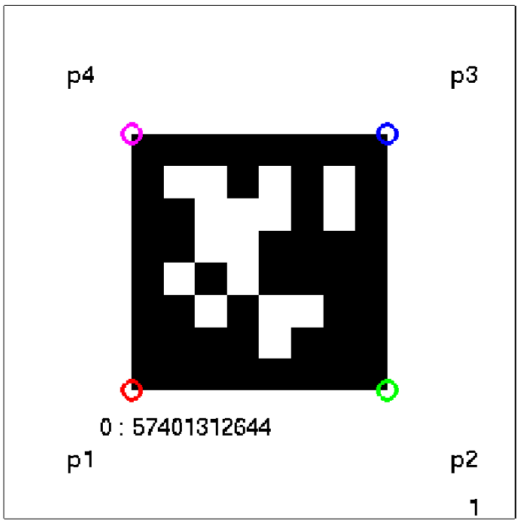
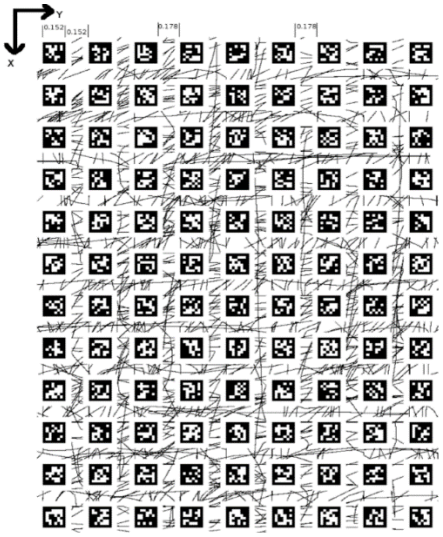
Part 1:

The following are the coordinates and the image description which were presented for the Project. Using this and the other necessary datas provided, this part was completed.





The aim of this project is to get the estimate of the Position and orientation of the detected April tags, given the data regarding the position of every April Tags and their respective IDs at given time stamp. For this, we need to find out the H matrix first which gives an estimate of points in the world frame as seen from the camera frame. Then find out the position and orientation according to the single value decomposition of the derived H matrix. Afterwards the position and orientation is transformed to the world frame. The structure of my code is as follows.

$$\text{Transformation: } \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim K(R_w^c \;\; T_w^c) \begin{pmatrix} P_w \\ 1 \end{pmatrix}$$

$$\text{Camera Coordinates} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_o \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} r_{12} r_{13} t_1 \\ r_{21} r_{22} r_{23} t_2 \\ r_{31} r_{32} r_{33} t_3 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

$$\text{Relation between camera and world coordinates: } \lambda_c p_c = H \lambda_w p_w \;\; (p_c \sim H p_w)$$

$$Ah = \begin{pmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i' x_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i' y_i & -y_i' \end{pmatrix} h = 0$$

$$\text{Singular Value Decomposition: } A = USV^T = h_9$$

Extracting rotation and translation from the given matrix:

$$(\widehat{R_1} \quad \widehat{R_2} \quad \widehat{T}) = \begin{pmatrix} \widehat{r_{11}} & \widehat{r_{12}} & \widehat{t_1} \\ \widehat{r_{21}} & \widehat{r_{22}} & \widehat{t_2} \\ \widehat{r_{31}} & \widehat{r_{32}} & \widehat{t_3} \end{pmatrix} = \begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{32} & h_{32} & h_{33} \end{pmatrix} = K^{-1} H$$

Taking the singular value decompositon again: $(\widehat{R}_1 \quad \widehat{R}_2 \quad \widehat{R}_1 \quad X \quad \widehat{R}_2) = USV^T$

$$\text{Final Rotation Matrix:} \quad R = U\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & \det(UV^T)\end{pmatrix}V^T$$

$$\text{Final Translation Vector:} \quad T = (\hat{T})/||\hat{R}_1||$$

To find out the corners of the April Tags, the top left corner is considered the origin. Then with the provided sizes of the lengths, breadths and the distance between each Tag, the four corners of each Tag's coordinates are calculated. The X and Y coordinates thus calculated are stored in a 12×9×9 matrix. The first matrix is considered the ID, and the ones after are respectively four X coordinates relative to each ID. Then after the following four matrices are, Y coordinates relative to each ID. Whenever the ID is called, all those coordinates are transferred to the 'estimatePose' function.

