Robot Localisation and Navigation Project Report

Extended Kalman Filter for State Estimation

Project 1

Professor: Dr. Giuseppe Loianno

Pavan Chowdary Cherukuri
N10938396
pc3088
22 March 2022

Introduction: This project is performed primarily for state estimation by using an Extended Kalman Filter. The whole code and the plots were made in Matlab 2021b. This Project is divided into two parts, in Part 1, the measurement update will be given by the Position and Orientation from Vicon. In Part 2 only the velocity from the Vicon is used for the measurement update. The given skeleton code and the defined functions were used for the state estimation using EKF. In the further sections, theory and my approach on making the code were explained.

Theory: The goal of this project is state estimation. The state vector that is being estimated is given by 15*1 column vector,

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

The State estimation is performed by using two steps, which basically are the core of the extended kalman filter: The Prediction step and the update step. For performing prediction and update steps, we need to first make our Process model and our observation model.

Process Model:

The process model is given by:

$$\dot{x} = \begin{pmatrix} x_3 \\ G(x_2)^{-1} \cdot R(x_2) \cdot (w_m - x_4 - n_g) \\ g + R(x_2) \cdot (a_m - x_5 - n_a) \\ n_{bg} \\ n_{ba} \end{pmatrix} = \text{f(x,u,n)}$$

Here, the first element in x_dot (or $\dot{x}$) is x3 which is p_dot. The second element will be the differentiation of the ‘q’ vector, by which the orientation of the body is given. Here, the orientation will be given by [roll, pitch, yaw] Euler angles. Hence, to find the Euler angle rates, which in other words is the derivative of ‘q’ (also known as q_dot), G(x2) and R(x2) matrices come in hand. G(x2) matrix (which is defined as ‘GZYX’ in Matlab code) maps the Euler Angle rates (q_dot) to the body's angular velocity in the world frame. GZYX is given as:

GZYX =

$$\begin{pmatrix} \cos(\text{pitch})\cos(\text{yaw}) & -\sin(\text{yaw}) & 0 \\ \cos(\text{pitch})\sin(\text{yaw}) & \cos(\text{yaw}) & 0 \\ -\sin(\text{pitch}) & 0 & 1 \end{pmatrix}$$

Hence the inverse(G(x2)) is multiplied to the body's angular velocity in the world frame to obtain q_dot. The body's angular velocity in the world frame is given by R(x2)*(wm-bg-ng). Where 'wm' is the measured angular velocity from the IMU in the body frame, 'ng' is the noise in the measurement as the gyroscope is assumed to give the noisy estimate of the angular velocity. The matrix R(x2) is multiplied to convert the body's angular velocity to the fixed frame. R(x2) (or RZYX in the code) is given as:

RZYX =

$$\begin{pmatrix} \cos(\text{pitch})\cos(\text{yaw}) & \cos(\text{yaw})\sin(\text{pitch})\sin(\text{roll}) - \cos(\text{roll})\sin(\text{yaw}) & \sin(\text{roll})\sin(\text{yaw}) + \cos(\text{roll})\cos(\text{yaw})\sin(\text{pitch}) \\ \cos(\text{pitch})\sin(\text{yaw}) & \cos(\text{roll})\cos(\text{yaw}) + \sin(\text{pitch})\sin(\text{roll})\sin(\text{yaw}) & \cos(\text{roll})\sin(\text{pitch})\sin(\text{yaw}) - \cos(\text{yaw})\sin(\text{roll}) \\ -\sin(\text{pitch}) & \cos(\text{pitch})\sin(\text{roll}) & \cos(\text{pitch})\cos(\text{roll}) \end{pmatrix}$$

The accelerometer is assumed to give a noisy estimate of the linear acceleration. Hence the third element of x_dot matrix can be given by: g + R(x2)*(am-x5-na). Here 'g' is the acceleration due to gravity vector, 'am' is the measurement given by the IMU, and 'na' is the noise in the measurement as the accelerometer gives the noisy estimate of the acceleration. The drift in the gyroscope bias is assumed to be described by a Gaussian white noise process 'nbg' which is the fourth element of x_dot. The drift in the accelerometer bias is assumed to be described by a Gaussian white noise process 'nba' which is the fifth element of x_dot.
That describes the process model which is also given by 'f(x,u,n)' where 'x' is the state vector, 'u' describes the control inputs, which in our model are the body frame acceleration(am) and the angular velocity(wm) from the onboard IMU and 'n' describes the Gaussian White noise.

