**AGV SOFTWARE TASK-5**

KALMAN FILTER

-Sanskar Mittal

21CS10057

The Kalman Filter is one of the most important and common estimation algorithms. The Kalman Filter produces estimates of hidden variables based on inaccurate and uncertain measurements. Also, the Kalman Filter provides a prediction of the future system state based on past estimations.

Kalman filter in its most basic form consists of 3 steps.

A) Predict — Based on previous knowledge of a vehicle position and kinematic equations, we predict what should be the position of vehicle after time delta\_t.

B) Measurement — Get readings from sensor regarding position of vehicle and compare it with Prediction

C) Update — Update our knowledge about position (or state) of vehicle based on our prediction and sensor readings.

From usual kinematics equation-

**x = x0 + ut + 0.5at2**

**v = u + at**

Using these equations in 2-D:

1. **Px(t+delta\_t) = Px(t) + delta\_t \* vx + 0.5 \* ax \* delta\_t2**

2. **Py(t+delta\_t) = Py(t) + delta\_t \* vy + 0.5 \* ay \* delta\_t2**

3. **Vx(t+delta\_t) = Vx(t) + ax \* delta\_t**

4. **Vy(t+delta\_t) = Vy(t) + ay \* delta\_t**

Calculations become fairly simple using matrices. So I’ll be using matrix to evaluate results in the Kalman filter.

Since Kalman Filter is a wide topic, I am giving a brief description of the variables I have used in the python file to create the Kalman filter.

**x\_new = A \* x + B \* u**

**x\_new** = [px\_new **A** = [[1 , 0 , delta\_t ,0] **x** = [px,

py\_new [0 , 1 , 0 ,delta\_t] py,

vx\_new [0 , 0 , 1 ,0] vx,

vy\_new] [0 , 0 , 0 ,1]] vy]

**B** = [[0.5\*delta\_t\*delta\_t,0], **u** = [acceleration\_x,

[0,0.5\*delta\_t\*delta\_t], acceleration\_y]

[delta\_t,0],

[0,delta\_t]]

Doing usual matrix multiplication, you can find this represents the kinematics equations only.

The State Covariance Matrix is represented by-

**P** = [[variance\_px, 0, 0, 0],

[0, variance\_py, 0, 0],

[0, 0, variance\_vx, 0],

[0, 0, 0, variance\_vy]]

where variance\_px, variance\_py, variance\_vx, variance\_vy represent uncertainty in predicted

x position, y position, x velocity and y velocity respectively.

Also, a matrix Q has been defined which is Noise Covariance Matrix.

**P = A \* P \* At + Q**

Here,At is transpose of matrix A

The Measurement Covariance Matrix is defined by-

**R** = [[variance\_in\_pos\_x, 0]

[0, variance\_in\_pos\_y]]

To calulate the **Kalman Gain= Uncertainty in predicted state / (Uncertainty in predicted state +**

**Uncertainty in measurement readings)**

We define **H** = [ [1, 0, 0, 0],

[0, 1, 0, 0] ]

So Kalman Gain, **K = ( P \* HT ) / ( ( H \* P \* HT ) + R )**

Let Z be matrix containing the position readings i.e. **Z**=[px,

py]

Also Y is a matrix which gives the difference between measured and predicted values.

So, **Y** = Z – H \* X

The final value of X would then be-

**X = X + K \*Y**

**P = (1-K \* H) \* P**

Summarising all the equations used-

A. Predict:

a. X = A \* X + B \* u

b. P = A \* P \* AT + Q

B. Measurement

a. Y = Z — H \* X

b. K = ( P \* HT ) / ( ( H \* P \* HT ) + R )

C. Update

a. X = X + K \* Y

b. P = ( I — K \* H ) \* P

In the code, I directly wrote the final relations which are a result of Kalman filter.

Also, I took variance in position x & y to be 0.0225 along with noise in a\_x and a\_y to be 5.