Grant Davis

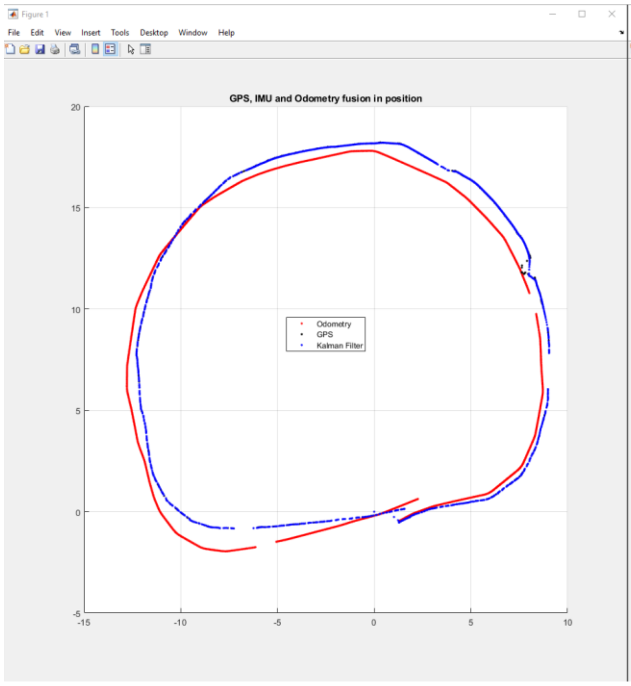
CPE 470

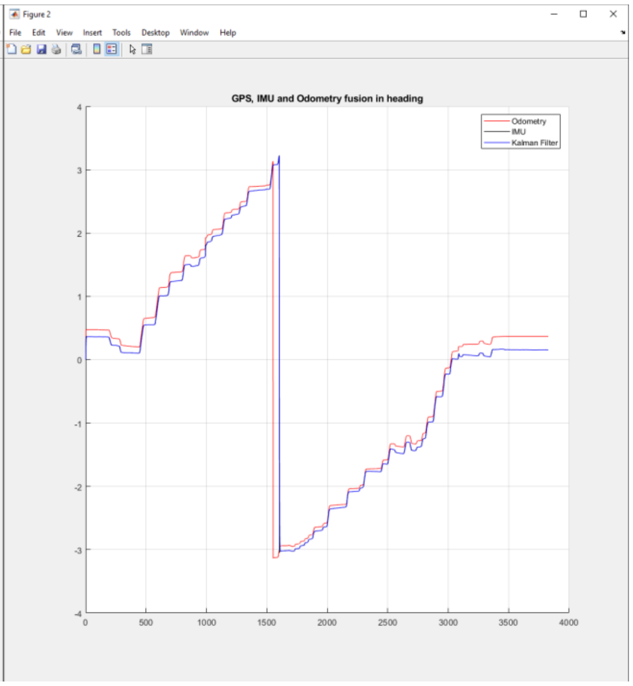
03/01/2022

Project 1. Mobile Robot Localization using Kalman Filter Report

The Kalman Filter is used to optimally estimate the variables of interest when they can't be measured directly, but an indirect measurement is available. Also, when noise is present the Kalman filter is used to find the best estimate of states by combining measurements from the various sensors. Since individual sensors can be unreliable with only their data, Kalman filter must be implemented to compensate. The data for this report comes from GPS, IMU, and Encoder sensors.

Results of Kalman Filter:

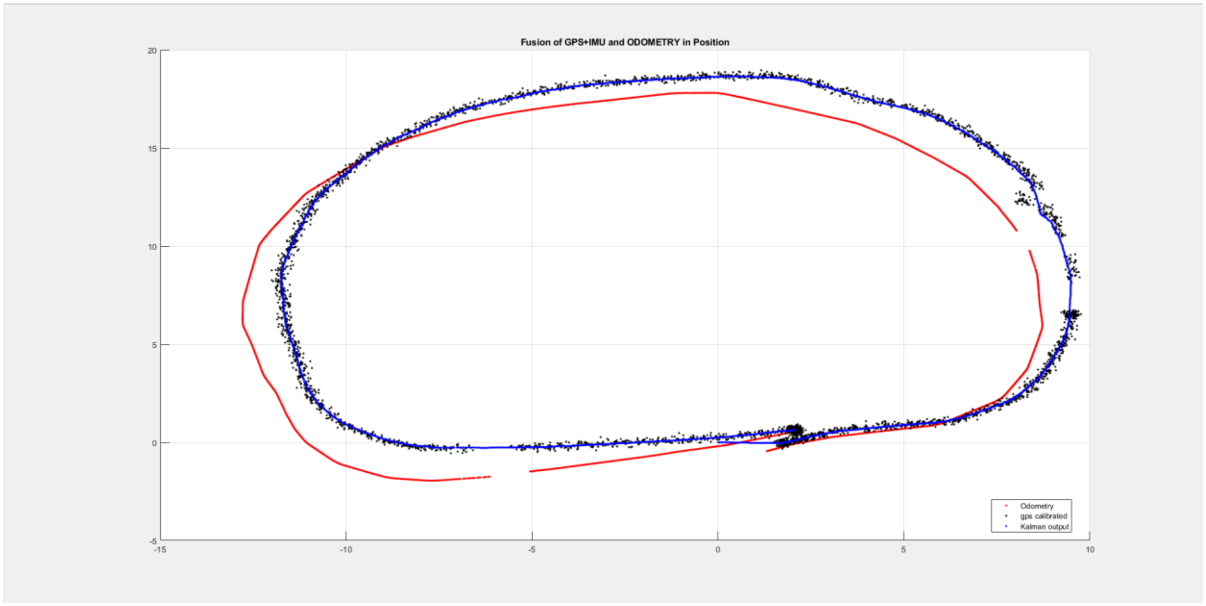




In the results above, the Kalman filter data is shown compared to GPS, IMU, and Odometry. The robot's GPS, IMU, and Odometry data shows the pathing, which creates a circle. The Kalman filter data is nearly the same as the GPS data except when the GPS data becomes unreliable, then the Kalman filter creates an estimate for the current location, using GPS, IMU, and Odometry.

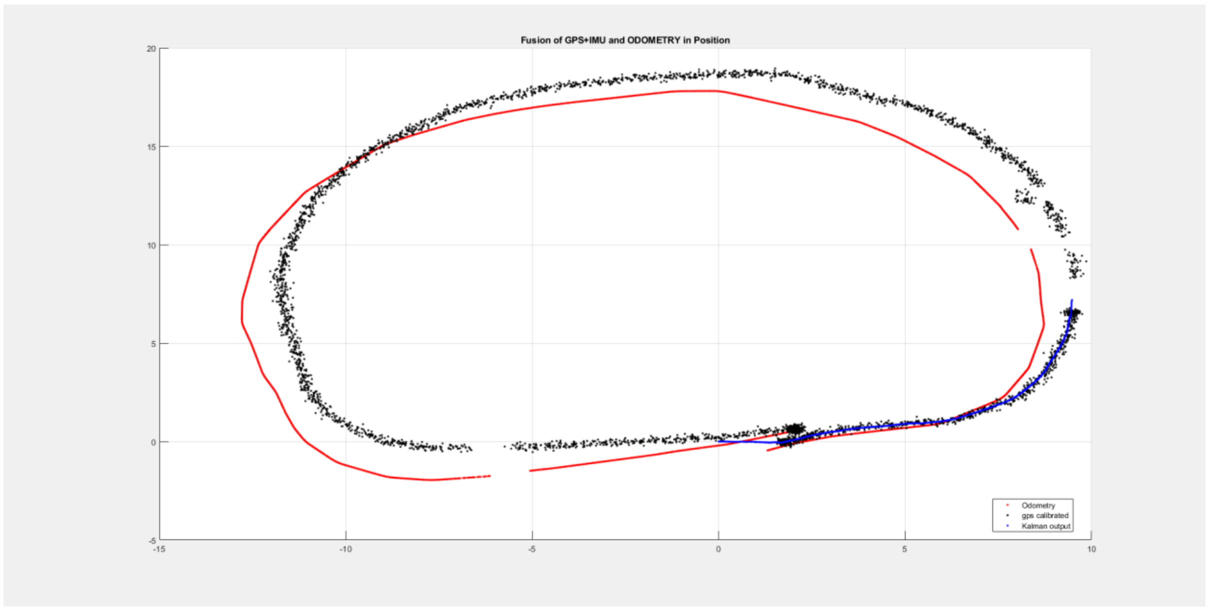
Covariance of GPS sensor data:

Noise to all the data set:



Adding noise to the entire GPS sensor data set, tries to give more accurate estimation for the Kalman Filter. More accuracy allows for better prediction of robot's location throughout the entire data set.

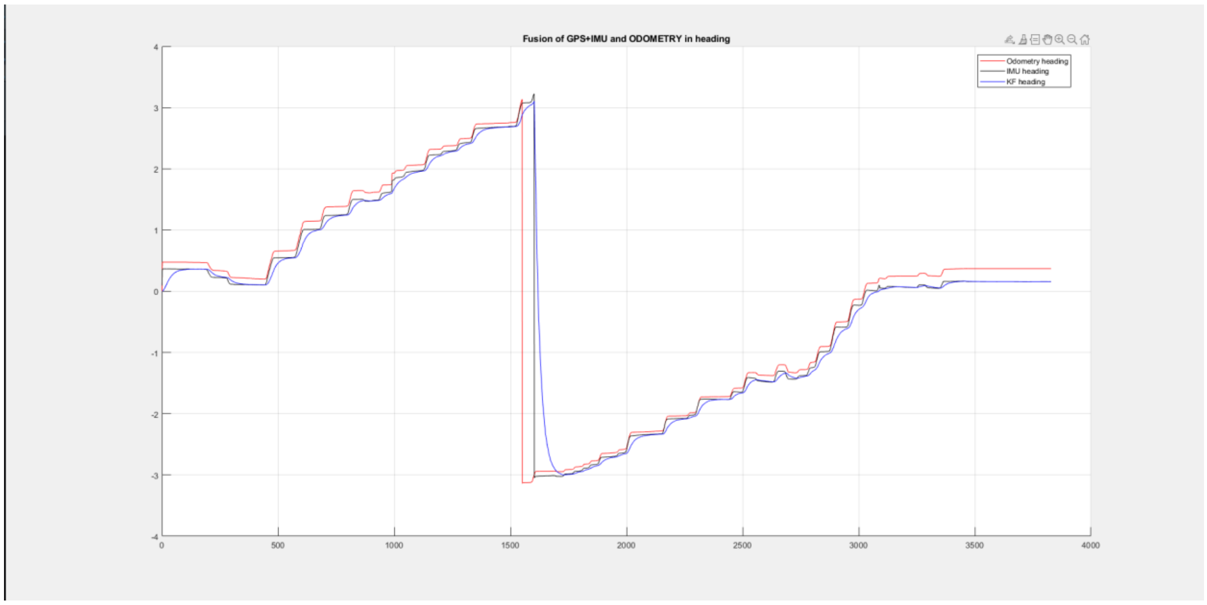
Noise to certain periods of data set:



Adding noise to only part of the GPS sensor data set, by using periods. The accuracy of the Kalman Filter is improved for the periods where the data has noise. Predicting the robot's location in the periods of the data set become more accurate.

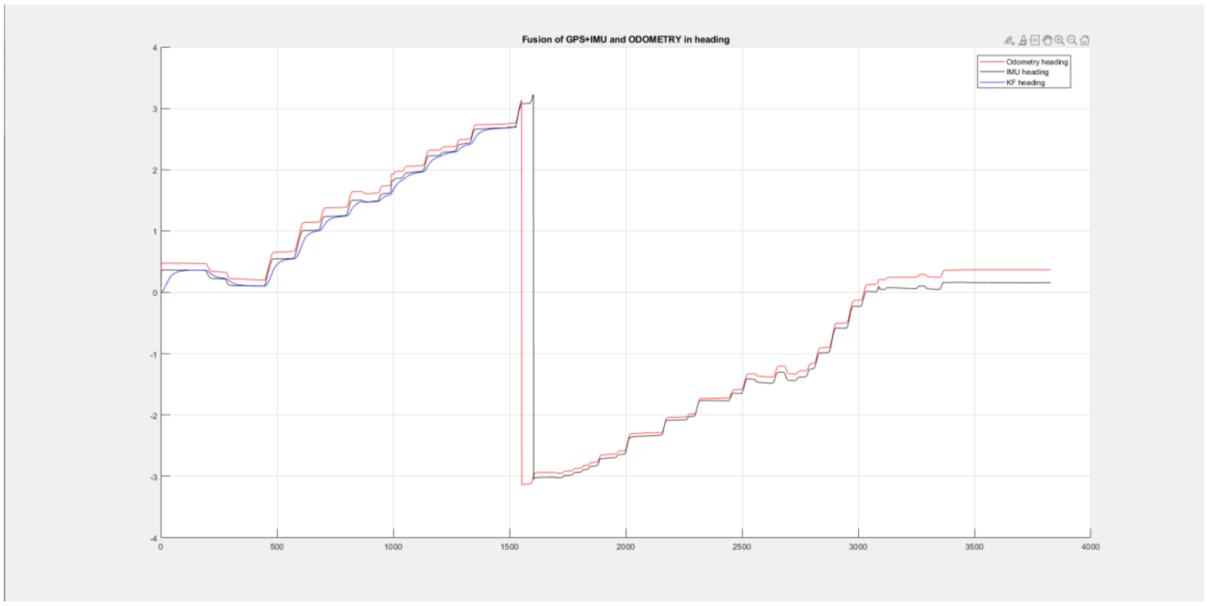
Covariance of IMU sensor data:

Noise to entire data set:



Adding noise to the entire IMU sensor data set, tries to give more accurate estimation for the Kalman Filter. Kalman filter has more IMU sensor data points to work with to produce more accuracy, allowing for better prediction of robot's location throughout the entire data set.

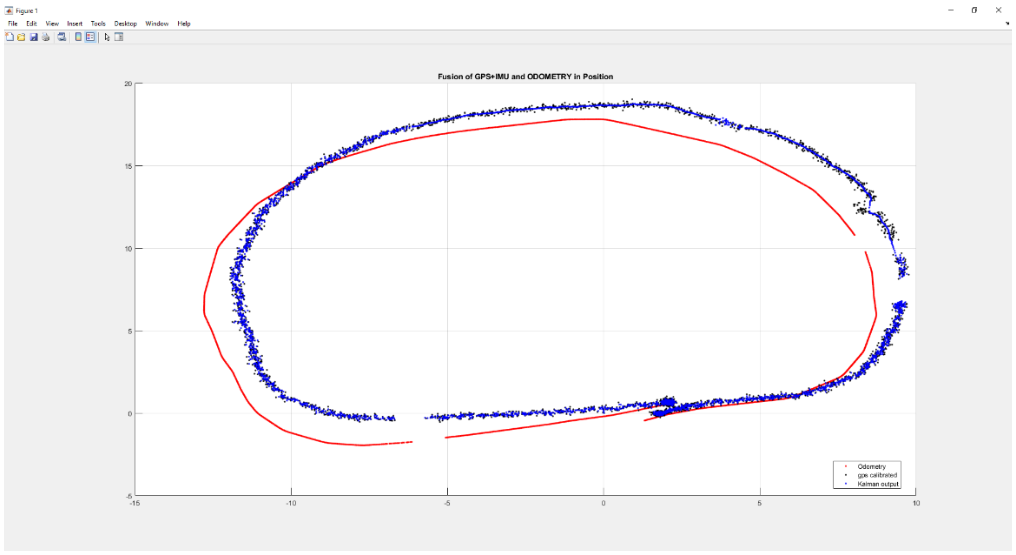
Noise to certain periods of data set:



Adding noise to only part of the IMU sensor data set, by using periods. The accuracy of the Kalman Filter is improved for the periods where the data has noise. Predicting the robot's location in the periods of the IMU sensor data set noise become more accurate.

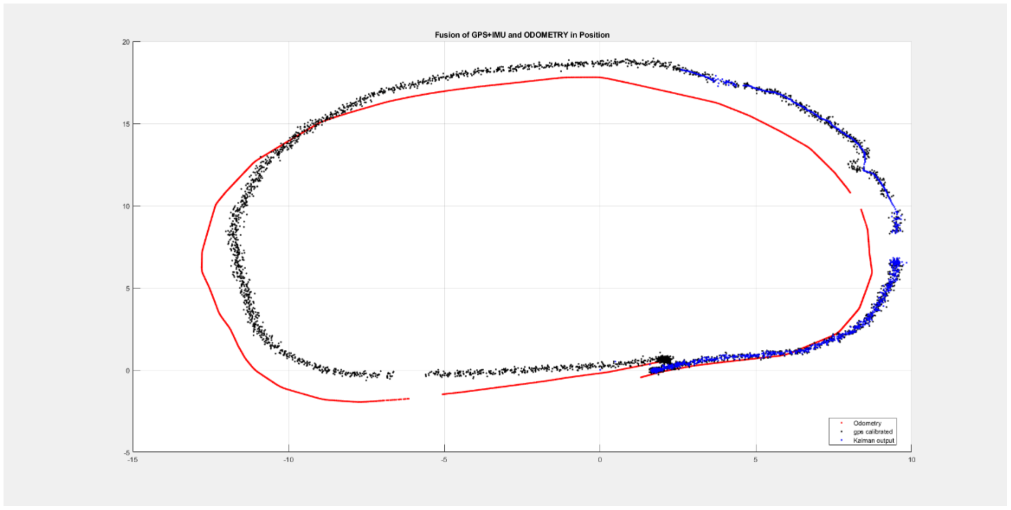
GPS position with changed covariance:

Noise to entire data set:



Adding noise to the entire GPS position data set, tries to give more accurate estimation for the Kalman Filter. Kalman filter has more data points to work with to produce accuracy, allowing for better prediction of robot's location throughout the entire data set.

Noise to certain periods of data set:



Adding noise to only part of the GPS position data set, by using periods. The accuracy of the Kalman Filter is improved for the periods where the data has noise. Predicting the robot's location in the periods of the data set become more accurate.

Appendix: Software Source Code

Kalman Filter (MATLAB source code)

1. main.m

```matlab
%Grant Davis
%CPE 470 Project 1
%03/01/2022

%initialize
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

%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;

noise_mean = 0.5;
noise_std = 0.1;
Gps_noise = noise_std .* randn(length(Odom_x), 2)+ noise_mean.*ones(length(Odom_x),
2);
IMU_noise = noise_std .* randn(length(Odom_x), 2)+ noise_mean.*ones(length(Odom_x),
2);

option = input('Project 1: Mobile Robot Localization using Kalman Filter\n1. Kalman
Filter\n2. GPS Covariance noise\n3. IMU Covariance noise\n4. GPS position w/ changed
covariance\n');
switch option
    case 1
    case 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);
   case 3
          IMU_Co_heading = IMU_Co_heading + IMU_noise(:, 1);
   case 4
          Gps_x = data.G_x + Gps_noise(:,1);
          Gps_y = data.G_y + Gps_noise(:,2);
end
```

```
% Calibrate IMU to match with the robot's heading initially
IMU_heading = IMU_heading +(0.32981-0.237156)*ones(length(IMU_heading),1);

%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];

%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
%Noise option menu
j = length(total) / 4;%partition start pointof data set
        k = length(total) *.8;%partition end point
menu = input('Distribute noise:\n1. period of set\n2. entire set\n');
switch menu
     case 1
        j = length(total) / 4;%partition start pointof data set
        k = length(total) * .4;%partition end point
        for t=1:j
            %Enter transition matrix A(5x5)
            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 transition 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 covariance matrix R(5x5) for measurement model z
            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];
            s(t).z = [Gps_x(t); Gps_y(t); V; IMU_heading(t); Omega];
         %State Prediction
        s(t+1) = Kalman_Filter(s(t));

        X = s(t).x;
        X1(t,:) = X(1,:);
```

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

X_theta = s(t).x;
X_heading(t,:) = X_theta(4,:);
end
for t=j:k %introduce noise
    %Enter transition matrix A(5x5)
    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 transition 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 covariance matrix R(5x5) for measurement model z
    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];
    s(t).z = [Gps_x(t); Gps_y(t); V; IMU_heading(t); Omega];
 %State Prediction
s(t+1) = Kalman_Filter(s(t));

X = s(t).x;
X1(t,:) = X(1,:);
X2(t,:) = X(2,:);

X_theta = s(t).x;
X_heading(t,:) = X_theta(4,:);
end

case 2
for t=1:length(total)
    %Enter transition matrix A(5x5)
    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 transition 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 covariance matrix R(5x5) for measurement model z
    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];
            s(t).z = [Gps_x(t); Gps_y(t); V; IMU_heading(t); Omega];
         %State Prediction
        s(t+1) = Kalman_Filter(s(t));

        X = s(t).x;
        X1(t,:) = X(1,:);
        X2(t,:) = X(2,:);

        X_theta = s(t).x;
        X_heading(t,:) = X_theta(4,:);
        end
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','Location',
'Best')
     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','Location', 'Best')
    title('Fusion of GPS+IMU and ODOMETRY in heading')

%hold off
```

2. rtpload.m

```
function [time, data] = rtpload(filename)
fid = fopen(filename);
if (fid < 0)
  error('Unable to open file %s', filename);
end

line = fgetl(fid);
fclose(fid);
```

```
% Make sure the file contains something.
if (line <0)
  error('Empty file %s', filename);
end

% Load the actual data.
raw = load(filename);

% Restructure the data.
column = 0;
while (~isempty(line))
  [token,line] = strtok(line,'%,');
  column = column+1;
  eval([token ' = raw(:,' num2str(column) ');']);
end

% Move to the correct output variables.  The first column
% is 'time', the rest are 'field.item1' 'field.item2' etc.
time = time;
data = field;
return;
```

3. **Kalman_Filter.m**

```
function [s] = Kalman_Filter(s)
Ak = s.A * s.x;
Bk = s.A * s.P * transpose(s.A) + s.Q;
Ck = Bk * s.H / (s.H * Bk * transpose(s.H) + s.R);
s.x = Ak + Ck * (s.z - (s.H * Ak));
s.P = Bk - (Ck * s.H * Bk);
end
```

Execution Commands

To execute program using provided source code, must use MATLAB environment with all files in the same directory (main.m, rtpload.m, Kalman_Filter.m, and EKF_DATA_circle.txt). Open MATLAB and run main.m, this will prompt you to enter a choice into the command window. Choose your desired option from the UI.