Robot Localization and Navigation Project 1

Ali Rasteh
netID: ar7655

March 12, 2024

Problem Definition

In this project, we're going to estimate the state of a robot including its position, orientation, linear velocity, and bias of its gyroscope and accelerometers. To do this we will use measurements from the Vicon system along with a process model for the system state. We'll also use the measurement from onboard IMU (Inertial Measurement Unit) to estimate the input to the system. So we are actually combining the measurements from the IMU with the Vicon motion capture system measurements. To present the system the ZYX Euler representation is chosen and used. The project consists of two parts. In the first part, we'll use just the position and orientation measurements from the Vicon and in the second part, we'll use the velocity measurements.

Problem Formulation

Process Model

The following state vector is used as the state in system dynamics for the purpose of this project.

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \end{bmatrix} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b}_g \\ \mathbf{b}_{\boldsymbol{a}} \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{orientation} \\ \text{linear velocity} \\ \text{gyroscope bias} \\ \text{accelerometer bias} \end{bmatrix} \in \mathbf{R}^{15} \tag{1}$$

As mentioned in the course slides, the following formulation is used as the system dynamics equations for both part 1 and part 2 in this project.

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_3 \\ \mathbf{G}\left(\mathbf{x}_2\right)^{-1}\left(\boldsymbol{\omega}_m - \mathbf{x}_4 - \mathbf{n}_g\right) \\ \mathbf{g} + R\left(\mathbf{x}_2\right)\left(\mathbf{a}_m - \mathbf{x}_5 - \mathbf{n}_a\right) \\ \mathbf{n}_{bg} \\ \mathbf{n}_{ba} \end{bmatrix} = f(x, u, n) \tag{2}$$

where, $\mathbf{n}_a$ and $\mathbf{n}_g$ represent the accelerometer and gyroscope noise terms and $\mathbf{n}_{ba}$ and $\mathbf{n}_{bg}$ represent the dynamics of the accelerometer and gyroscope biases. As mentioned in the project the noise covariance matrix or noise standard deviation values should be tuned in a way that the estimation works well with all 3 datasets. However, in the real-world scenario, these Gaussian noise variances should be provided by the manufacturer of the sensors and used in the dynamics. However, in this project, we did according to the requirements and tuned the covariance matrix and our defined values could be observed in "pred_step.m". Furthermore, the accelerometer and gyroscope measurements are considered as the inputs to the process model and are indicated by $\boldsymbol{\omega}_m$ and $\mathbf{a}_m$ in Eq.2. Still $\mathbf{G}\left(\mathbf{x}_2\right)$ and $\mathbf{R}\left(\mathbf{x}_2\right)$ should be defined in the process equations. The $\mathbf{R}\left(\mathbf{x}_2\right)$ is the rotation matrix corresponding to $\mathbf{x}_2$ or the orientation of the system which are the Euler angles along Z, Y, and X axis. The $\mathbf{R}$ matrix for the ZYX rotation could be derived by multiplying the basic rotation matrices along each of the axes as follows.

$${}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}(\psi, \theta, \phi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{3}$$

where $\mathcal{W}$ represents the world inertial frame and $\mathcal{B}$ is the body frame fixed to the IMU. In this project, the IMU frame coincides with the robot body frame, and no more rotation is needed. A rotation matrix ${}^{A}\mathbf{R}_{B}$ rotates the frame A to align with frame B. So in Eq. 3 we first rotate the inertial frame around its z-axis for ψ radians and then around the y-axis of the resulting frame for θ radian and finally around the x-axis of the result for ϕ radians. In our codes, we use the "eul2rotm_zyx and" MATLAB builtin "eul2rotm" functions to do rotations.
The $\mathbf{G}(\psi, \theta, \phi)$ matrix maps the rate of the Euler parameters $\dot{\psi}, \dot{\theta}, \dot{\phi}$ to the body angular velocity of frame $\mathcal{B}$ with respect to frame $\mathcal{W}$ and expressed in frame $\mathcal{W}$, ${}^{\mathcal{W}}\boldsymbol{\omega}_{\mathcal{B}}^{\mathcal{W}}$.

