) P_{k \mid k-1}$$

Tracking a ball :

A ball is thrown at an angle ‘θ’ w.r.t ground , with a velocity ‘v’ .

A camera (sensor) tries to measure its x , y coordinates, which might be later used for image processing for blob detection, but it has some inherent measurement noise.

We want to apply a Kalman filter to keep track of the ball for smooth processing.

Case 1 : Ball thrown in vacuum (no non-idealities)

The state vector can be determined using Newton's law

The coordinates will be of the form :

$$x = v_x \Delta t + x_o ; v_x = v cos\theta$$
$$y = \frac{g}{2} \Delta t^2 + v_y \Delta t + y_o ; v_y = v sin\theta$$

We will now try to model the trajectory using a **linear kalman filter**.

Designing state models :

Prediction Model :

$$\bar{x} = Ax + Bu$$

$$\begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g \end{bmatrix}$$

Measurement Model :

$$\bar{y} = Hx$$

$$\begin{bmatrix} x_m \\ y_m \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix}$$

Noise characterisation : Q : Process noise, R : Measurement noise variance

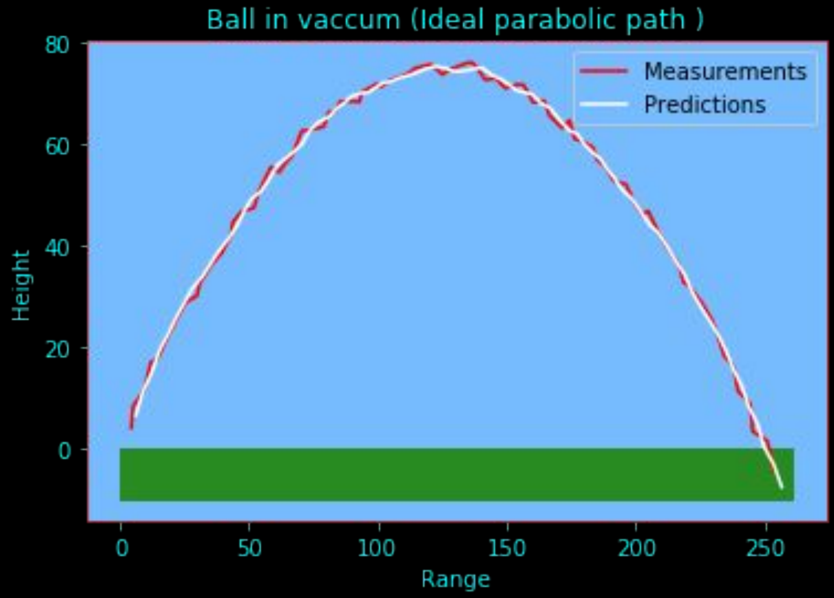
$$R = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} \qquad Q = \begin{bmatrix} q & 0 & 0 & 0 \\ 0 & q & 0 & 0 \\ 0 & 0 & q & 0 \\ 0 & 0 & 0 & q \end{bmatrix}$$

Implementation and Results of a Linear KF

$x_0 = 0$, $y_0 = 0$, $v = 50$, $\theta = 50°$

r = 0.5

q = 0.1



Case 2 : Adding non-linearity by considering air-drag

We now take air-drag into account , which is non-linearly dependent on v.

