%Kalman Filter for FUSION OF ODOMETRY+IMU+GPS

%BY HUNG LA, UNIVERSITY OF NEVADA RENO, SPRING 2020

%Initialization

clc, close all

clear

%Load data and correct IMU initial offset

[time, data] = rtpload('EKF\_DATA\_circle.txt'); %data of the circle in front of Engineering Building

%[time, data] = rtpload('EKF\_DATA\_Rutgers\_ParkingLot.txt'); %data of the circle in front of Engineering Building

%Get Odometry IMU and GPS data (x, y, theta, covariance)

Odom\_x = data.O\_x;

Odom\_y = data.O\_y;

Odom\_theta = data.O\_t;

Gps\_x = data.G\_x;

Gps\_y = data.G\_y;

Gps\_Co\_x = data.Co\_gps\_x;

Gps\_Co\_y = data.Co\_gps\_y;

IMU\_heading = data.I\_t;

IMU\_Co\_heading = data.Co\_I\_t;

%----------------------------------------------

% % Add noise to GPS

% noise\_mean = 0.5;

% noise\_std = 0.1;

% Gps\_noise = noise\_std .\* randn(length(Odom\_x), 2)+ noise\_mean.\*ones(length(Odom\_x), 2);

%

% Gps\_x = data.G\_x +Gps\_noise(:,1);

% Gps\_y = data.G\_y + Gps\_noise(:,2);

%

% Gps\_Co\_x = data.Co\_gps\_x +Gps\_noise(:,1);

% Gps\_Co\_y = data.Co\_gps\_y +Gps\_noise(:,2);

%----------------------------------------------

% % Add noise to Odometry

% noise\_mean = 0.5;

% noise\_std = 0.1;

% Odom\_noise = noise\_std .\* randn(length(Odom\_x), 2)+ noise\_mean.\*ones(length(Odom\_x), 2);

%

% Odom\_x = Odom\_x + Odom\_noise(:,1);

% Odom\_y = Odom\_y + Odom\_noise(:,2);

%---------------------------------------------

% %Add noise to IMU covariance

% noise\_mean = 0.5;

% noise\_std = 0.1;

% IMU\_noise = noise\_std .\* randn(length(Odom\_x), 2)+ noise\_mean.\*ones(length(Odom\_x), 2);

% IMU\_Co\_heading = IMU\_Co\_heading + IMU\_noise(:, 1);

%---------------------------------------------

% Calibrate IMU to match with the robot's heading initially

IMU\_heading = IMU\_heading +(0.32981-0.237156)\*ones(length(IMU\_heading),1); %For EKF\_DATA3

% figure(1), plot(time, data.O\_t)

% figure(2), plot(time, data.I\_t)

% figure(3), plot(data.O\_x, data.O\_y, 'x')

% figure(4), plot(time, IMU\_heading)

%Velocity of the robot

V = 0.14;%0.083;

%Distance between 2 wheel

L = 1; %meter

%Angular Velocity

Omega = V\*tan(Odom\_theta(1))/L;

%set time\_step

delta\_t = 0.001; %0.001

%total=1:delta\_t:length(Odom\_x);

total=1:length(Odom\_x);

%\*\*\*\*\*\*\*\*INITIALIZE STATES\*\*\*\*\*\*\*\*\*\*\*

s.x = [Odom\_x(1); Odom\_y(1); V; Odom\_theta(1); Omega]; %Enter State (1x5)

%Enter transistion matrix A (5x5)

s.A = [1 0 delta\_t\*cos(Odom\_theta(1)) 0 0;

0 1 delta\_t\*sin(Odom\_theta(1)) 0 0;

0 0 1 0 0;

0 0 0 1 delta\_t;

0 0 0 0 1];

%Define a process noise (stdev) of state: (Student can play with this number)

%Enter covariance matrix Q (5x5) for state x

s.Q = [.0004 0 0 0 0; %For EKF\_DATA\_circle

0 .0004 0 0 0;

0 0 .001 0 0;

0 0 0 .001 0;

0 0 0 0 .001];

% s.Q = [.000000004 0 0 0 0; %For EKF\_DATA\_Rutgers\_ParkingLot

% 0 .000000004 0 0 0;

% 0 0 .001 0 0;

% 0 0 0 .001 0;

% 0 0 0 0 .001];

%Define the measurement matricx H:

%Enter measurement matrix H (5x5) for measurement model z

s.H = [ 1 0 0 0 0;

0 1 0 0 0;

0 0 1 0 0;

0 0 0 1 0;

0 0 0 0 1];

%Define a measurement error (stdev)

%Enter covariance matrix R (5x5) for measurement model z

s.R = [.04 0 0 0 0;

0 .04 0 0 0;

0 0 .01 0 0;

0 0 0 0.01 0;

0 0 0 0 .01];

%B matrix initialization:

s.B = [ 1 0 0 0 0;

0 1 0 0 0;

0 0 1 0 0;

0 0 0 1 0;

0 0 0 0 1];

%Enter initial value of u (5x5)

s.u = [0; 0; 0; 0; 0];

%Enter initial covariance matrix P (5x5)

s.P = [.001 0 0 0 0;

0 .001 0 0 0;

0 0 .001 0 0;

0 0 0 .001 0;

0 0 0 0 .001];

%\*\*\*\*\*\*\*\*\*STORE DATA FOR PLOT\*\*\*\*\*\*\*\*\*\*\*

true=[]; % truth voltage

X1=[];

X2=[];

X\_heading=[];

%\*\*\*\*\*\*\*\*START KALMAN FILTER\*\*\*\*\*\*\*\*\*\*\*

for t=1:length(total)

s(t).A = [1 0 delta\_t\*cos(Odom\_theta(t)) 0 0;

0 1 delta\_t\*sin(Odom\_theta(t)) 0 0;

0 0 1 0 0;

0 0 0 1 delta\_t;

0 0 0 0 1]; %Enter transistion matrix A (5x5)

s(t).A = [1 0 delta\_t\*cos(IMU\_heading(t)) 0 0;

0 1 delta\_t\*sin(IMU\_heading(t)) 0 0;

0 0 1 0 0;

0 0 0 1 delta\_t;

0 0 0 0 1]; %Enter transistion matrix A (5x5)

s(t).R = [Gps\_Co\_x(t) 0 0 0 0;

0 Gps\_Co\_y(t) 0 0 0;

0 0 .01 0 0;

0 0 0 IMU\_Co\_heading(t) 0;

0 0 0 0 .01]; %Enter covariance matrix R (5x5) for measurement model z

%

%s(end).z = [Odom\_x(t); Odom\_y(t); V; Odom\_theta(t); Omega]; %Enter State (1x5); % create a measurement

%s(t).z = [Odom\_x(t); Odom\_y(t); V; IMU\_heading(t); Omega]; %Enter State (1x5); % create a measurement

s(t).z = [Gps\_x(t); Gps\_y(t); V; IMU\_heading(t); Omega]; %Enter State (1x5); % create a measurement with adding GPS data

s(t+1)=Kalman\_Filter(s(t)); % perform a Kalman filter iteration

**% All for you is to write a “Kalman Filter”**

%For Plot only

X = s(t).x;

X1(t,:) = X(1,:);

X2(t,:) = X(2,:);

X\_theta = s(t).x;

X\_heading(t,:) = X\_theta(4,:);

% % % %pause(1);

% hold on

% grid on

% % plot Odometry(x,y) data:

% plot(Odom\_x(t), Odom\_y(t), 'r');

% % plot Odometry(x,y) data:

% hold on

% plot(Gps\_x(t), Gps\_y(t), 'k');

% % plot KF estimates:

% hold on

% plot(X1(t), X2(t),'b');

% %legend([hz hgps hk],'Odometry','gps calibrated','Kalman output', 0)

% %title('Fusion of GPS+IMU and ODOMETRY in Position')

% M(t) = getframe;

% hold off

end

%%

% \*\*\*\*\*\*\*\*\*\*\*\*\*Plot the Position\*\*\*\*\*\*\*\*\*\*\*

figure(1)

hold on

grid on

% plot Odometry(x,y) data:

hz = plot(Odom\_x, Odom\_y, '.r');

% plot Odometry(x,y) data:

hgps = plot(Gps\_x, Gps\_y, '.k');

% plot KF estimates:

hk=plot(X1, X2,'.b');

legend([hz hgps hk],'Odometry','gps calibrated','Kalman output', 0)

title('Fusion of GPS+IMU and ODOMETRY in Position')

%\*\*\*\*\*\*\*Plot Heading\*\*\*\*\*\*\*\*\*

figure(2)

hold on

grid on

odom\_heading=plot(time, data.O\_t, 'r');

imu\_heading=plot(time, IMU\_heading, 'k');

KF\_heading = plot(X\_heading, 'b');

legend([odom\_heading, imu\_heading, KF\_heading],'Odometry heading','IMU heading', 'KF heading',0)

title('Fusion of GPS+IMU and ODOMETRY in heading')

%hold off