$$ {}^{\mathcal{W}}\boldsymbol{\omega}_{\mathcal{B}}^{\mathcal{W}} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \mathbf{G}(\psi, \theta, \phi) \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{4} $$

The expression for this mapping is derived based on the law of adding angular velocities. For the ZYX convention in this project, it can be derived as follows:

$$ \begin{aligned} \mathbf{G}(\psi, \theta, \phi) = & \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dot{\psi} \\ & + \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \dot{\theta} \\ & + \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \dot{\phi} \end{aligned} \tag{5} $$

The Eq. 5 we first express the rotation rate around the z-axis, $\dot{\psi}$ in the world frame, then we express the rotation rate around the y-axis, $\dot{\theta}$ in the world frame, and finally we express the rotation rate around the x-axis, $\dot{\phi}$ in the world frame. We sum all of these rotation rates expressed in the world frame to find the complete transformation matrix. The resulting expression is then computed as follows:

$$ \mathbf{G}(\psi, \theta, \phi) = \begin{bmatrix} cos(\theta)cos(\psi) & -sin(\psi) & 0 \\ cos(\theta)sin(\psi) & cos(\psi) & 0 \\ -sin(\theta) & 0 & 1 \end{bmatrix} \tag{6} $$

It's very important to note that the gyroscope, measures the body angular velocity as expressed in the body frame, ${}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}}^{\mathcal{W}}$, while Eq.4 expresses the Euler parameter rates to the body angular rate expressed in the world frame. So we should pre-multiply $\mathbf{G}(\psi, \theta, \phi)$ by the inverse of $\mathbf{R}$ shown in Eq. 3 to find the right and final transformation matrix. So the final $\mathbf{G}$ matrix is as follows and appears the same in our codes.

$$ \mathbf{G}(\psi, \theta, \phi) = \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\phi) & sin(\phi)cos(\theta) \\ 0 & -sin(\phi) & cos(\phi)cos(\theta) \end{bmatrix} \tag{7} $$

Measurement Model for Part1

In the first part of this project, the measurements are assumed to be the position and orientation of the robot reported by the Vicon system. These measurements are directly accessible in the states, so the measurement model is simply as follows. It simply extracts the first 6 items in systems states which correspond to the position and orientation of the drone.

$$ \begin{aligned} \mathbf{C} &= \begin{bmatrix} \mathbf{I}_{6\times 6} & \mathbf{0}_{6\times 9} \end{bmatrix}, \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + v \end{aligned} \tag{8} $$

Measurement Model for Part2

In the second part of this project, the measurements are the velocity of the drone reported by the Vicon system. As this measurement is also directly accessible in system states, the measurement model for this part is also a simple linear model that just extracts the velocity from the states as follows:

$$ \begin{aligned} \mathbf{C} &= \begin{bmatrix} \mathbf{0}_{3\times 6} & \mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 6} \end{bmatrix}, \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + v \end{aligned} \tag{9} $$

Extended Kalman Filter Equations

As the state dynamics in this project are nonlinear we need to use the Extended Kalman Filter as the estimating filter. There are two steps, known as prediction and update in the Extended Kalman Filter and we are going to explain each of them in the following sections.

Prediction

In the first step known as prediction, we predict the expected values for the mean and covariance of the system states in the current time step and to do that we use the mean and covariance of the previous time step with the linearized dynamics of the system. The following equation shows the procedure.

$$\begin{aligned} \bar{\boldsymbol{\mu}}_t &= \boldsymbol{\mu}_{t-1} + \delta t \mathbf{f}\left(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, 0\right) \\ \bar{\boldsymbol{\Sigma}}_t &= \mathbf{F}_t \boldsymbol{\Sigma}_{t-1} \mathbf{F}_t^T + \mathbf{V}_t \mathbf{Q}_d \mathbf{V}_t^T \end{aligned} \tag{10}$$

