ROBOT LOCALIZATION AND NAVIGATION PROJECT REPORT 2

Kota Naveen Chinta (N14724517)

Project 3 Part 1

To Find: - To develop an Unscented Kalman Filter (UKF) to fuse the inertial data already used in project one and the vision-based pose to find the current and predicted mean.

Given:-

- The Euler angles convention is ZYX to find Rotation Matrix(R) and angular velocity(G).
- A sensor packet for the IMU data is a *struct* that contains the following fields:

```
sensor.is_ready    % True if a sensor packet is available, false otherwise
sensor.t           % Time stamp for the sensor packet, different from the Vicon time
sensor.omg         % Body frame angular velocity from the gyroscope
sensor.acc         % Body frame linear acceleration from the accelerometer
```

- visual pose estimation is given to use as a measurement.

```
pos = proj2Data.position;
pose = proj2Data.angle;
```

Approach to the results of part1 of the project3

Prediction Step

- Initially, we assume that states(x) and noise(q) compute joint Gaussian distribution to approximate the joint Gaussian distribution of x and q.

Unscented Transform-Non Additive noise

The unscented transform (Julier et al., 1995, 2000) is a numerical method for approximating the joint distribution of a Gaussian random variable $\boldsymbol{x} \in \mathrm{R}^n$ and a nonlinear transformation h of it

$$\boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{q}), \qquad \boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \qquad \boldsymbol{q} \sim N(\boldsymbol{0}, \boldsymbol{Q}), \qquad \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} \sim N\left(\begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{m}_U \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{C}_U \\ \boldsymbol{C}_U{}^T & \boldsymbol{S}_U \end{pmatrix}\right)$$

Let the dimensionalities of $\boldsymbol{x}$ and $\boldsymbol{q}$ be n and n_q and let us define $n' = n + n_q$

Let define the augmented random variable $\boldsymbol{x}_{aug} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{q} \end{pmatrix}$ with mean $\boldsymbol{\mu}_{aug} = \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{0} \end{pmatrix}$ and covariance $\boldsymbol{\Sigma}_{aug} = \begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q} \end{pmatrix}$

(i) Where augmented mean is a vector of the previous mean and noise mean (=0).
(ii) Similarly augmented covariance is a vector of previous covariance and noise covariance(Q).

- **Step1:-** Discretizing x_dot as xt using Euler Method

Discretization

$$\dot{\mathrm{x}} = \begin{bmatrix} \mathrm{x}_3 \\ G(\mathrm{x}_2)^{-1}(\omega_m - \mathrm{x}_4 - \mathrm{n}_g) \\ \mathrm{g} + R(\mathrm{x}_2)(\mathrm{a}_m - \mathrm{x}_5 - \mathrm{n}_a) \\ \mathrm{n}_{bg} \\ \mathrm{n}_{ba} \end{bmatrix} = f(x, \boldsymbol{u}, \boldsymbol{n}) \longrightarrow x_t = f(x_{t-1}, \boldsymbol{u}_t, \boldsymbol{n}_t)$$

```
x_t=x_aug+[v;Gi*R*(wm-bg-ng);g+(R*(am-ba-na));nbg;nba]*dt;
```

- **Step2:-** Computing sigma points for every column of the Cholesky factorization of augmented covariance.

Compute Sigma Points with the augmented state slide 26

$$\mathcal{X}^{(0)} = \boldsymbol{\mu}_{aug}, \qquad \mathcal{X}^{(i)} = \boldsymbol{\mu}_{aug} \pm \sqrt{n' + \lambda'}\left[\sqrt{\boldsymbol{\Sigma}_{aug}}\right]_i \qquad i = 1, \ldots n'$$

```
 g= [0; 0; -9.81];
wm = [angVel(1); angVel(2); angVel(3)];
am = [acc(1); acc(2); acc(3)];
alpha=0.001;
kappa=1;
Qt=1;
beta=2;

%computing sigma points
n_dash=27;
lambda = ((alpha)^2)*(n_dash+kappa)-n_dash;
u_aug = [uPrev;zeros(12,1)];
covar_aug = [covarPrev,zeros(15,12);zeros(12,15),eye(12)*Qt];
X_p = [u_aug];
ch_covar = chol(covar_aug,"lower");
X_p = (X_p + [zeros(n_dash,1) sqrt(n_dash+lambda)*ch_covar -sqrt(n_dash+lambda)*ch_covar]);
```

(i) For Gaussian prior distribution good parameters are alpha=0.001, kappa=1, beta=2(given in lecture slides).

(ii) n_dash=n+q, where n=15x1 (number of states) and q=12x1 (mean of all noise (ng,na,nbg,nba))

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \end{bmatrix} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b}_g \\ \mathbf{b}_a \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{orientation} \\ \text{linear velocity} \\ \text{gyroscope bias} \\ \text{accelerometer bias} \end{bmatrix} \in \mathbf{R}^{15}$$

- **Step3:-** Propagating sigma points through the non-linear discretized function.

Propagate Sigma Points through the nonlinear function f slide 26

$$\chi_t^{(i)} = f(\chi_{aug,t-1}^{(i),x}, u_t, \chi_{aug,t-1}^{(i),n}) \qquad i = 0, \dots 2n'$$

```
%propagating sigma points through the non linear function
X_t=[];
for j=1:2*n_dash+1
    x_aug=X_p(1:15,j);
    n_aug=X_p(16:27,j);
    v=[x_aug(7);x_aug(8);x_aug(9)];
    bg=[x_aug(10);x_aug(11);x_aug(12)];
    ba=[x_aug(13);x_aug(14);x_aug(15)];
    ng=[n_aug(1);n_aug(2);n_aug(3)];
    na=[n_aug(4);n_aug(5);n_aug(6)];
    nbg=[n_aug(7);n_aug(8);n_aug(9)];
    nba=[n_aug(10);n_aug(11);n_aug(12)];
    roll=x_aug(4);
    pitch=x_aug(5);
    yaw=x_aug(6);
    G =[0,-sin(yaw),cos(pitch)*cos(yaw);
        0,cos(yaw),cos(pitch)*sin(yaw);
        1,0,-sin(pitch)];
    R =[cos(pitch)*cos(yaw), cos(yaw)*sin(pitch)*sin(roll) - cos(roll)*sin(yaw), sin(roll)*sin(yaw) + cos
        cos(pitch)*sin(yaw), cos(roll)*cos(yaw) + sin(pitch)*sin(roll)*sin(yaw), cos(roll)*sin(pitch)*sir
        -sin(pitch), cos(pitch)*sin(roll), cos(pitch)*cos(roll)];
    Gi=inv(G);
    x_t=x_aug+[v;Gi*R*(wm-bg-ng);g+(R*(am-ba-na));nbg;nba]*dt;
    X_t=[X_t,x_t];
end
```

(i) Iteration through sigma points and substituting it in the discretized function,

- **Step4:-** Computing the predicted mean and covariance.

$$W_0^{(m)\prime} = \frac{\lambda'}{n' + \lambda'} \qquad W_i^{(m)\prime} = \frac{1}{2(n' + \lambda')}. \qquad i = 1, \dots 2n'$$

$$W_0^{(c)\prime} = \frac{\lambda'}{n' + \lambda'} + (1 - \alpha^2 + \beta) \quad W_i^{(c)\prime} = \frac{1}{2(n' + \lambda')}$$

```
%computing the predicted mean,predicted covariance of the
 %meaasurement,and predicted cross-covariance

 W_0m = lambda/(n_dash+lambda);
 W_im = 1/(2*(n_dash+lambda));
 for m= 1:2*n_dash+1
    if m==1
        initial=W_0m*X_t(:,m);
        pinitial= initial;
    else
        remaining=W_im*X_t(:,m);
        pinitial= pinitial+remaining;
    end

 end
 uEst=pinitial;

 W_0c= (lambda/(n_dash+lambda))+(1-(alpha^2)+ beta);
 W_ic=1/(2*(n_dash+lambda));
 for n= 1:2*n_dash+1
    if n==1
         initial_covar=W_0c*(X_t(:,n)-uEst)*(X_t(:,n)-uEst)';
         p_cinitial=initial_covar;
    else
        remaining_covar=W_ic*(X_t(:,n)-uEst)*(X_t(:,n)-uEst)';
         p_cinitial=p_cinitial+remaining_covar;
    end
 end
covarEst=p_cinitial;
```

Update Step

- Fusing the inertial data with the position and orientation of the world. Since position and orientation are linear, the update step of the Kalman filter is used.

Part 1: Vision pose update

$$\boldsymbol{z_t} = \boldsymbol{Cx} + \boldsymbol{\eta} \qquad \boldsymbol{\eta} \sim N(\boldsymbol{0}, \boldsymbol{R})$$

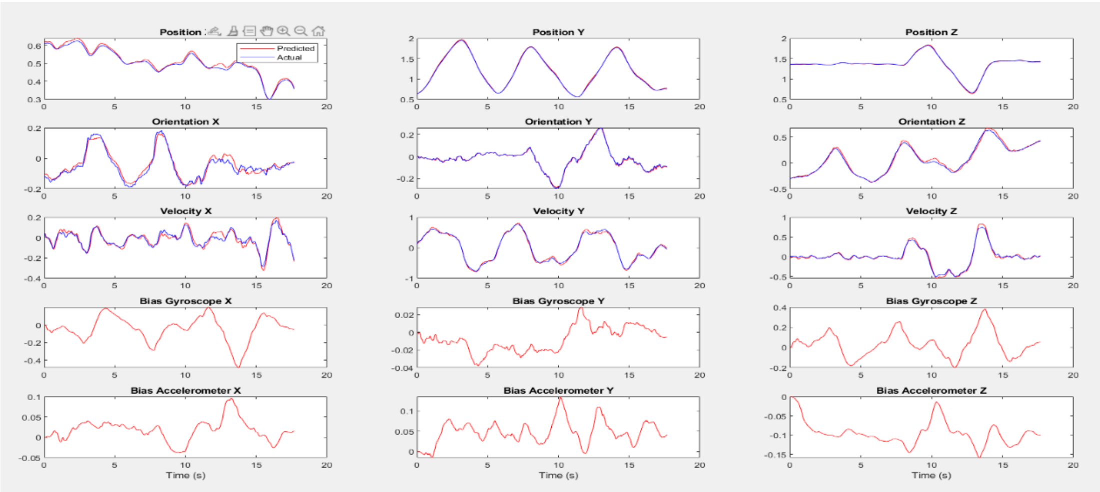
$$\boldsymbol{\mu_t} = \boldsymbol{\bar{\mu}_t} + \boldsymbol{K_t}(\boldsymbol{z_t} - C\,\boldsymbol{\bar{\mu}_t})$$

$$\boldsymbol{\Sigma_t} = \boldsymbol{\bar{\Sigma}_t} - \boldsymbol{K_t}\,\boldsymbol{C}\,\boldsymbol{\bar{\Sigma}_t}$$

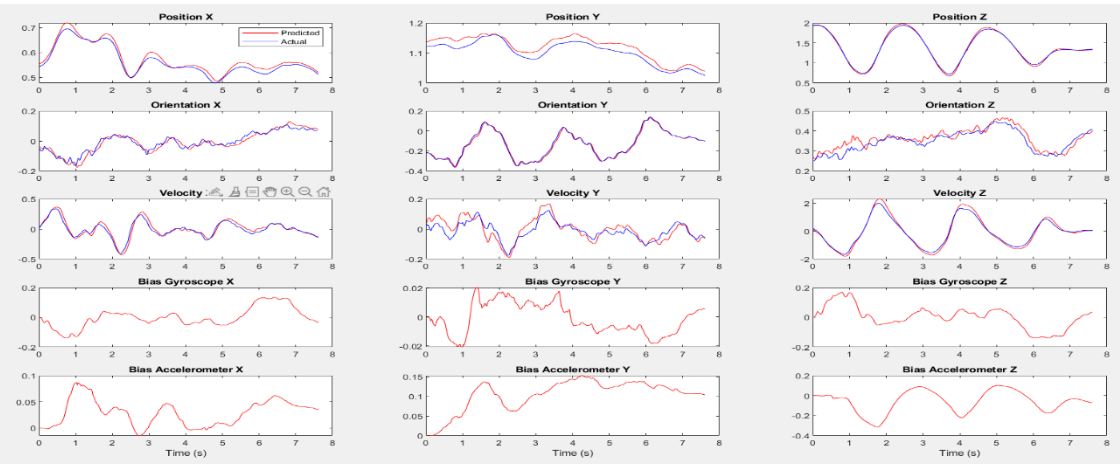
$$\boldsymbol{K_t} = \boldsymbol{\bar{\Sigma}_t}\,\boldsymbol{C^T}\,(\boldsymbol{C}\,\boldsymbol{\bar{\Sigma}_t}\,\boldsymbol{C^T} + \boldsymbol{R})^{-1}$$

```
function [uCurr,covar_curr] = upd_step(z_t,covarEst,uEst)
%% BEFORE RUNNING THE CODE CHANGE NAME TO upd_step
    %% Parameter Definition
    %z_t - is the sensor data at the time step
    %covarEst - estimated covar of the  state
    %uEst - estimated mean of the state
    c=eye(6,15);
    Kt=covarEst*c'*inv(c*covarEst*c'+0.01*eye(6));
    covar_curr=covarEst-Kt*c*covarEst;
    uCurr=uEst+Kt*(z_t-c*uEst);
end
```

Results:-



(a) Dataset 1 of Part 1 project 3



(b) Dataset 4 of Part 1 Project 3

ROBOT LOCALIZATION AND NAVIGATION PROJECT REPORT 2

Kota Naveen Chinta (N14724517)

Project 3 Part 2

To Find:- In this project, you have to develop an Unscented Kalman Filter (UKF) to fuse the inertial data already used in project 1 and the vision-based pose and velocity estimation developed in project 2.

Given:-

- The Euler angles convention is ZYX to find Rotation Matrix(R) and angular velocity(G).
- A sensor packet for the IMU data is a *struct* that contains the following fields:

```
sensor.is_ready    % True if a sensor packet is available, false otherwise
sensor.t           % Time stamp for the sensor packet, different from the Vicon time
sensor.omg         % Body frame angular velocity from the gyroscope
sensor.acc         % Body frame linear acceleration from the accelerometer
```

- Linear and Angular velocity of Camera with respective world expressed in the camera frame.

```
vel = proj2Data.linearVel;
angVel2 = proj2Data.angVel;
```

Approach to the results of part2 of project 3

Prediction Step

- Since we are using the same process model, the prediction step is the same as part 1 of project 3.

Update Step

- Using the rotation and translation of camera with respective body from project 2 to find rotation and translation of body with respective camera.

```
%%Using rotation and translantion of camera w.r.t body from proj 2 to
%%convert into body w.r.t camera
t_cb = [-0.04;0;-0.03];
R_cb = [1.414/2,-1.414/2,0;-1.414/2,-1.414/2,0;0,0,-1];
R_bc = (R_cb)';
H_cb = [R_cb,t_cb;0,0,0,1];
Hb_c = inv(H_cb);
```

$$ {}^{C}\dot{p}_{C}^{W} = R_{B}^{C} l(x_{2},, x_{3}) - R_{B}^{C} S(r_{BC}^{B}) \; R_{C}^{B} \, {}^{C}\omega_{C}^{W} $$

(i) Where X2, X3 is the orientation and linear velocity of the body with respective world.

(ii) Rcb=rotation of camera with respective body.

(iii) Rbc=rotation of body with respective camera.

(iv) Angular velocity of world with respective camera in camera frame is given by angVel2.

(v) rbbc=translation component of body with respective camera.

- **Step 1:-** Computing sigma points for every column of the Cholesky factorization of augmented covariance.

$$z_t = g(x_t, v_t) \qquad v_t \sim N(0, V_t)$$

Let the dimensionalities of x and v be n and n_v, respectively, and $n'' = n + n_v$

Step 1: Compute Sigma Points

Augment the state x_t as $x_{aug} = \begin{pmatrix} x_t \\ v_t \end{pmatrix}$ and form the sigma points for the augmented random variable

Obtained at prediction stage

$$\chi_{aug,t}^{(i)} = \mu_{aug,t} \pm \sqrt{n'' + \lambda''}\left[\sqrt{\Sigma_{t,aug}}\right]_i \qquad \mu_{aug,t} = \begin{pmatrix} \bar{\mu}_t \\ 0 \end{pmatrix} \qquad \Sigma_{t,aug} = \begin{pmatrix} \bar{\Sigma}_t & 0 \\ 0 & V_t \end{pmatrix}$$

```
%computing Sigma Points
n_ddash = 15;
lambda_ddash =(alpha^2 * (n_ddash + kappa)) - n_ddash;
u_auge = uEst;
covar_auge = covarEst;
cho_covar = chol(covar_auge,"lower");
Q =  0.04 * eye(3);
X_i=[u_auge];
X_i=(X_i + [zeros(n_ddash,1) sqrt(n_ddash+lambda_ddash)*cho_covar -sqrt(n_ddash+lambda_ddash)*cho_covar]);
```

- **Step2:-** Propagating sigma points through the non-linear function.

Step 2: Propagate Sigma Points through the nonlinear function g

$$Z_t^{(i)} = g(\chi_{aug,t}^{(i),x}, \chi_{aug,t}^{(i),v}) \qquad i = 0, \dots 2n''$$

```
%propagating sigma points through the non linear function
Z_t=[];
for zu = 1 :(2*n_ddash+1)
   x_auge=X_i(1:15,zu);
   roll = x_auge(4);
   pitch = x_auge(5);
   yaw = x_auge(6);
   R=[cos(pitch)*cos(yaw), cos(yaw)*sin(pitch)*sin(roll) - cos(roll)*sin(yaw), sin(roll)*sin(yaw) + cos(roll)*cos(yaw)*sin(pitc
       cos(pitch)*sin(yaw), cos(roll)*cos(yaw) + sin(pitch)*sin(roll)*sin(yaw), cos(roll)*sin(pitch)*sin(yaw) - cos(yaw)*sin(ro
       -sin(pitch), cos(pitch)*sin(roll), cos(pitch)*cos(roll)];
   zt = (R_cb*R'*x_auge(7:9))-(R_cb*skew(Hb_c(1:3,4))*R_bc*z_t(4:6));
   Z_t=[Z_t zt];
end
```

(i) Multiplied with rotation matrix with Rcb to change from body frame with world frame.

- **Step3:-** Compute the predicted mean, predicted covariance of the measurement and predicted cross-covariance.

Step 3: Compute the predicted mean, predicted covariance of the measurement, and predicted cross-covariance

$$z_{\mu,t} = \sum_{i=0}^{2n''} W_i^{(m)''} Z_t^{(i)} \quad \longleftarrow \quad \text{Update}$$

$$C_t = \sum_{i=0}^{2n''} W_i^{(c)''} \left(\chi_{aug,t}^{(i),x} - \bar{\mu}_t\right)\left(Z_t^{(i)} - z_{\mu,t}\right)^T \qquad S_t = \sum_{i=0}^{2n''} W_i^{(c)''} \left(Z_t^{(i)} - z_{\mu,t}\right)\left(Z_t^{(i)} - z_{\mu,t}\right)^T$$

$$m_U = \sum_{i=0}^{2n'} W_i^{(m)'} y^{(i)} \qquad W_0^{(m)'} = \frac{\lambda'}{n' + \lambda'} \qquad W_i^{(m)'} = \frac{1}{2(n' + \lambda')}. \qquad i = 1, \ldots 2n'$$

$$S_U = \sum_{i=0}^{2n'} W_i^{(c)'} (y^{(i)} - m_U)(y^{(i)} - m_U)^T \quad W_0^{(c)'} = \frac{\lambda'}{n' + \lambda'} + (1 - \alpha^2 + \beta) \quad W_i^{(c)'} = \frac{1}{2(n' + \lambda')}$$

$$C_U = \sum_{i=0}^{2n'} W_i^{(c)'} (x^{(i),x} - \mu)(y^{(i)} - m_U)^T$$

```
 %computing the predicted mean,predicted covariance of the
  %meaasurement,and predicted cross-covariance

W0_mz = lambda_ddash/(n_ddash + lambda_ddash);
Wi_mz = 1/(2 * (n_ddash + lambda_ddash));

for zu=1:(2*n_ddash + 1)
    if zu==1
        initial=W0_mz*Z_t(:,zu);
        z =initial;
    else
        remaining=Wi_mz * Z_t(:,zu);
        z = z +remaining;
    end
end


W0_cz = (lambda_ddash/(n_ddash + lambda_ddash)) + (1 - alpha^2 - beta);
Wi_cz = 1/(2 * (n_ddash + lambda_ddash));

for zu=1:(2*n_ddash + 1)
    if zu == 1
        cinitial=W0_cz*(X_i(1:15 ,zu)-uEst)*(Z_t(:, zu)-z)';
        Ct = cinitial;
        sinitial=W0_cz*(Z_t(:,zu)-z)*(Z_t(:,zu)-z)';
        St = sinitial;
    else
        cremaining=(Wi_cz*(X_i(1:15 ,zu)-uEst)*(Z_t(:,zu)-z)');
```

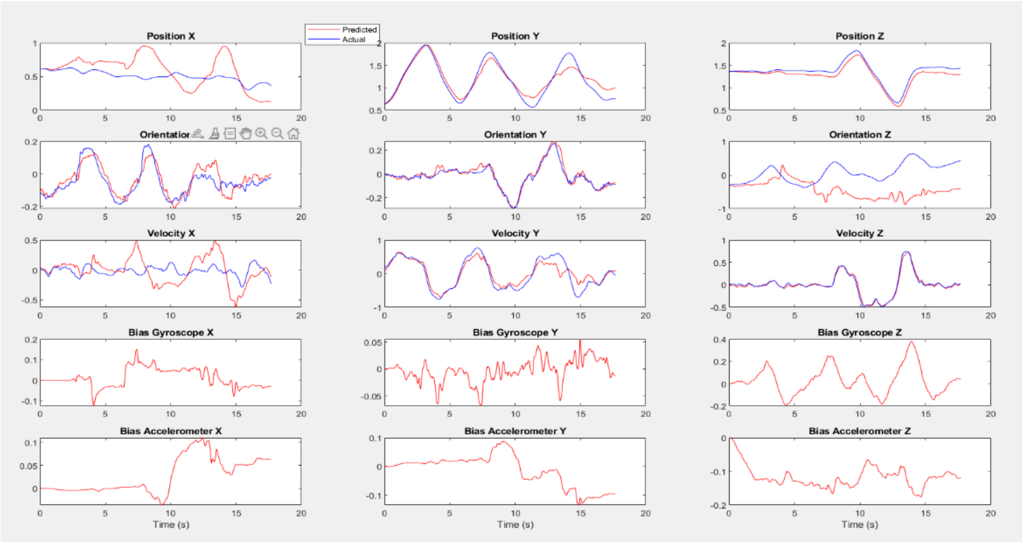
- **Step4:-** Compute the filter gain and the filtered state mean and covariance, conditional to the measurement.

Compute the filter gain and the filtered state mean and covariance, conditional to the measurement

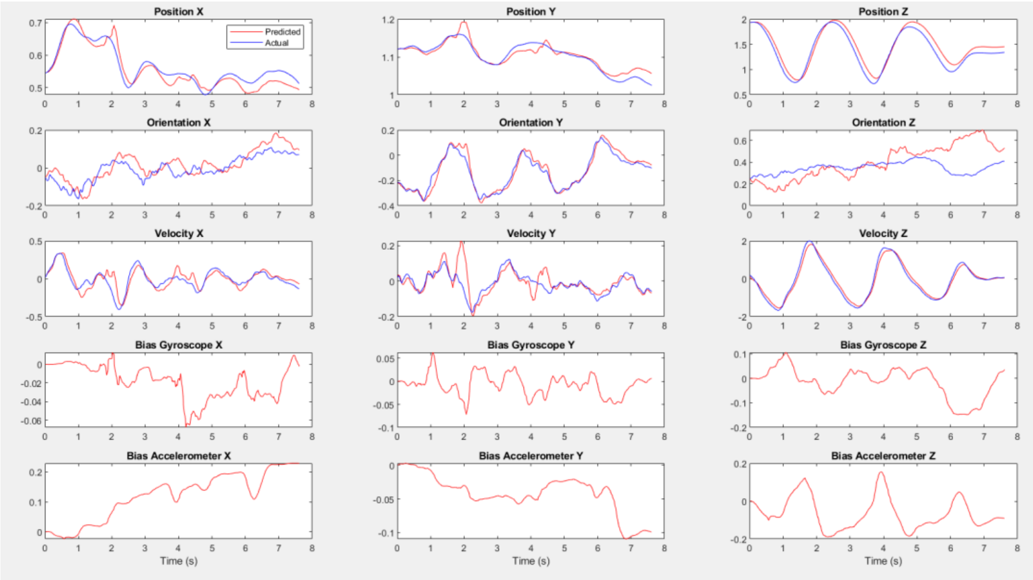
- $\mu_t = \bar{\mu}_t + K_t (z_t - z_{\mu,t})$
- $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$
- $K_t = C_t S_t^{-1}$

```
%Computing the filter gain and the filtered state mean and
% covariance,conditional to the measurement
Kt = Ct/St;
uCurr = uEst+(Kt*(z_t(1:3)-z));
covar_curr =covarEst-(Kt*St*transpose(Kt));
```

Results: -



(c) Dataset 1 of Part2 Project 3



(d) Dataset 4 of Part2 Project 3