ME-7933 Fundamentals of Robot Mobility **Project-2 Report**

Name: Harish Kuppam

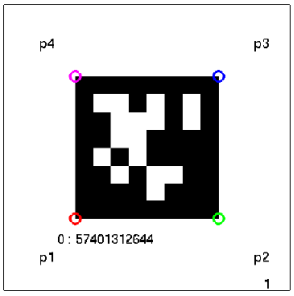
Net ID: vvk243 **NYU ID: N19024713**

Project Introduction:

Here we are working on a Micro Air Vechicle, which has a camera and an IMU sensor mounted on it and the MAV is driven in a known environment where all the required states can be measured using a Vicon sensor. The whole project is composed of two different parts where in the first part we estimate the pose of MAV using a mat of April Tags present on the ground. In the later part we estimate the linear and angular velocity of the MAV. All the work here is done using techniques of Computer Vision.

Pose Estimation (Part-1):

Here we estimate the pose of the UAV at all the times using the image data provided by the camera feed. A mat containing a matrix of 108 April Tags in (9×12) manner is laid on the ground and the MAV is driven such that the on board camera captures a few tags all the time. Now the top left corner of top left tag is considered as $(0, 0)$ point in the world frame. The dimensions of tags are defined as shown in the image below,



Each tag is a $0.152m$ square with $0.152m$ between tags with the exception of the space between columns 3 and 4, and 6 and 7, which is $0.178m$. Using this information we create a function named getCorner, which takes the ID of the tag as input and returns the co-ordinates of all the points each tag in the world-frame as output. The remainder of ID when divided by 12 is taken as x and the integer part of ID when divided by 12 is taken as y. Now going right in the mat the y-coordinate of top left corner is calculated and by going down the x-coordinate is calculated by going down in the mat by making appropriate calculations. Since we know relationship between each corner we can measure the coordinates of all the points in the tag.

The next step is to use the known equation of projective transformation to calculate the transformation from the image frame to world frame. The equation is,

$$\lambda_i \begin{bmatrix} x_i^{'} \\ y_i^{'} \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} * \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{1}$$

The above equation is then converted to a following more suitable form,

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i^{'}x_i & -x_i^{'}y_i & -x_i^{'} \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i^{'}x_i & -y_i^{'}y_i & -y_i^{'} \end{bmatrix}_{2\times 9} h_{9\times 1} = 0 \implies Ah = 0 \tag{2}$$

h is a (9×1) matrix but since all the calculations are made to scale it has only 8 degrees of freedom. We can see that each point contributes 2 rows so we need at-least four points to calculate h. In our case at each instant we have multiple tags detected. So we stack up all the rows obtained from all the points from all the detected tags and form an A matrix. Now, we perform SVD and estimate h in the following way,

$$A = USV^T \implies h = V_9$$

Since we know that all the tags are on the ground, we can consider $Z_w = 0$ and obtain the following equation.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} \implies R = \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} = K^{-1}h_{3\times 3} \tag{3}$$

Given,

$$K = \begin{bmatrix} 311.0520 & 0 & 201.8724 \\ 0 & 311.3885 & 113.6210 \\ 0 & 0 & 1 \end{bmatrix}$$

Now using the above obtained R matrix we calculate ${}^{C}R_W$ and ${}^{C}T_W$ in the following way,

$$\begin{bmatrix} R_1 & R_2 & R_1 \times R_2 \end{bmatrix} = USV^T$$

$${}^{C}R_W = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(UV^T) \end{bmatrix} V^T \text{ and } {}^{C}T_W = \frac{R_3}{\|R_1\|}$$

The pose of the MAV in the world frame is equivalent to ${}^{W}R_B$ and ${}^{W}T_B$ which can be obtained by, ${}^{W}H_B = {}^{W}H_C \times {}^{C}H_B$, ${}^{W}H_C$ can be found by using ${}^{C}R_W$ and ${}^{C}T_W$ obtained above. From, given Camera-Body calibration, ${}^{C}H_B$ can be found as,

$${}^{C}H_B = \begin{bmatrix} 0.707 & -0.707 & 0 & -0.04 \\ -0.707 & -0.707 & 0 & 0 \\ 0 & 0 & -1 & -0.03 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

After obtaining ${}^{W}R_B$ and ${}^{W}T_B$, the required result is obtained by transforming ${}^{W}R_B$ to euler angle format in XYZ manner.