From the data, we consider the X and Y coordinates of these 4 points p1, p2, p3 and p4. Once we get all these points, we calculate the A matrix. From the A matrix, singular value decomposition is performed. Thus, we get a 9×9 Matrix. From there the final column is extracted and reshaped. To get the properties of matrix right, the sign of the last element of the matrix is extracted and multiplied by the H matrix. To normalize the matrix in the camera frame, the matrix is normalized by the inverse of K matrix provided. Therefore, we need to satisfy some valid constraints. Then after, we again do a single value decomposition of the matrix generated to generate a transformation matrix from the world frame to the camera frame. According to the given parameters, the transformation matrix from the camera frame to the IMU frame is also calculated. Multiplying them gives the proper transformation of every point from the world frame to the IMU frame. This information is used to derive the position and orientation.

The data is compiled up in a format of 'data'. This gives us a series of ID elements and time stamps. From these, the A matrix is of dimension 2×9. The computed matrix is of 3×3 matrix. Then the position sent over to the primary matrix is 3×1. The orientation sent over is 3×1 as well.

The only variable, which can affect the differences, is the argument part in this problem. Since it is just a general solution and not the actual one. Due to presence of noise, the actual solution is hard to reach. Still with repeated iterations and exact values, it might give us exact results.

Analysis for Model Pose Estimation – Student Data 1:

For Dataset 1, mostly the results are convergent. The Position Z and Orientation Z seems to be a good match. Along with that Position and Orientation of Y also seems okay. However, there seems to be some distortions in the Position X due to some noise obtained from the given data. Most probably, the arguments from the single value decomposition was not enough. Thus, we can see the deviations in the curve. Still the Position and Orientation of X seem convergent. Therefore, overall this graph seems to be a good plot. Still the plot can be improved by continuing in a different way other than using SVD if possible or by recording a

better motion with fewer noises in it. Applying a low pass filter could have helped it get back to shape. It almost coincides with Vicon Data.