$\boldsymbol{\mu}, \boldsymbol{\Sigma}$ represent the mean and covariance of the state and $\mathbf{u}_t$ consists of the IMU measurements as input. The $\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}$ represent the predicted value for the mean and covariance of the states in the current time step. δt specifies the sampling period. The matrices $\mathbf{F}_t$ and $\mathbf{U}_t$ in Eq.10 are the discretized version of the jacobians of the nonlinear state dynamics with respect to the state and noise and could be computed as follows:

$$\begin{aligned} \mathbf{F}_t &= \mathbf{I} + \delta t \mathbf{A}_t = \mathbf{I} + \delta t \frac{\partial \mathbf{f}}{\partial \mathbf{x}}|_{\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, 0} \\ \mathbf{U}_t &= \frac{\partial \mathbf{f}}{\partial \mathbf{n}}|_{\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, 0} \\ \mathbf{V}_t &= \mathbf{U}_t \end{aligned} \tag{11}$$

Furthermore, $\mathbf{Q}_d = \delta t \mathbf{Q}$ is the process covariance of the linearized and discredited system. In our codes the $\mathbf{A}, \mathbf{U}$ matrices are computed using symbolic algebra in "F2A_symbolic.m" and "F2U_symbolic.m" and then the $\mathbf{F}, \mathbf{V}$ matrices are computed from $\mathbf{A}, \mathbf{U}$ in "LinearizeDiscretize.m" which have $\mathbf{A}, \mathbf{U}$ ready from the symbolic computations.

Update

The next step in the Kalman filter is to update our prediction based on the observations received from our measurements. By doing the update step, our sense of the system states statistically approaches the actual values. The following equations show the update procedure.

$$\begin{aligned} \boldsymbol{\mu}_t &= \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t\left(\mathbf{z}_t - g\left(\bar{\boldsymbol{\mu}}_t, 0\right)\right) \\ \boldsymbol{\Sigma}_t &= \bar{\boldsymbol{\Sigma}}_t - \mathbf{K}_t \mathbf{C} \bar{\boldsymbol{\Sigma}}_t \\ \mathbf{K}_t &= \bar{\boldsymbol{\Sigma}}_t \mathbf{C}^T \left(\mathbf{C} \bar{\boldsymbol{\Sigma}}_t \mathbf{C}^T + \mathbf{W} \mathbf{R} \mathbf{W}^T\right)^{-1} \end{aligned} \tag{12}$$

where $\mathbf{z}_t$ and $\mathbf{R}$ are the measurement value and corresponding noise covariance matrix. $\mathbf{g}$ is the measurements function and $\mathbf{C}$ is the matrix of states to measurements mapping. In this project $\mathbf{W}$ is the identity matrix. Also according to the project requirements, the value of the noise covariance matrix should be tuned in a way that the estimator converges. We have considered a standard deviation of 2 mm in position, 1° in rotation, and 0.05 m/s in velocity for the Vicon system.

Project Results

Part1

The results of the first part of the project are shown in Fig.1 to Fig.3. As shown in the figures the actual and predicted values of the system states are close to each other which shows the system is observable. It was predictable as in this part we have measurements from both the position and orientation of the drone. Also, it can be inferred that the fluctuation in IMU sensor biases is small and not sharp in time which makes it possible for the filter to use the IMU measurements in an effective way.

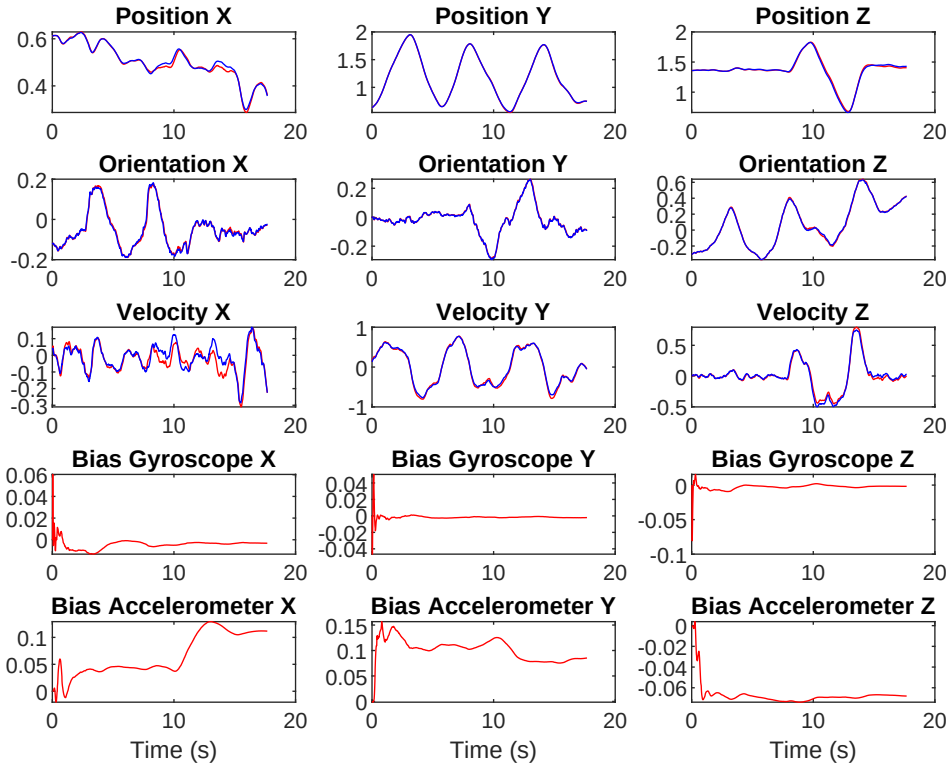

Figure 1: Results of part 1 for Dataset 1. The blue curves show the actual values of states while the red curves show the predicted values. The EKF estimator is clearly converging to the actual values using position and rotation measurements.

Part2

The results of the second part of the project are shown in Fig.4 to Fig.6. In this part, the filter only uses measurements of the velocity in the Vicon system. As seen in the figures, this time the estimate of the position and the velocity of the drone are quite accurate but on the other hand, the estimates of the orientation are not as accurate as the last part. So the observability of the system seems to be weaker in this part. Similar to part 1 in this part also the IMU sensor biases seem to be stable in time and don't have sharp transitions except for very few cases but generally the fluctuations in sensor biases are more than part 1 in this part.
Lastly, it's important to note that generally the accuracy of the estimates highly depends on the noise covariance matrices and more specifically the variance and power of noises on the sensors and measurements. The smaller the noise is, the more accurate the estimates will be which is very intuitive.

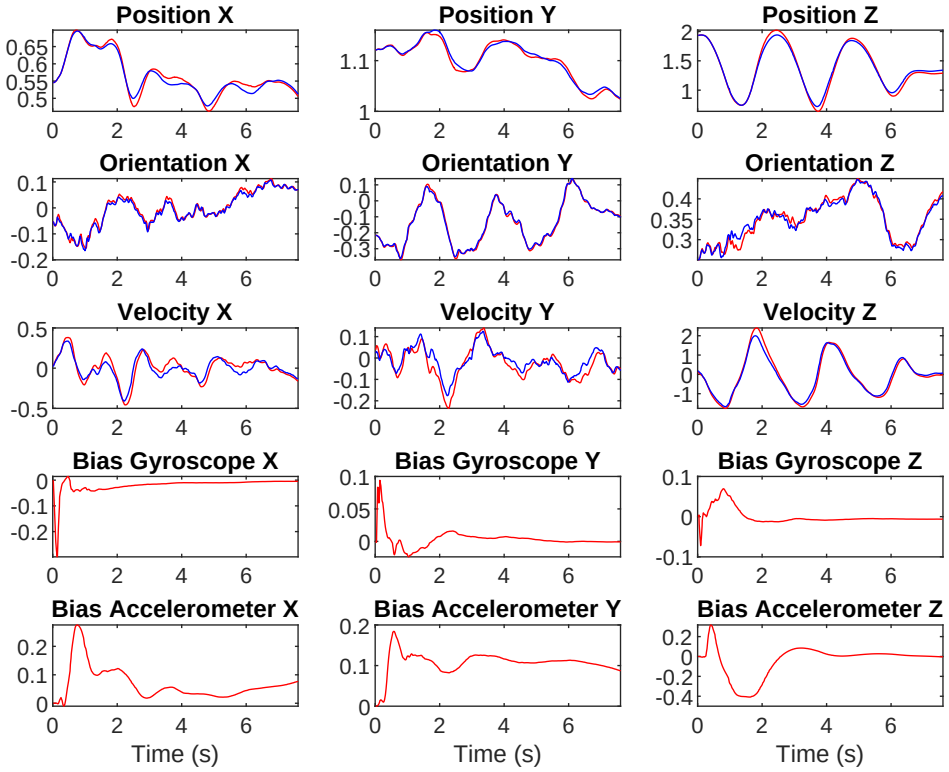

Figure 2: Results of part 1 for Dataset 4. The blue curves show the actual values of states while the red curves show the predicted values. The EKF estimator is clearly converging to the actual values using position and rotation measurements.

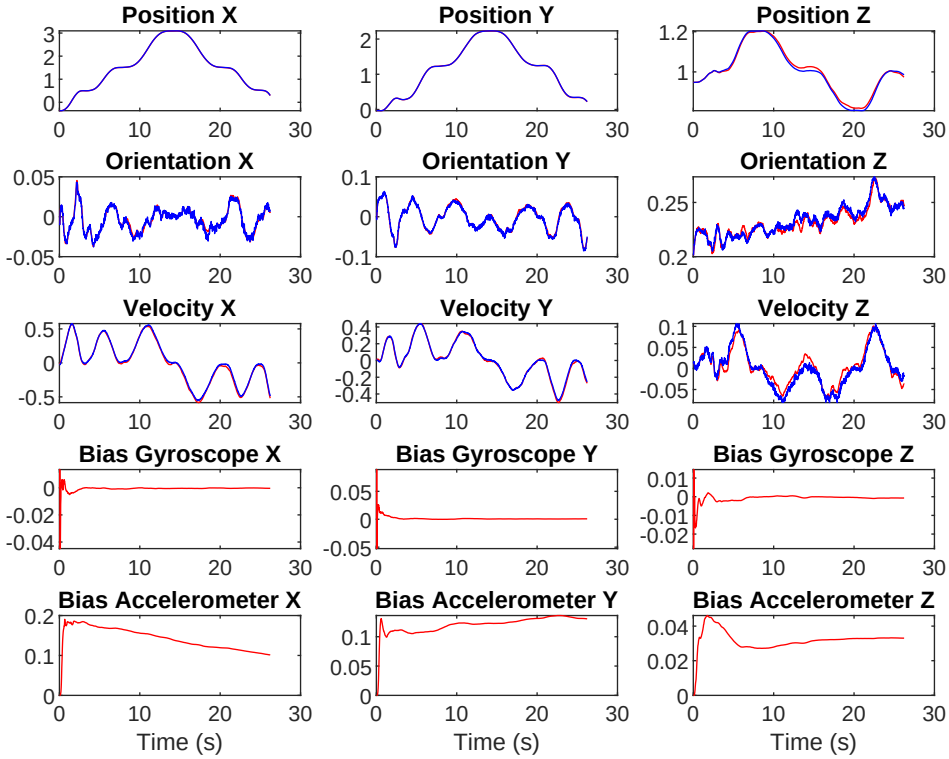

Figure 3: Results of part 1 for Dataset 9. The blue curves show the actual values of states while the red curves show the predicted values. The EKF estimator is clearly converging to the actual values using position and rotation measurements.

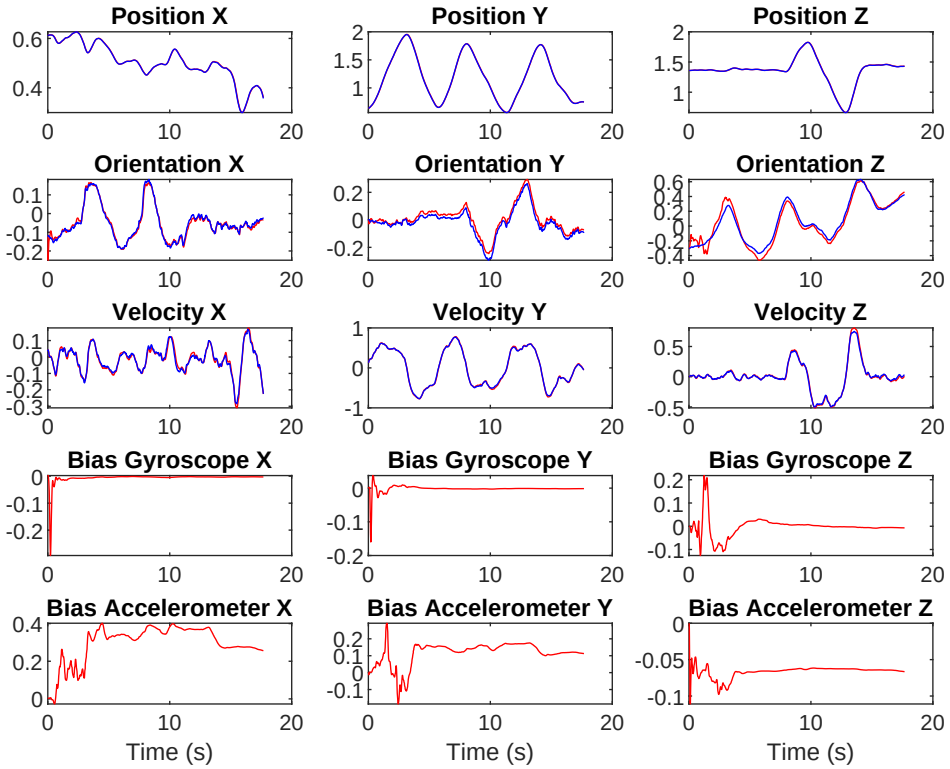

Figure 4: Results of part 2 for Dataset 1. The blue curves show the actual values of states while the red curves show the predicted values. The EKF estimator is clearly converging to the actual values using velocity measurements. The estimate of orientation is somehow less stable compared to part 1.

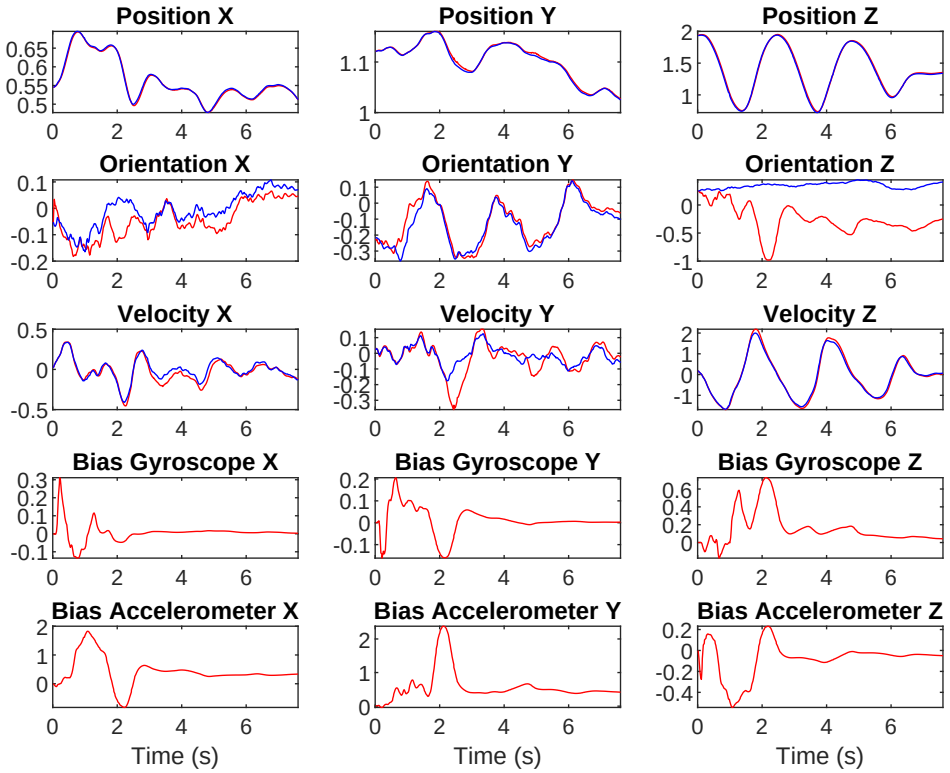

Figure 5: Results of part 2 for Dataset 4. The blue curves show the actual values of states while the red curves show the predicted values. The EKF estimator is clearly converging to the actual values using velocity measurements. The estimate of orientation is somehow less stable compared to part 1.

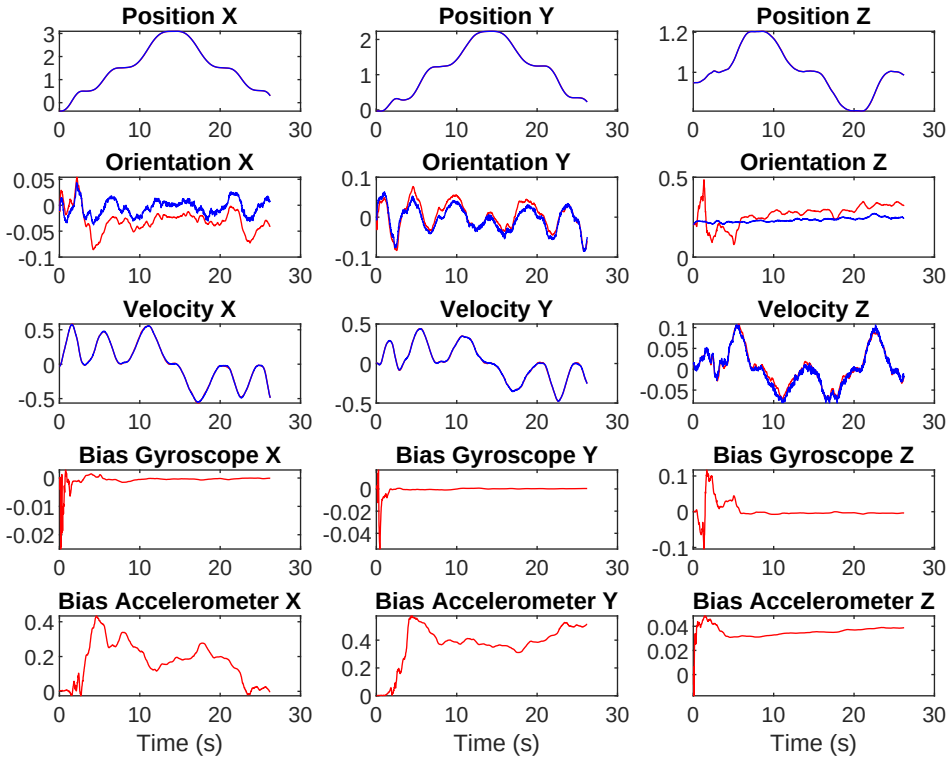

Figure 6: Results of part 2 for Dataset 9. The blue curves show the actual values of states while the red curves show the predicted values. The EKF estimator is clearly converging to the actual values using velocity measurements. The estimate of orientation is somehow less stable compared to part 1.