Results:

The above steps for pose estimation are performed for each time step and the pose data is plotted. This data is compared with vicon data to check the validity.

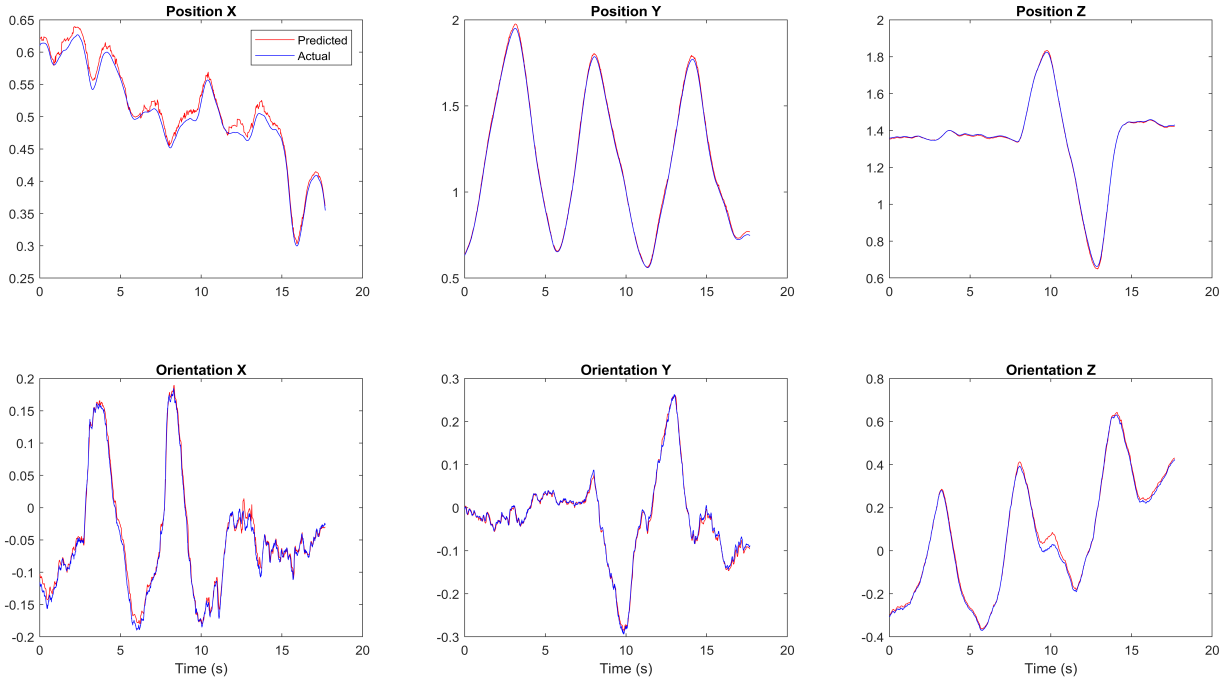


Figure 1: **Project Part-1, DataNum: 1**

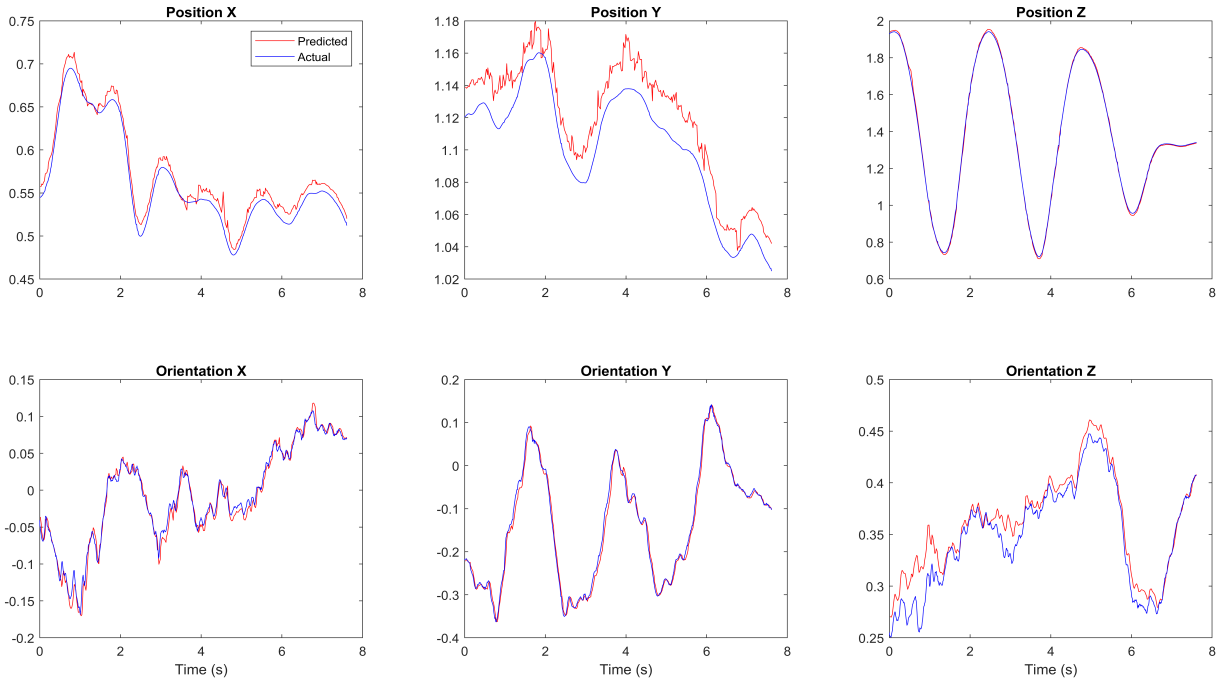


Figure 2: **Project Part-1, DataNum: 4**

Velocity Estimation (Part-2):

In this part we calculate the velocity of the MAV using the video feed of the on-board camera. This is achieved by first calculating the optical flow using KLT image tracking technique. Later appropriate calculations are made to calculate linear and angular velocity of the MAV w.r.t world in the world frame. Firstly, we detect good points in a frame using corner detector functions in Matlab. Later we choose strongest 50 of those points to reduce the iterations. In the next frame, point tracker techniques are used to track the position of previously detected points in the current frame. The difference of these two sets of points from two frames is calculated and divided by δt to calculate the optical flow,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} / \delta t$$

Now that we have the optical flow for every point of the image, we use the following equation to estimate the required linear and angular velocities of camera.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{4}$$