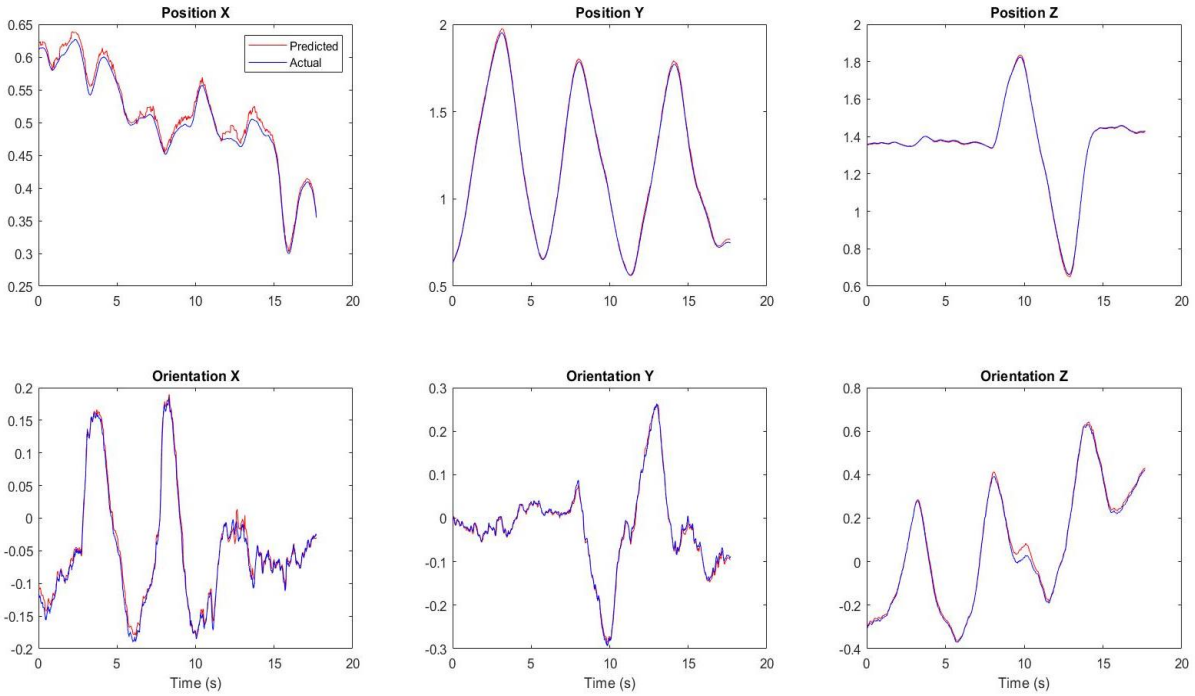


Fig. 1: Pose Estimation – Student Data 1

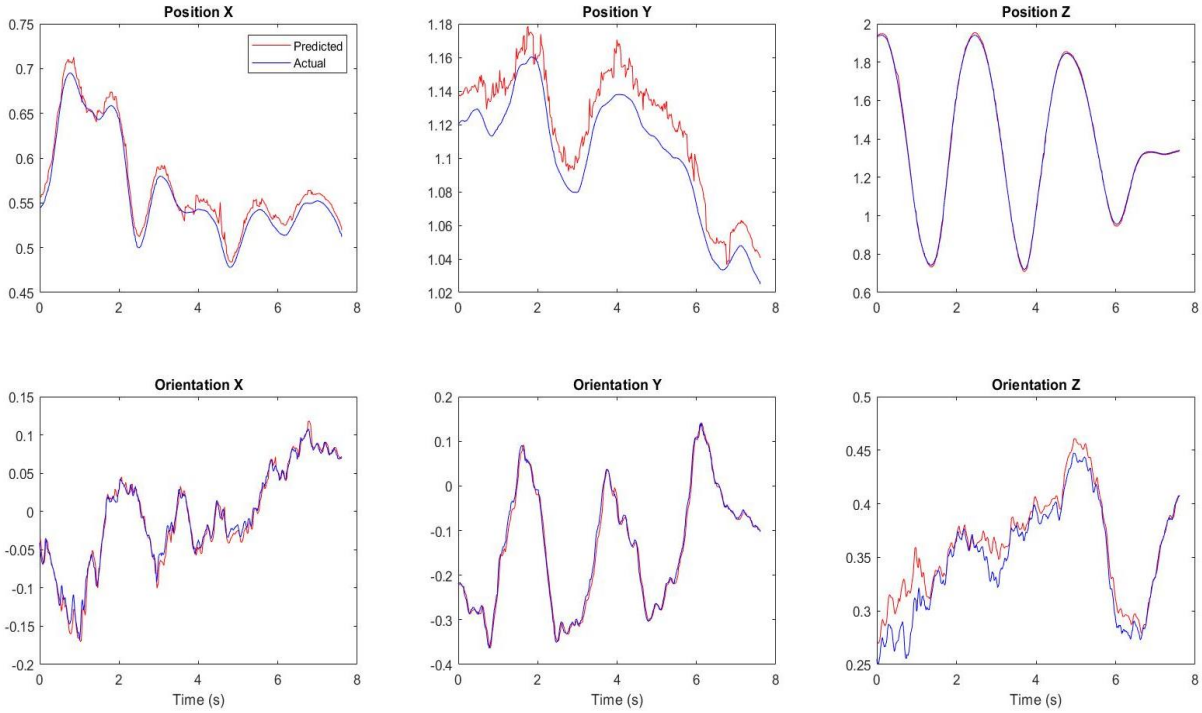


Fig. 2: Pose Estimation – Student Data 4

Analysis for Model Pose Estimation – Student Data 4:

For Dataset 2, the orientation seems to be a bit more off the hook. Thus, it brings with itself the deviations in the plots of Orientation of Z. This part almost coincides with the Vicon Data. Though the position seem to be fine, the orientation seems to deviate a little as we can see, due to noise. Then again, The Position of Y seems pretty off the base, whereas the orientation seems fine. Again, like the previous plot, we see some deviations in the Position of X. A better way to calculate the SVD or by lowering the noise using a low pass filter would give better results. Even applying threshold limits seem helpful.

Part 2:

The aim of this part of the project is to find out how the Velocities between the rates from change of frames as the camera is moved around. The basic concept behind this is based on object detection and tracking. From there we detect the velocities of the detected points and then track them across frames. For this, we need the objects' orientations and positions. We get that from the results of previous Part 1. We get the points, as in the April Tag's positions and orientations with respect to the IMU Frame. Using that data, we calculate the height of the object from the camera. Thus via least square method, we calculate the Linear Velocity and angular velocity. For this part of the project, the problem was which are the 'good points' we should select instead of choosing all the points. The explanation of good points being the ones, which exactly had the true values in the tracked image. This is a tedious computation, thus was computed via RANSAC. Once the velocities were calculated, a threshold velocity bound them. Afterwards, the velocities had to be converted in their proper frames. Finally, a low pass filter was applied. The structure of my code is as follows.

Linear and angular velocity: $$\dot{p} = \frac{1}{Z}A(p)V + B(p)\Omega$$

Motion Field Equation:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{Z}\begin{pmatrix} -1 & 0 & x \\ 0 & -1 & y \end{pmatrix}\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} + \begin{pmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \end{pmatrix}\begin{pmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix}$$

Argument to be satisfied for the velocities to be unique:

$$V^*, \Omega^* = \arg\min_{V,\Omega} \sum_{i=1}^{n} \left\| \left(\frac{1}{Z_i} A(p_i)\ B(p_i) \right) \begin{pmatrix} V \\ \Omega \end{pmatrix} - \dot{p}_i \right\|^2$$

Determining the probability of incorporating all the inliers:

$$k = \frac{\log(1 - p_{success})}{\log(1 - \varepsilon^M)}$$

Firstly, the Camera matrix provided to us, an average time difference between transitions from every frame to the next one a skew symmetric matrix and a transformation matrix of the translation between the IMU and the camera were initialized. Firstly, the loops were initialized with the current and the upcoming frame. Then, two features were continuously

used, to detect the strongest points to be tracked in the upcoming frame. Then a reverse tracking mechanism was used while initializing those points in the current frame and then tracking them in the upcoming one. Then after, the points from both the frames were normalized using the Camera matrix, and their differences were taken to find the change in displacement of those points. Then after those points were divided by the average time difference for the transition between two frames for the ease of computation. From this process, we get the Optimal Velocity, a 50×2 matrix required to compute RANSAC. The normalized new points were taken up as the Optimal Position, a 50×2 matrix.

From the given data, we computed Part 1 just as explained before and we got the position and orientation of the each object in the world frame with respect to the body frame. Using that data and the matrix required to change the objects' orientations from the body frame to the camera frame and from the world frame to the camera frame we can calculate the normal height an object is from the camera at any instant of time. Once the height was received, the next step was to carry out a probability search of our acquired data to be exact, and that was considered to be 0.4. Thus, that explains the fact of carrying all the inliers with them. The minimum number of points for the RANSAC model was 3. Thus the probability of constricting all the inliers were 0.4^3. The probability of hitting at least one inlier set was raised to 0.99. Thus, the probability of hitting the inliers were calculated based on made assumptions. Then after, the following equation was used to iterate the values for the given number of iterations. Each of these matrices were 2×6 considering the A and B matrices together for n number of values.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{Z}\begin{pmatrix} -1 & 0 & x \\ 0 & -1 & y \end{pmatrix}\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} + \begin{pmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \end{pmatrix}\begin{pmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix}$$

Any three random index numbers were selected and a matrix was made out of them as suggested by the above equation. Then it was subtracted from the corresponding matrix generated before to find out the error for each of those indexes of those Velocities. Thereafter, a loop was run to calculate the square root of the summation of velocities related to the X and Y coordinates of each of the velocities. Then if they were below a certain range, they were considered among the inliers. The total number of inliers being the most in one of those iterations were considered the one responsible to derive the final most Optimal velocity. Thus, the work of RANSAC was over with this determination. The best velocity points was derived and selected as a 6×1 matrix.

Now it had to be bound by an upper and lower threshold velocity. These would constrict the plots within a certain range. Considering the Vicon data, these were set to be $(-2,2)$. Thus, beyond the range the Velocities could not rang out. Up next, the Velocities were to be modified and changed in the proper orderly manner. The Linear velocity had to be changed to the camera frame and the angular velocity had to be shifted as well. Using the rotation matrices from the camera to the world frame, this was performed. Once we have fixed the Velocities, some low pass filters had to be applied to smoothen the curve. So for that purpose, for every few points as mentioned in the file, the previously estimated velocities

were added and then an average was taken. The number of points selected were 2. Finally, the output 6×1 matrix was the solution and it was plotted.

The only variables, which could affect this, are the proportion of inliers and the probability of hitting an inlier and the detailed calculating power of MATLAB. In addition, variables like average time were used for the inconsistency due to data noise, selecting only 50 strongest points when the feature was applied. In addition, the whole tracking comes out of a black box in Computer Vision Toolbox. Tracking it perfectly is a very tough duty, even for RANSAC. In addition, the low pass filters can have variations and instead affect our result according to the kind of points it detects. Finally the same low pass filter was again applied.

Analysis for Model Optical Flow – Student Data 1:

According the shown plots, it is visible that the Position and Orientation of Z is well defined. It almost coincides with the data from the Vicon. Again, we see that the Velocity of Z does not coincide with the Vicon Data. Probably due to much smoothening of the curve it might not have reached the peaks it should have reached. Though all the velocities are well in the range $(-2,2)$, the threshold limits applied. Therefore, this indicates less amount of error in the process module. Something similar takes place in case of Velocity of X. The distortions in the Positions of X and Y are a contributing factor to this issue. The noise from there, affects the velocities as well. Most importantly, more iterations on the error threshold in RANSAC, the proportion of inliers and the probability of hitting an inlier should help the Velocities. Amongst other uncertainties, the black box detecting the best points are also dependable on the algorithm behind selecting a point. Though the Angular velocities of X and Y seem to follow the trends and mostly do fine.

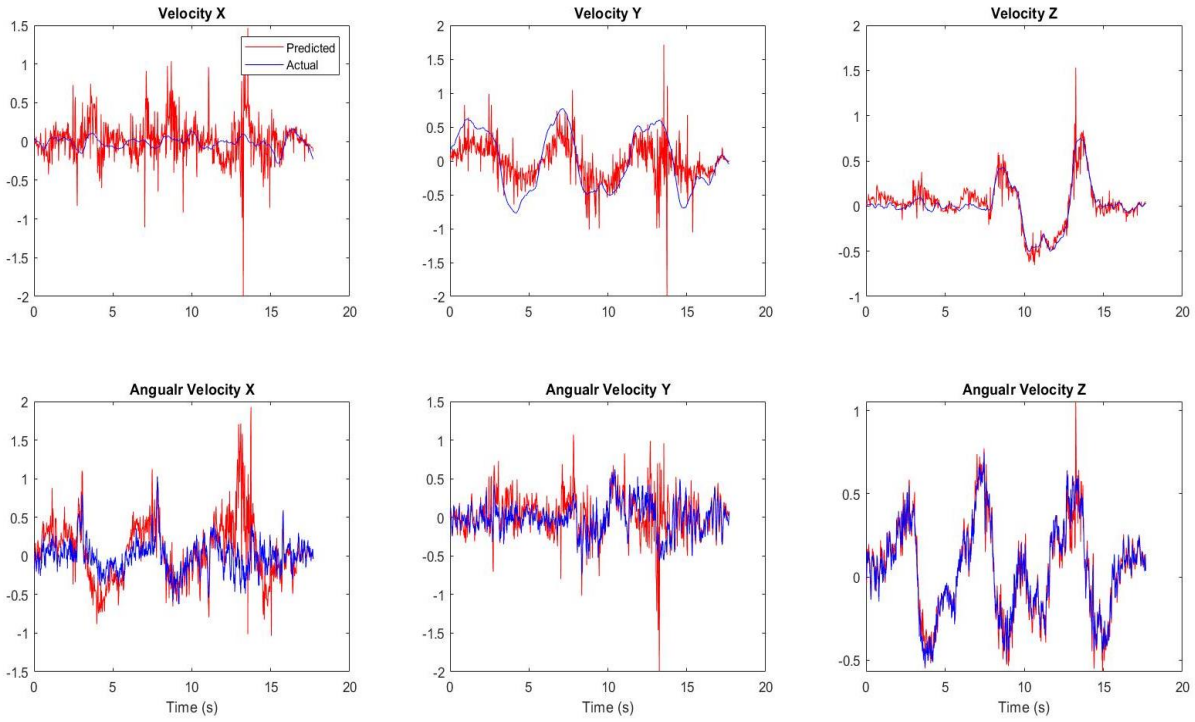


Fig.3: Optical Flow – Student Data 1

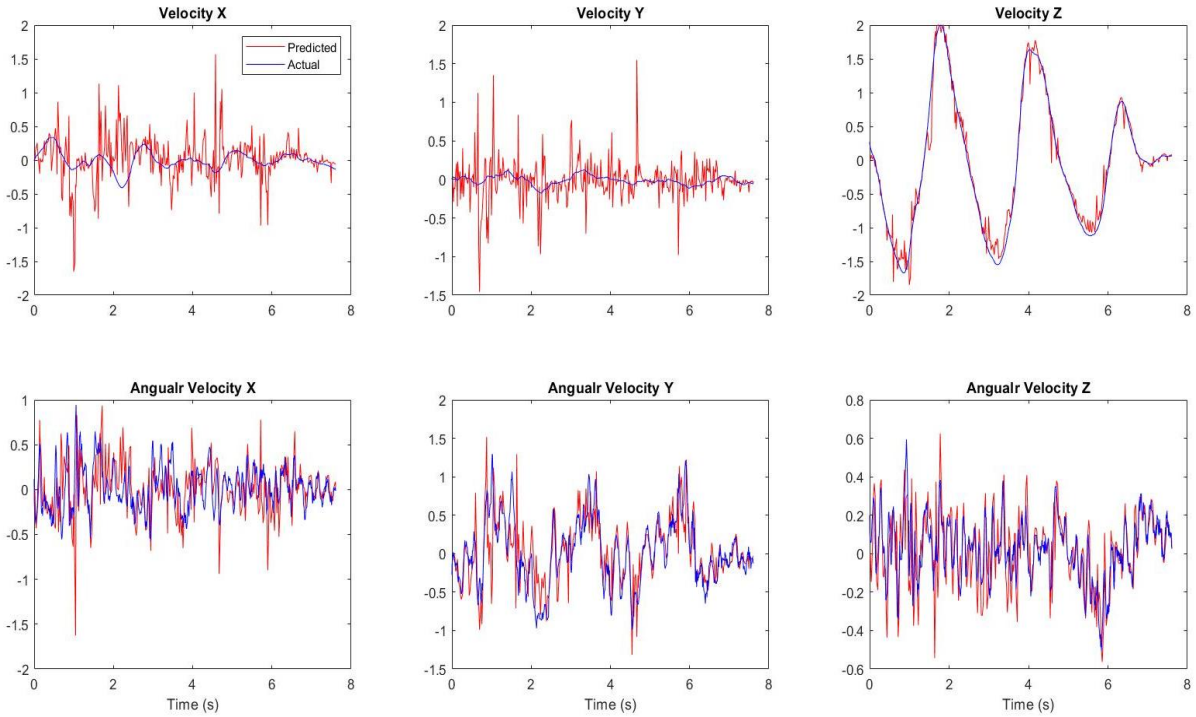


Fig.4: Optical Flow – Student Data 4

Analysis for Model Optical Flow – Student Data 4:

According the shown plots, it is visible that the Position and Orientation of Z is somewhat defined. It pretty much follows the trend of the Vicon Data and almost coincides with it as well. Again, we see that the Velocity of Y does not coincide with the Vicon Data. Working on the low pass filters must be reduced to get better values of this. Although all the velocities are well in range $(-2,2)$, the threshold limits applied. Therefore, this indicates less amount of error in the process module. Something similar takes place in case of Velocity of X. The distortions in the Positions of X and Y are a contributing factor to this issue. The angular velocities of X and Y are still better and I range. The noise from there, affects the velocities as well. Most importantly, more iterations on the error threshold in RANSAC, the proportion of inliers and the probability of hitting an inlier should help the Velocities. Amongst other uncertainties, the black box detecting the best points are also dependable on the algorithm behind selecting a point. Either ways, better iterations with wider number of variable can be expected fruitful.