Measurement Model: The measurement model is given by "z = C*x + v", where 'v' is the noise in measurement. The measurement model is also defined as function g(x,v). In our Project, there are two parts, hence the two measurement models are used. In part 1, the measurement update will be given by the position and orientation from the vicon. Hence, 'C' Matrix will be a 6*15 matrix given by:

$$C = \begin{pmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \end{pmatrix}.$$

In part 2, the measurement update will be given only by the velocity from the vicon. Hence, 'C' matrix will be a 3*15 matrix given by: C = (0 0 *I* 0 0).

Prediction Step: Since, the Process and Measurement models are known, now the prediction step is performed. The equations involved in the prediction step are mentioned below. 'uEst' given as $\bar{\mu}$ is calculated by the below mentioned formula, where μ_{t-1} is previous mean or 'uPrev' and 'δt' is the difference in time between two steps. '$\bar{\Sigma}$' is the predicted covariance or 'covarEst' and 'Σ_{t-1}' is the previous covariance or the 'covarPrev'. 'At' and 'Ut' are the Jacobian matrices given by the following equations.

Prediction step:

- $\bar{\mu}_t = \mu_{t-1} + \delta t\, f(\mu_{t-1}, u_t, 0)$
- $\bar{\Sigma}_t = F_t\, \Sigma_{t-1}\, F_t^T + V_t\, Q_d\, V_t^T$

- $\dot{x} = f(x, u, n)$
- $n \sim N(0, Q)$

Assumptions

- $A_t = \left.\frac{\partial f}{\partial x}\right|_{\mu_{t-1}, u_t, 0}$
- $U_t = \left.\frac{\partial f}{\partial n}\right|_{\mu_{t-1}, u_t, 0}$

Linearization

- $F_t = I + \delta t\, A_t$
- $V_t = U_t$
- $Q_d = Q\, \delta t$

Discretization

Update Step: The update step is given by the following equations mentioned below. 'μ_t' is the current mean also known as 'uCurr'. 'Σ_t' is the current covariance also known as 'covar_curr'. 'Ct' and 'Wt' are the Jacobian matrices given by the below equations. 'Kt' is the Kalman gain.

Update step:

- $\mu_t = \bar{\mu}_t + K_t\, (z_t - g(\bar{\mu}_t, 0))$
- $\Sigma_t = \bar{\Sigma}_t - K_t\, C_t\, \bar{\Sigma}_t$
- $K_t = \bar{\Sigma}_t\, C_t^T\, (C_t\, \bar{\Sigma}_t\, C_t^T + W_t\, R\, W_t^T)^{-1}$

- $z = g(x, v)$
- $v \sim N(0, R)$

Assumptions

- $C_t = \left.\frac{\partial g}{\partial x}\right|_{\bar{\mu}_t, 0}$
- $W_t = \left.\frac{\partial g}{\partial v}\right|_{\bar{\mu}_t, 0}$