Here (x, y) are point coordinates in the image frame. Z is the depth of that corresponding point in the camera frame and (u, v) is the optical flow calculated above. The only unknown in the above equation in Z which can be calculated by using results obtained from part1. We know, λ_i in the equation (1) is the scaling factor which is equivalent to Z_c i.e, the depth required,

$$\implies Z_c = \lambda_i$$

We know $H = \begin{bmatrix} R_1 & R_2 & T \end{bmatrix}$ where $R = {}^{C}R_W$ and $T = {}^{C}T_W$. These are obtained by transforming the pose obtained from the part-1.

$$R = R^T_{c2w} \text{ and } T = {}^{C}T_B - R^T_{c2w} \times {}^{w}T_B$$

Now restructuring the above equation (1),

$$A = H^{-1} \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} x_i/Z \\ y_i/Z \\ 1/Z \end{bmatrix}$$

$$\implies Z = 1/A(3)$$

Now that we have calculated all the required variables in (4), we can calculate the 6×1 V vector which can be calculated in two different ways.

Equation (4) can be represented as,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_1(x, y, Z) \\ f_2(x, y, Z) \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

We obtain 50 sets of U and F vectors for 50 detected points. They are stacked up below each other to from two 100×1 vectors are both sides of equation. Now using the least-square principle we can estimate the most optimal velocity vector.

$$\implies \begin{bmatrix} V \\ \omega \end{bmatrix} = lsq(F_{100\times1}, \begin{bmatrix} u \\ v \end{bmatrix}_{100\times1})$$

Least square solution is great but it will not give the best solution because it may include a few outliers. In order to get rid of these outliers we use RANSAC technique. In can be observed from (4) that atleast three points are required to find the velocity. So we perform three point based ransac to obtain inliers. We know number of iterations k is,

$$k = \frac{log(1 - \rho_{success})}{log(1 - e^M)}$$

In our case we know $M = 3$. Considering the trends of data $\rho_{success}$ and e are taken as 0.95 and 0.5 respectively.

$$\implies k = 22$$

Now we run k iterations where in each iteration we select random three points and velocity vector is calculated. Then this velocity is used to calculate $\begin{bmatrix} u \\ v \end{bmatrix}$ for every other point and the norm of the difference between this and the optical flow obtained before is measured to differentiate outliers. cost

$$= \left\| F_i \begin{bmatrix} V \\ \omega \end{bmatrix} - \dot{p_i} \right\|^2$$

If cost$<$ 0.001, then that point is considered as inlier. Later we consider the iteration with most number of inliers. All the F matrices of these inliers are stacked up and the corresponding optical flow values are stacked up in a vector U. The velocity is then calculated using the least squares method as above.

$$\implies \begin{bmatrix} V \\ \omega \end{bmatrix} = lsq(F_{2*inls\times1}, U_{2*inls\times1})$$

This will give more optimal result compared to a simple least square method because we have rejected all the outliers.

The above obtained velocities are ${}^{C}\dot{P}_{W}^{C}$ and ${}^{C}\omega_{W}^{C}$. In order to match them with the data given by vicon we need to transform the above veclocities to ${}^{W}\dot{P}_{W}^{B}$ and ${}^{B}\omega_{W}^{B}$. This transformation is done using,

$$\begin{bmatrix} {}^{B}\dot{P}_{W}^{B} \\ {}^{B}\omega_{W}^{B} \end{bmatrix} = \begin{bmatrix} R_{B}^{C} & -R_{B}^{C}S(r_{CB}^{C}) \\ 0 & R_{B}^{C} \end{bmatrix}, {}^{W}\dot{P}_{W}^{B} = R_{B}^{W}{}^{B}\dot{P}_{W}^{B}$$

Results:

The following plots are of results obtained before applying filter,

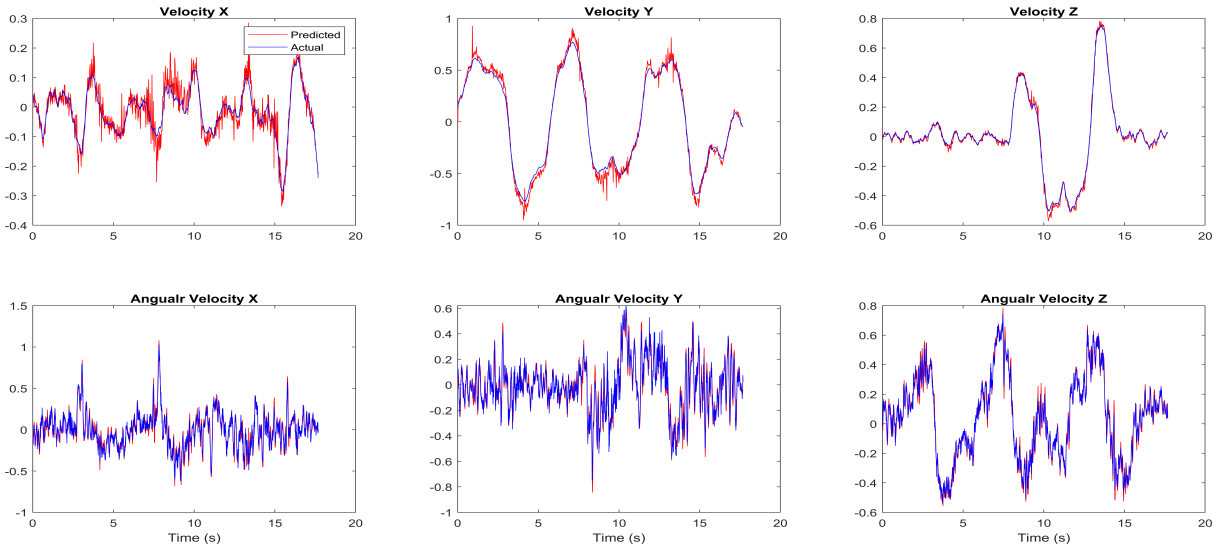


Figure 3: **Project Part-2 before filter, DataNum: 1**

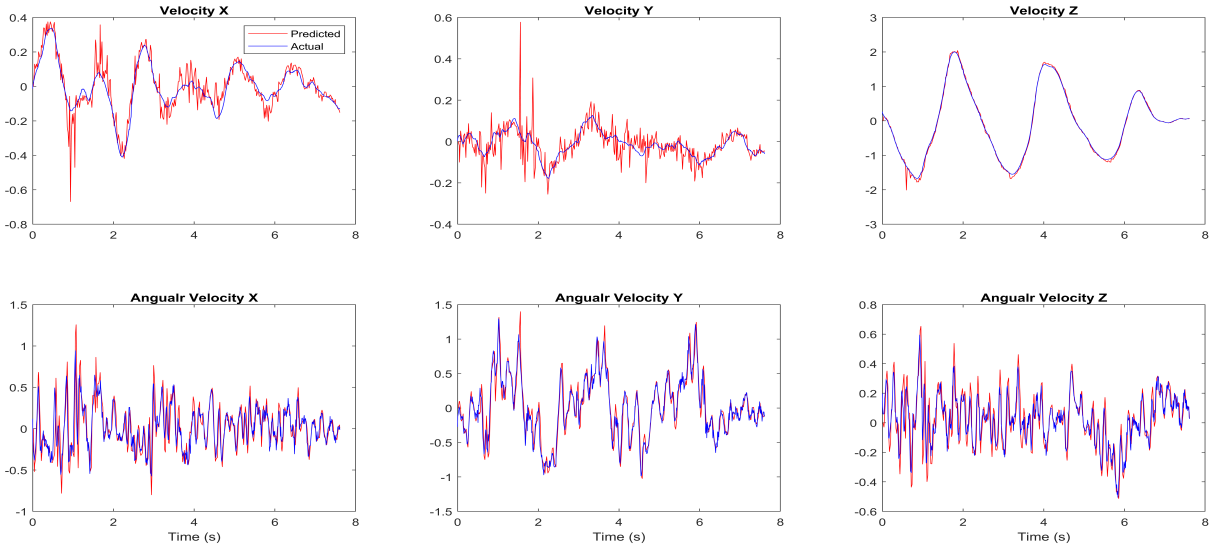


Figure 4: **Project Part-2 before filter, DataNum: 4**

It has been observed that the results obtained contain a lot of noise so, a lowpass filter is applied to make the signals smoother. In our case a moving average filter of degree 4 is applied.

The following plots are of results obtained after applying filter,

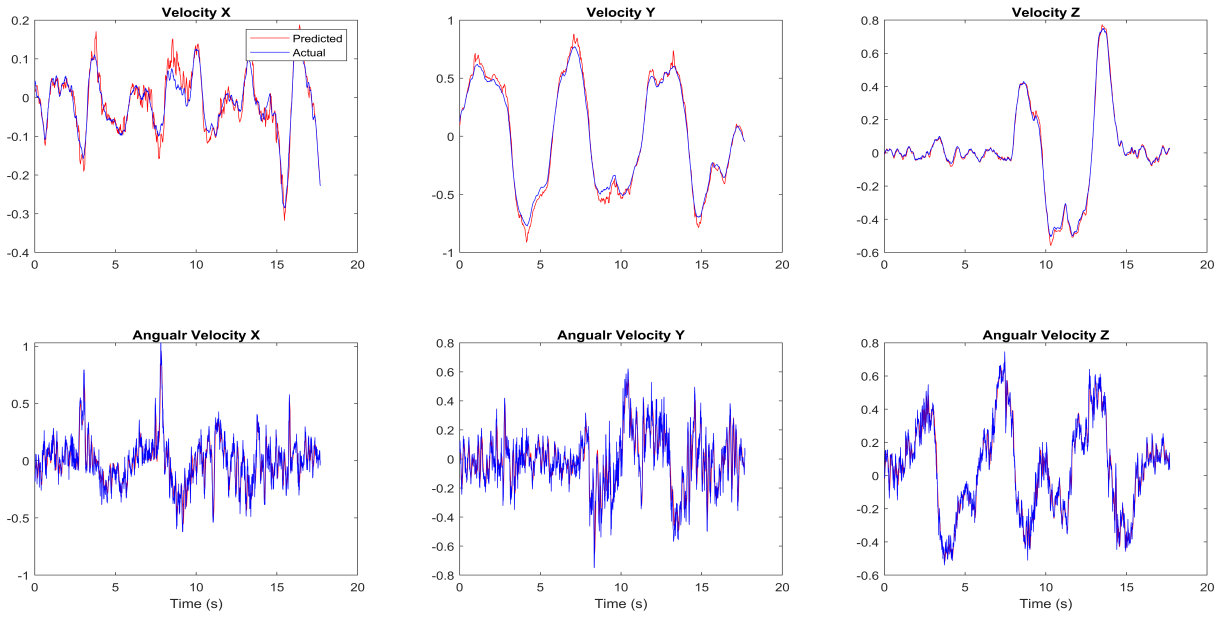


Figure 5: **Project Part-2 after applying filter, DataNum: 1**

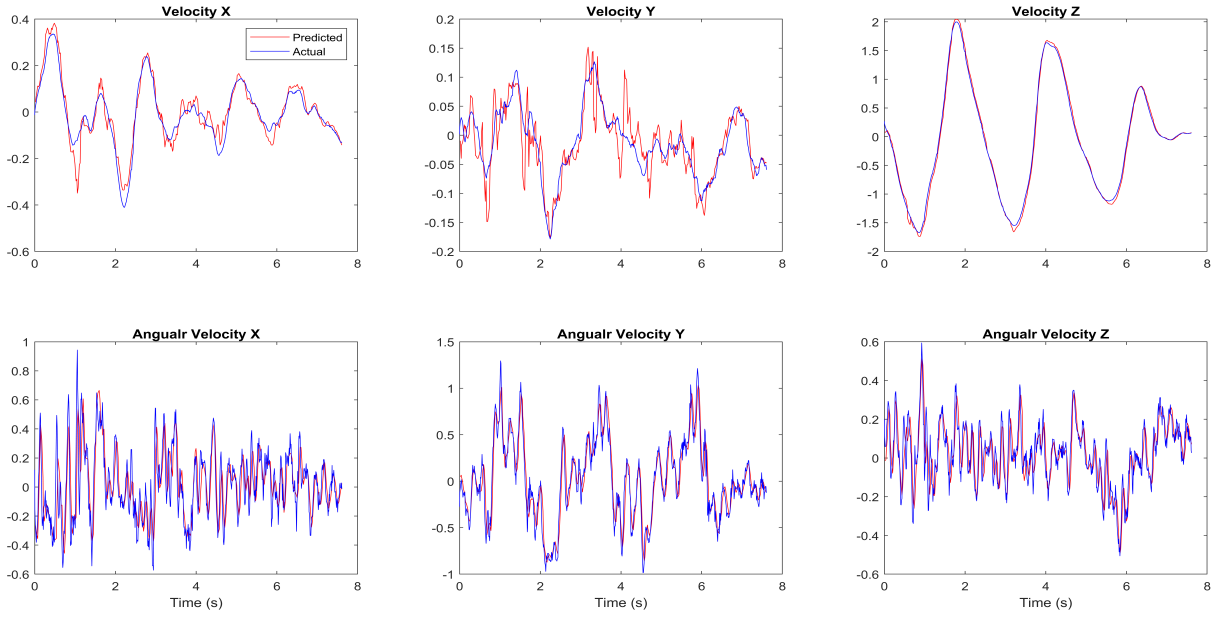


Figure 6: **Project Part-2 after applying filter, DataNum: 4**

Conclusion:

From the results in part-1 it can be observed that position and orientation aligns almost perfectly with the vicon data stating that this method of finding out pose is reliable and later when we use this data as sensor measurement in kalman filter, we can keep the noise very low. The results obtained in part-2 are more noisy and sometimes goes out of bounds. Though filtering helps to some extent it doesn't completely eliminate the problem. So we can use this data as measurement with a little high noise. One more thing to note is that the computation time to calculate pose and velocity differs by huge factor and we'll have to accommodate this by setting the proper frequencies while capturing data.