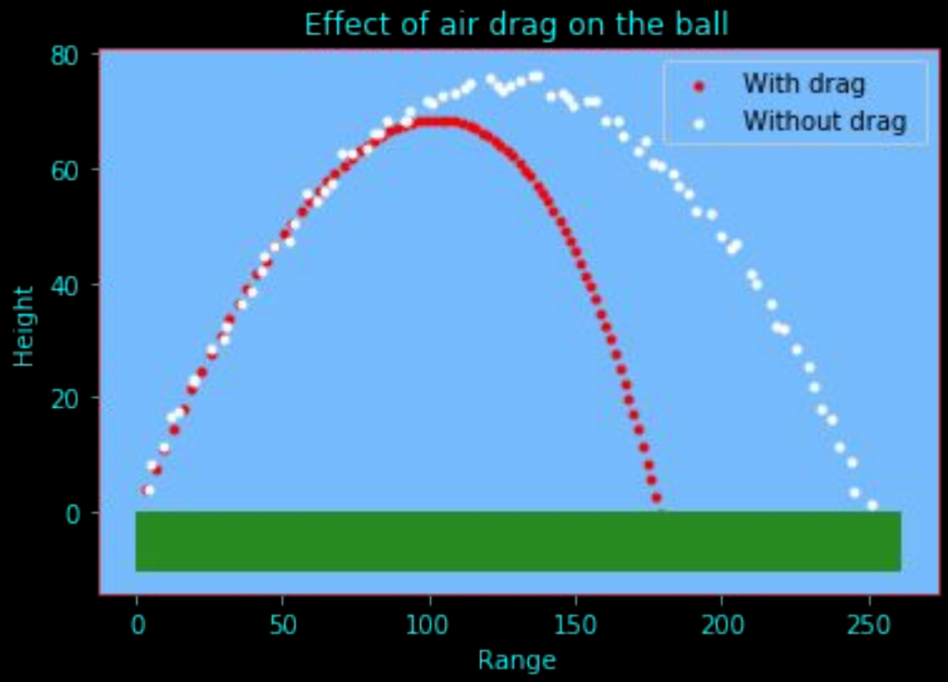
Force acting on the ball due to air drag : F_{drag}

$$F_{drag} = -B_2 v^2 ; v = \sqrt{v_x^2 + v_y^2}$$

$$a_x = -\frac{B_2}{m} v v_x$$
$$a_y = -\frac{B_2}{m} v v_y - g$$

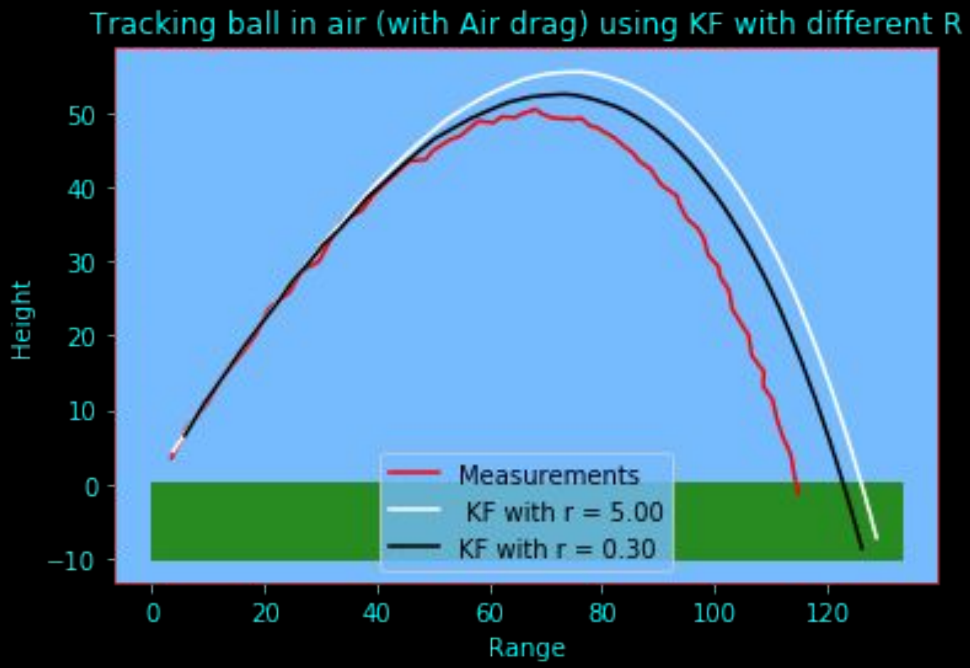
$$\frac{B_2}{m} = \frac{0.0058}{1+exp(\frac{v-35}{5})}$$

Effect of air-drag on the ball



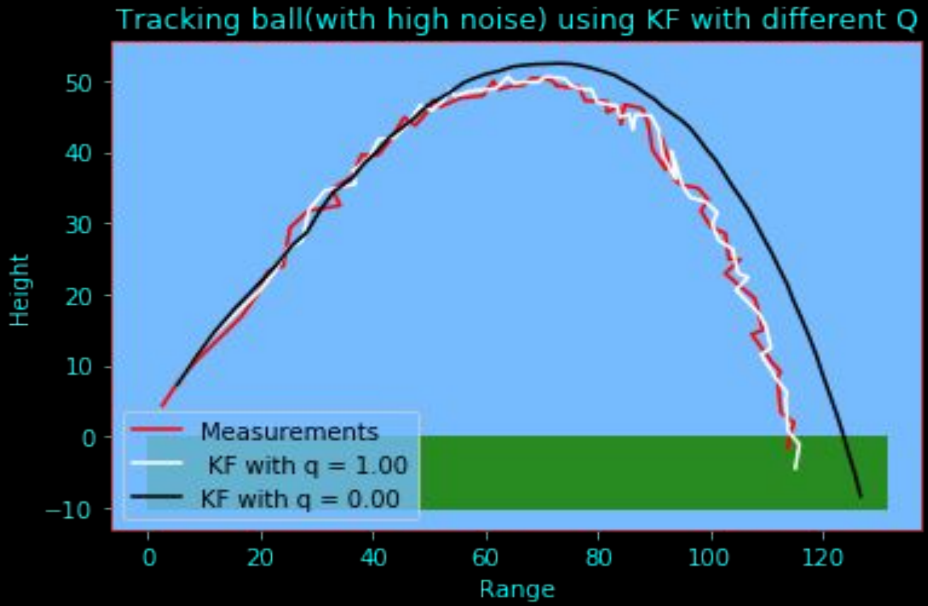
Applying Linear KF to tackle non-linearity

Performs poorly due to the inability to model non-linearity.



Using Process noise (Q) for better fitting

A smart engineering trick done to fit the predictions, but at the cost of smoothness.



How to deal with this …???

Extended Kalman Filters (to the rescue ...)

Finding A , to approximate the non-linearity . Computed using Jacobian matrix.

State vectors are calculated directly using the non-linear function.

State updation will be as follows :

$$\begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix} = \begin{bmatrix} x + v_x \Delta t \\ v_x - F.v_x.\Delta t \\ y + v_y \Delta t \\ v_y - 9.8\Delta t - F.v_y.\Delta t \end{bmatrix}$$

$$F = \frac{0.0058}{1+exp(\frac{v-35}{5})}.v$$

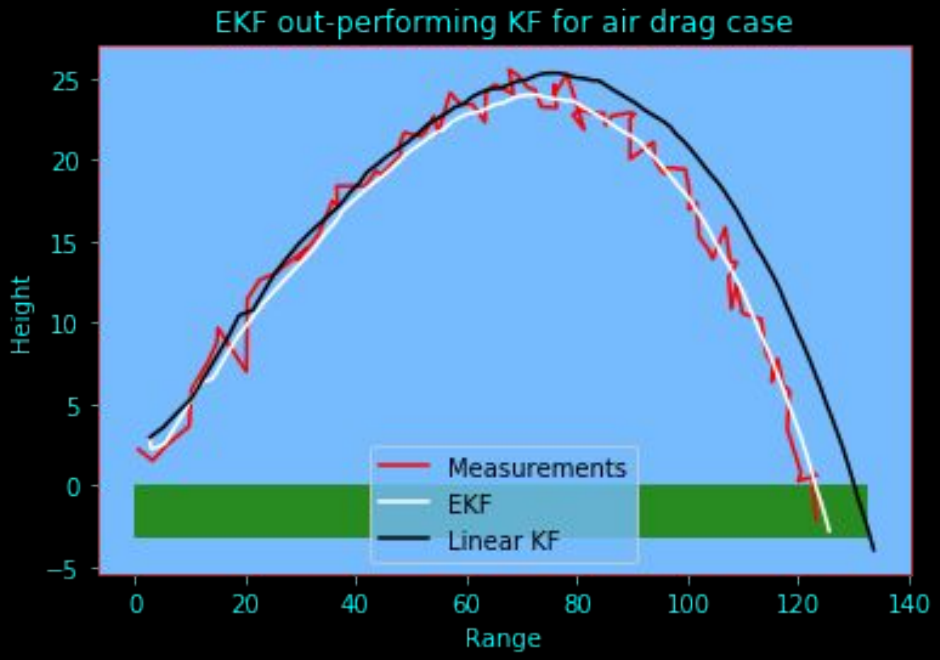
$$v = \sqrt{v_x^2 + v_y^2}$$

Rest all equations remains the same as in the linear KF

Computing the Jacobian : (done using SymPy)

$$A_n = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & \frac{0.00116 dt v_x^2 e^{\frac{\sqrt{v_x^2+v_y^2}}{5}-7}}{\sqrt{v_x^2+v_y^2}\left(e^{\frac{\sqrt{v_x^2+v_y^2}}{5}-7}+1\right)^2} - \frac{0.0058dt}{e^{\frac{\sqrt{v_x^2+v_y^2}}{5}-7}+1} + 1 & 0 & \frac{0.00116 dt v_x v_y e^{\frac{\sqrt{v_x^2+v_y^2}}{5}-7}}{\sqrt{v_x^2+v_y^2}\left(e^{\frac{\sqrt{v_x^2+v_y^2}}{5}-7}+1\right)^2} \\ 0 & 0 & 1 & dt \\ 0 & \frac{0.00116 dt v_x v_y e^{\frac{\sqrt{v_x^2+v_y^2}}{5}-7}}{\sqrt{v_x^2+v_y^2}\left(e^{\frac{\sqrt{v_x^2+v_y^2}}{5}-7}+1\right)^2} & 0 & \frac{0.00116 dt v_y^2 e^{\frac{\sqrt{v_x^2+v_y^2}}{5}-7}}{\sqrt{v_x^2+v_y^2}\left(e^{\frac{\sqrt{v_x^2+v_y^2}}{5}-7}+1\right)^2} - \frac{0.0058dt}{e^{\frac{\sqrt{v_x^2+v_y^2}}{5}-7}+1} + 1 \end{bmatrix}$$

EKF out-performing Linear KF...



Sensor Fusion using Kalman Filtering

- What is different?
  - Multiple sensors.
  - May measure the same state variable or different state variables.

- Can performance be improved using multiple sensors?

- Applications/ Uses?

An example

Train on rails(1-D)

- GPS sensor that gives noisy measurement of position of train.
- Another sensor on the wheels that give the number of rotations of the wheel (also noisy).
- Wheel sensor is twice as noisy as the gps sensor (standard deviation is twice).
- 1 rotation = 2m

Which sensor to choose ?

An example(contd.)

For GPS sensor

$$X_n = AX_{n-1} + v_p$$
$$Y = HX + v_M$$

State Vector : $X = \begin{bmatrix} x \\ v \end{bmatrix}$

Process Noise Matrix : $Q = \begin{bmatrix} \Delta \frac{t^2}{3} & \Delta \frac{t^2}{2} \\ \Delta \frac{t^2}{2} & \Delta t \end{bmatrix} \sigma_P^2$

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

Measurement Noise Matrix : $R = \begin{bmatrix} \sigma_M^2 \end{bmatrix}$

Transition Matrix : $A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$

An example(contd.)

For Wheel Sensor

$$X_n = AX_{n-1} + v_p$$
$$Y = HX + v_M$$

State Vector : $X = \begin{bmatrix} x \\ v \end{bmatrix}$

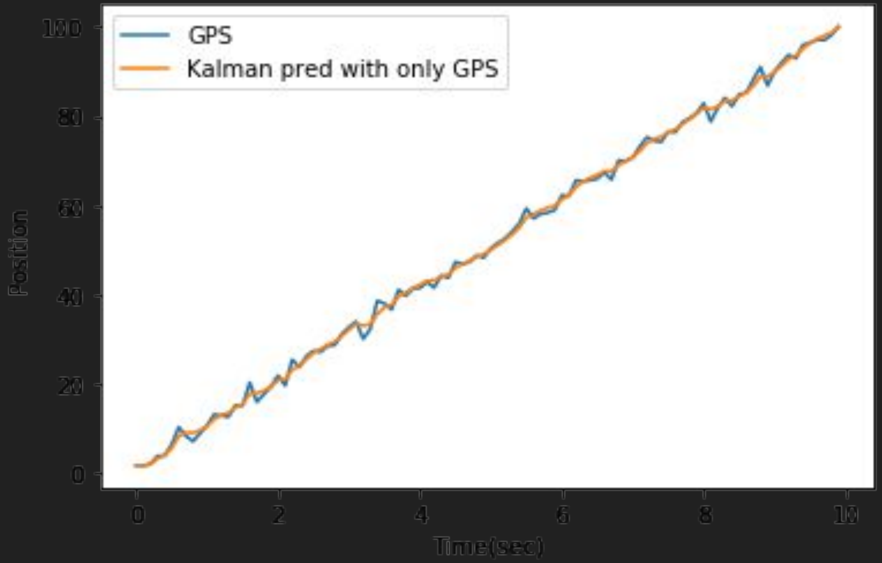
Process Noise Matrix : $Q = \begin{bmatrix} \Delta\frac{t^2}{3} & \Delta\frac{t^2}{2} \\ \Delta\frac{t^2}{2} & \Delta t \end{bmatrix} \sigma_P^2$

Transformation Matrix : $H = \begin{bmatrix} 0.5 & 0 \end{bmatrix}$

Measurement Noise Matrix : $R = \begin{bmatrix} \sigma_M^2 \end{bmatrix}$

Transition Matrix : $A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$

So, which sensor does better?



The real question

Q. Sensor 2 is more noisy than sensor 1. Still, does it resolve any further information even after measurement from sensor 1 is observed ?

Ans: YES!!

Thumb Rule - Never discard any information(even from the inaccurate sensors)

An intuitive example

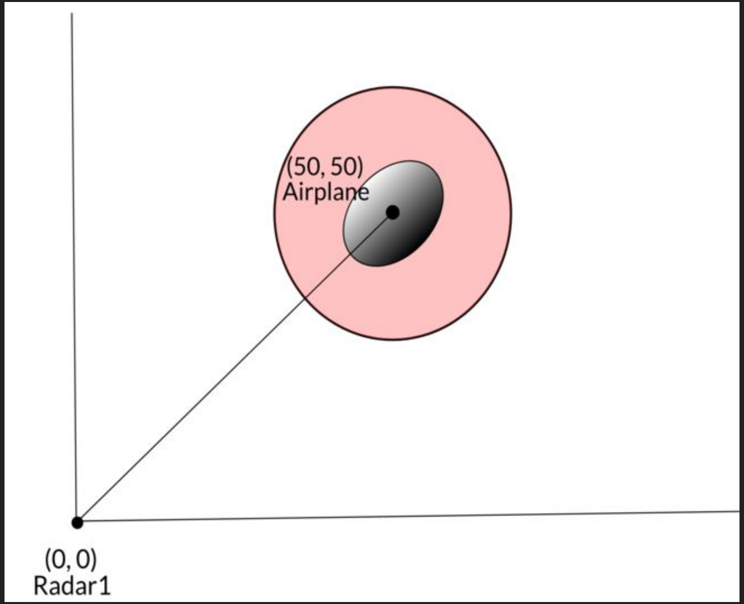
Radars at two stations are being used to track a fighter plane

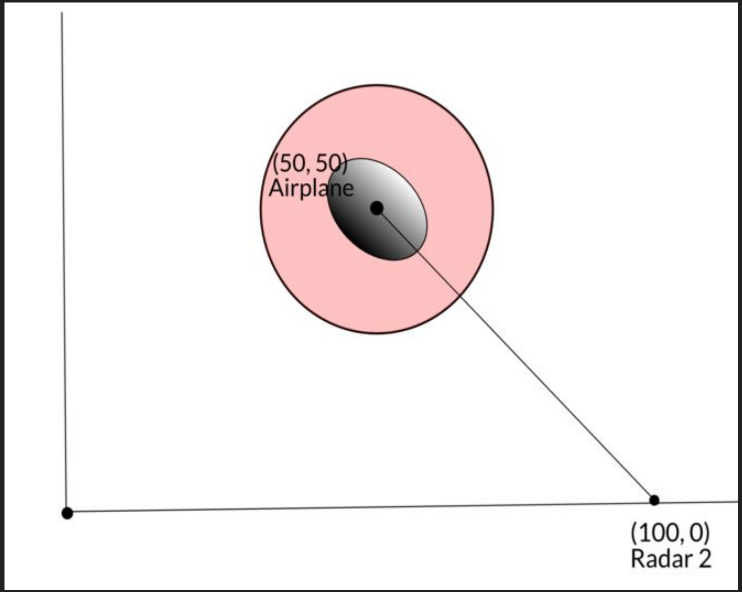
- Radar 1 is at (0, 0)
- Radar 2 is at (100, 0)

Each radar measures the angle and distance of the plane from its station.

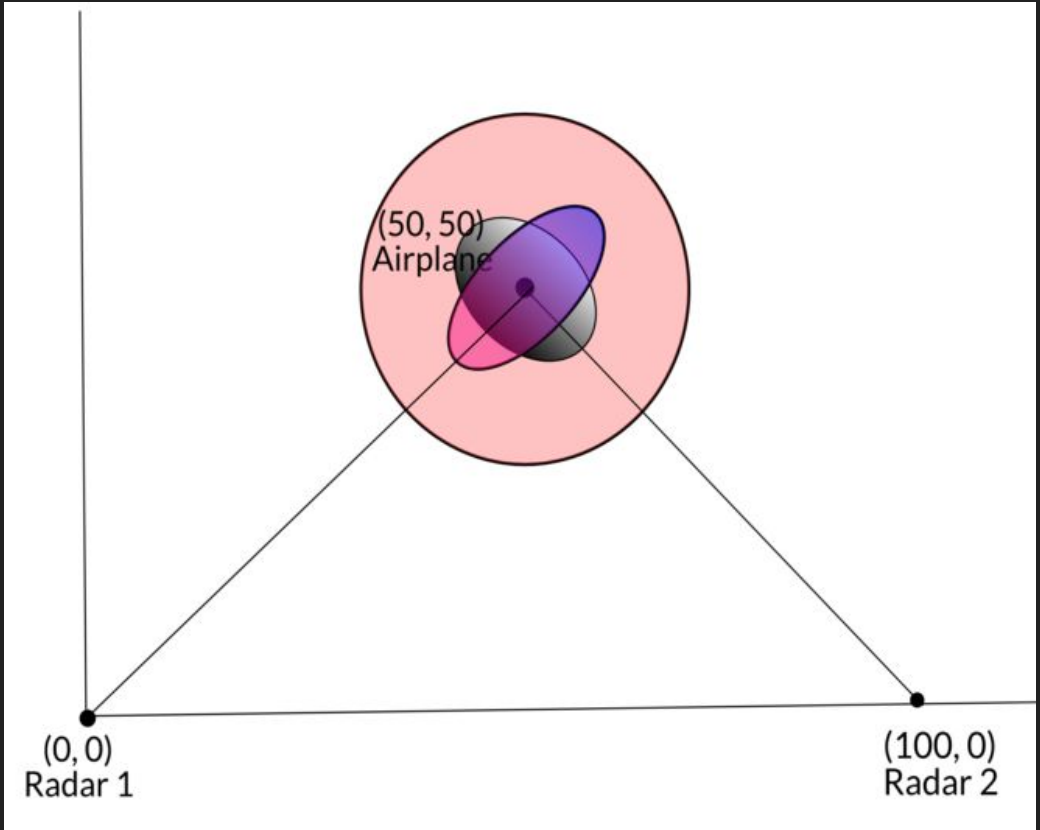
Distance measurement is noisier than angle measurement.

An intuitive example(contd.)





An intuitive example(contd.)



Enough of intuition, let's simulate

For the previous train on rails example

State Vector : $$X = \begin{bmatrix} x \\ v \end{bmatrix}$$

Transformation Matrix : $$H = \begin{bmatrix} 1 & 0 \\ 0.5 & 0 \end{bmatrix}$$

Transition Matrix : $$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

Sensor Inputs : $$Y = \begin{bmatrix} Y_{GPS} \\ Y_{Wheel} \end{bmatrix}$$

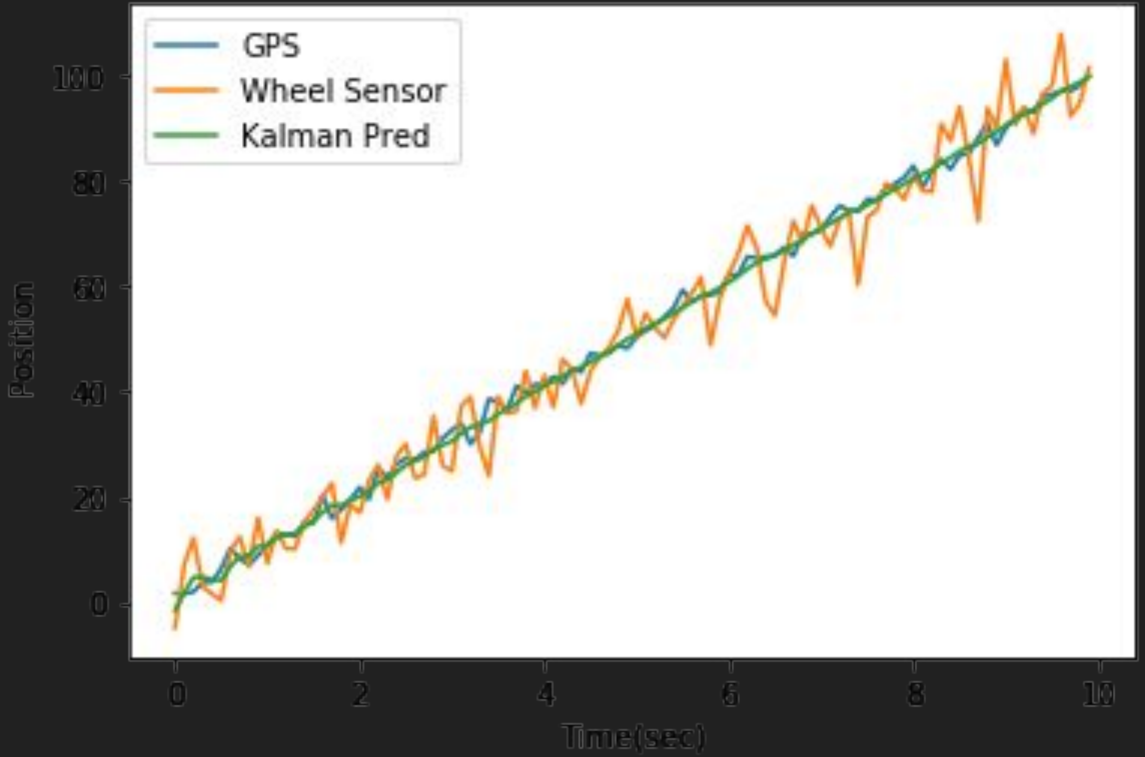
Process Noise Matrix :

$$Q = \begin{bmatrix} \Delta \frac{t^2}{3} & \Delta \frac{t^2}{2} \\ \Delta \frac{t^2}{2} & \Delta t \end{bmatrix} \sigma_P^2$$

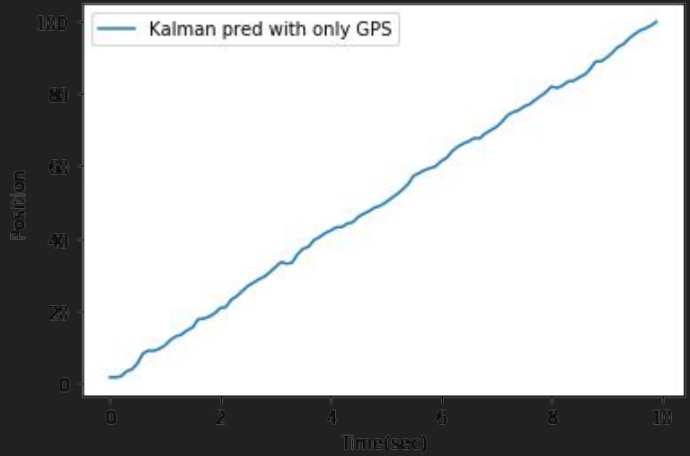
Measurement Noise Matrix :

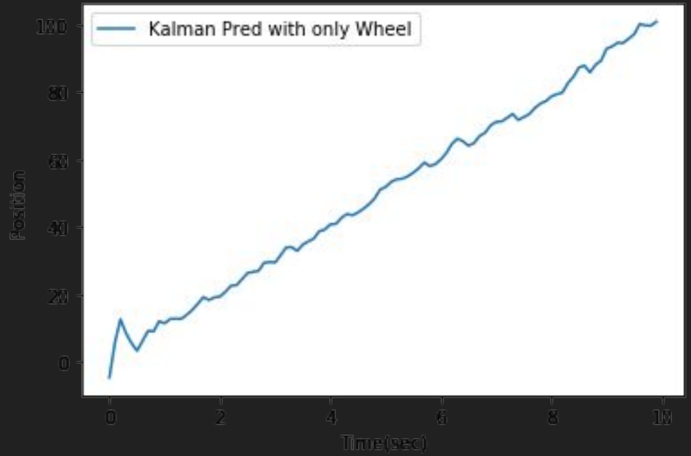
$$R = \begin{bmatrix} \sigma_{GPS}^2 & 0 \\ 0 & \sigma_{Wheel}^2 \end{bmatrix}$$

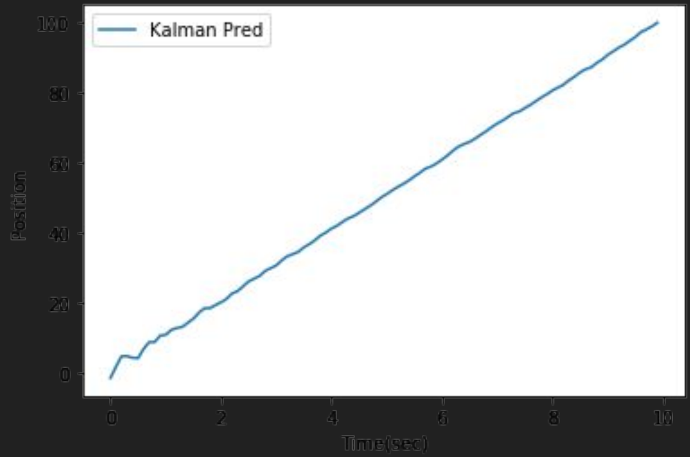
The fused sensor predictions



Comparison


Kalman pred with only GPS
Position
Time(sec)
0
20
40
60
80
100
0
2
4
6
8
10


Kalman Pred with only Wheel
Position
Time(sec)
0
20
40
60
80
100
0
2
4
6
8
10


Kalman Pred
Position
Time(sec)
0
20
40
60
80
100
0
2
4
6
8
10

Applications

- ❖ Aircrafts use it to fuse data from
  - GPS
  - INS
  - Doppler radar,
  - VOR
- ❖ Moisture sensor with a temperature sensor to calculate relative humidity.
- ❖ Calculating orientation of body using gyroscope, magnetometer etc.

Acknowledgement

Blog post on kalman filter mathematics and sensor fusion-

https://medium.com/@cotra.marko/wtf-is-sensor-fusion-part-1-laying-the-mathematical-foundation-89e2d304e23e

A book on kalman filtering and variants -
https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python

"Kalman filter is no rocket science....

(little joke - it is exactly this math that got Apollo to the moon and back)"

- Anonymous

Thank You !!