Linearization

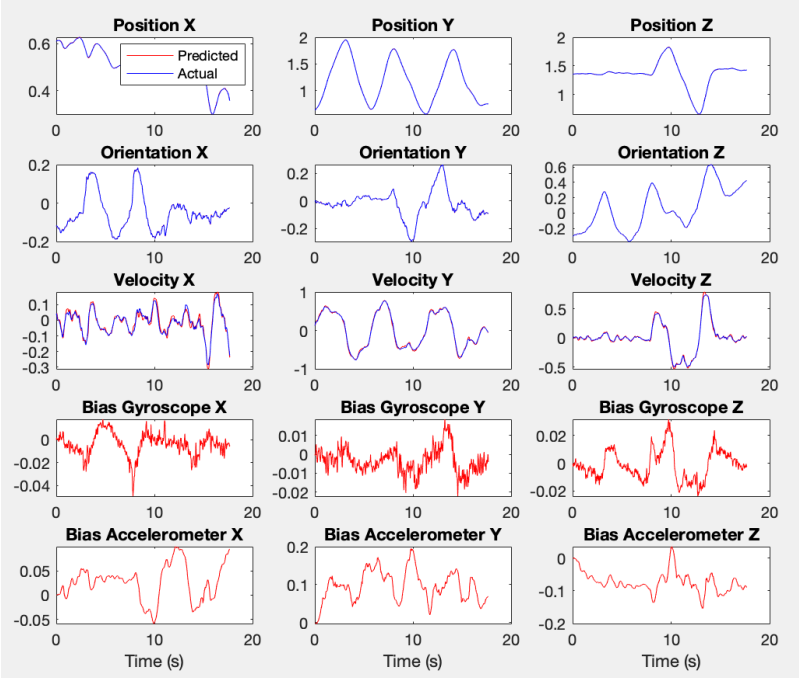
Approach and Coding: In this section the approach to solve the given model and how the code is made and guidelines to access the code are mentioned. Firstly, there is a file attached in addition to the main files, which is known as 'Code_for_Jacobian_Matrices.m'. This file consists of the approach and the methodology, equations used to calculate all the non-zero elements of Jacobian Matrices used in solving the E.K.F. These elements and matrices are used in the matlab code. Initially in the prediction step function, all the required elements are extracted from the input and 'RZYX' , 'GZYX' matrices are made.Then the Jacobian matrix 'dfdx' is defined. All the non-zero elements and zero elements were clearly indicated and commented on in the code. In the similar manner, the Jacobian matrix 'dfdn' is also calculated. Then the required variables were defined and equations were written as above discussed in the prediction step . Here, the covariance 'Q' is initialized as an identity matrix multiplied by a value '0.1'. Note here is by altering the factor multiplied, the prediction is subject to change. In the update function, the Jacobian matrix 'Ct' is defined, which is calculated in a similar manner as mentioned above. 'g($\bar{\mu}_t$,0)' is defined as the 'g_uEst' variable which is extracted from uEst. Since there are two parts in the project, 'Ct' and 'g_uEst' values vary in these parts. In the first part 'g_uEst' consists of the first six elements from 'uEst' and in the second part 'g_uEst' contains 7,8,9 elements from'uEst'. Covariance of noise given by 'R' is defined as an identity matrix multiplied by a value '0.01'. Similar to 'Q', if 'R' is varied, the model is subject to change. It is suggested that these values be optimized, but in this scenario there isn't much optimization performed as the plots look good with these values and there isn't much change observed. The required equations were written further in the function code. In the Kalman Filter code, the prediction and update functions were called in the given for loop. The inputs and outputs of these functions were clearly commented on in the code.

After the functions are called, the 'prevTime' variable is re-initialised to calculate 'dt' in the next iteration. The calculated 'uCurr' value is saved to the 'savedStates' matrix, which is further used for plotting. Finally the 'uPrev' and 'covarPrev' variables are re-initialized to 'uCurr' and 'covar_curr' for the next iteration. In the next section, the results were shown in the form of plots for both the parts of the project consisting 3 datasets in each part.

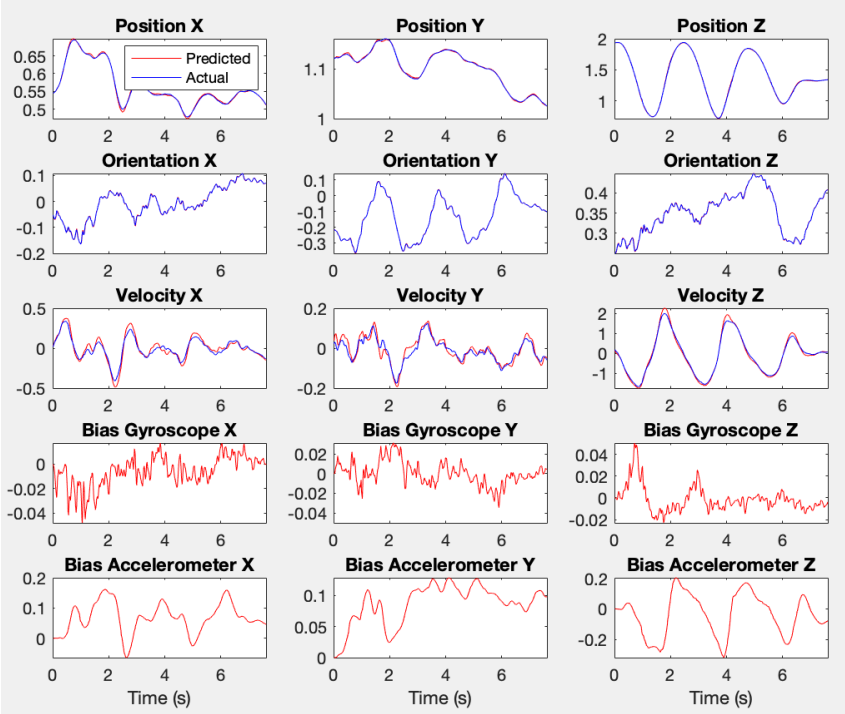
Results:

Part1:

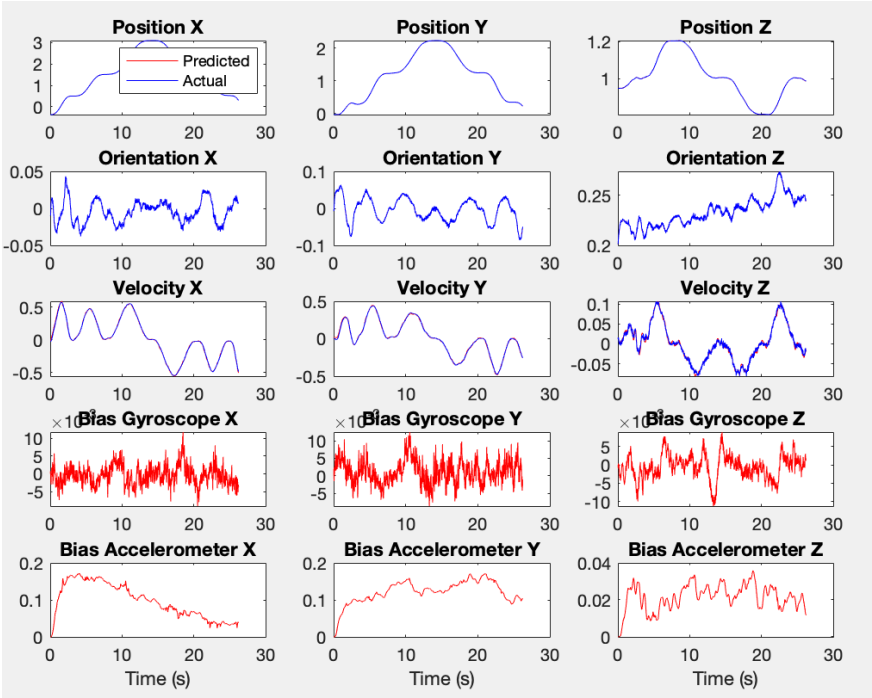
DataSet1:



DataSet4:

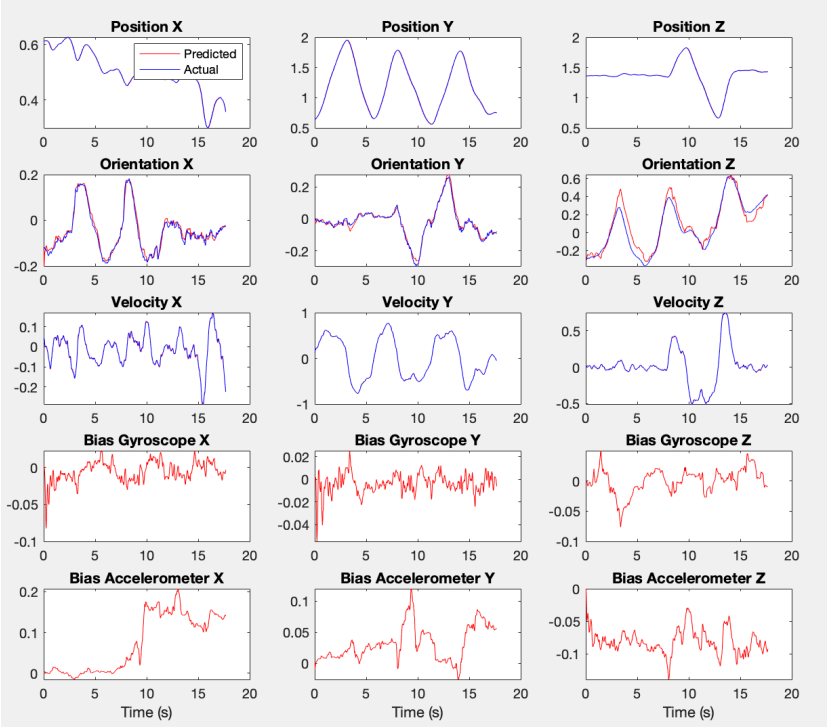


DataSet9:

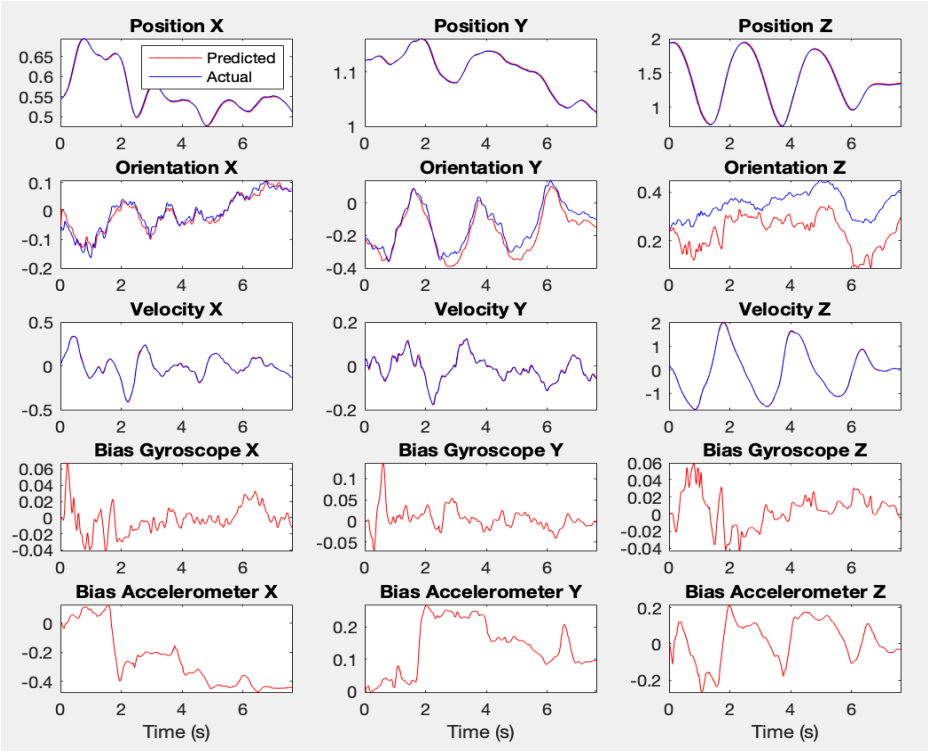


Part2:

Dataset1



Dataset4:



DataSet9: