There are 3 parts to the project 2

1. Vision-Based Pose Estimation
2. Vision-Based Velocity Estimation using corner extraction and tracking
3. Reject outliers using RANSAC

1. Vision-Based Pose Estimation

In order to do that, we have used April Tags on the chart. We have been provided with the number of April Tag and their location mentioned in the handout. We have to write a function "getCorner.m" in such a way that if we provide an ID to "getCorner.m" it gives the center of the ID and all 4 corners of the ID.

Now, we have to make the "A" matrix mentioned in slide 63 of Lecture 8. For the first ID, A matrix would be 10 by 9, the first two-row for the center, the third and fourth row for the bottom left, the fifth and sixth for the bottom right, the seventh and eighth for the top right, and ninth and tenth for the top left. Now, we have to stack the A matrices row-wise for each ID per frame. Now, we have to perform SVD(Singular Value Decomposition) of the A matrix and we have to use the 9^{th} column of the V matrix which gives us the "h" matrix. Which is nothing but the multiplication of the Camera Calibration Matrix and the Matrix which has the first two columns of the rotation matrix between the camera and world frame and the translational vector between the camera frame and world frame.

So, in order to get the first two columns of the rotation matrix and the translation vector, we have to multiply the inverse of the Camera Calibration matrix with the "h" matrix. Now, we have to make a matrix first column R1, the second column R2, and the third column is the cross product of R1 and R2 mentioned in slide 78 of Lecture 8. We have to perform SVD on the given matrix and calculate the Rotation matrix and translational vector mentioned in the formula on slide 78.

This is how we can calculate the Pose of the Quadrotor having the camera.

2. Vision-Based Velocity Estimation using corner extraction and tracking

In order to this task, We have two images one is the previous image, and the other is the current image. We have to track the features in order to calculate the Pixel Velocity. Corners are good features to be tracked.

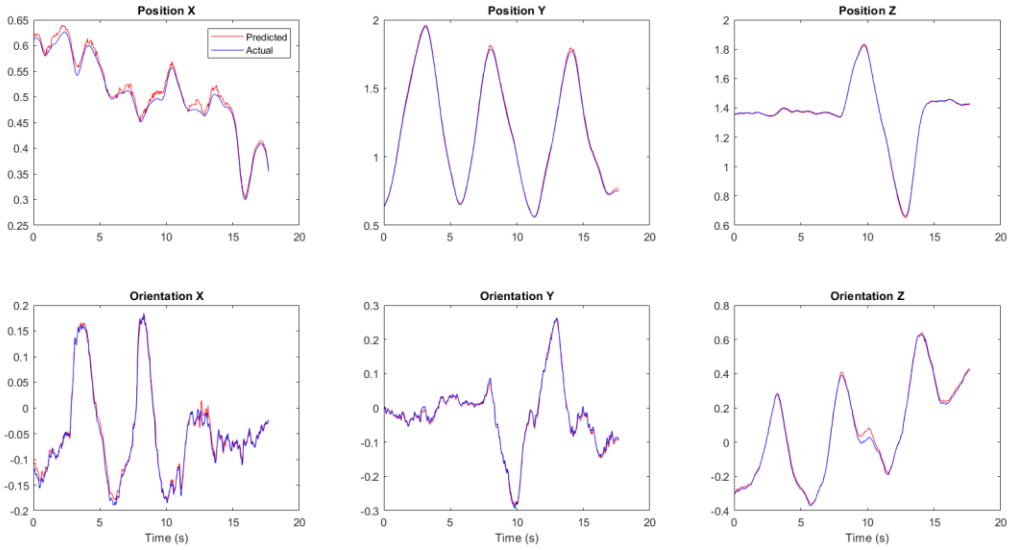
- First, we have to detect the corners in the previous image.
- We have to choose some good points from the corners, that can be choose by Variable.selectStrongest(Number of points).
- After that, we have to use vision.PointTracker to track the good corner points.
- It will help us to find the next location of the points. Now, we have to find the difference between the positions of the corners in the current frame and the previous frame. We can delete those points which are not tracked. After that, we can calculate the pixel velocity in the camera frame by dividing the difference between the positions of the corners in the current frame and the previous frame by a difference in time.

- Now, we have to calculate the depth of the corner or the middle pixel or height of the camera in the camera frame represented by Z that can be calculated from the multiplication of the Camera Calibration Matrix and Transformation matrix between the camera and world frame.
- In order to calculate the transformation between camera and world, we have to multiply the transformation between camera and body frame and the transformation between body and world frame. In order to calculate the transformation between body and camera, we have provided the angle and translation between body and camera frame in the text file. In order to calculate the transformation between body and world, we have to use eul2rotm function on orientation coming from the estimatePose function and translation from position coming from estimatePose function.
- Now, we have to take the inverse of Transformation from world to body and multiply it with the camera to body transformation.
- We can calculate the Linear and Angular Velocity of the camera from one pixel from the formula mentioned in the slide 51 of Lecture 9 and by stacking the velocity of the camera row-wise coming from each pixel and calculate the final Linear and Angular velocity of the Camera in Camera frame according to the formula mentioned in the slide 57 of Lecture 9.
- In order to calculate the camera velocity in the world frame, we have to multiply both the linear and angular velocities coming from the camera frame to the rotation matrix from camera to world frame.

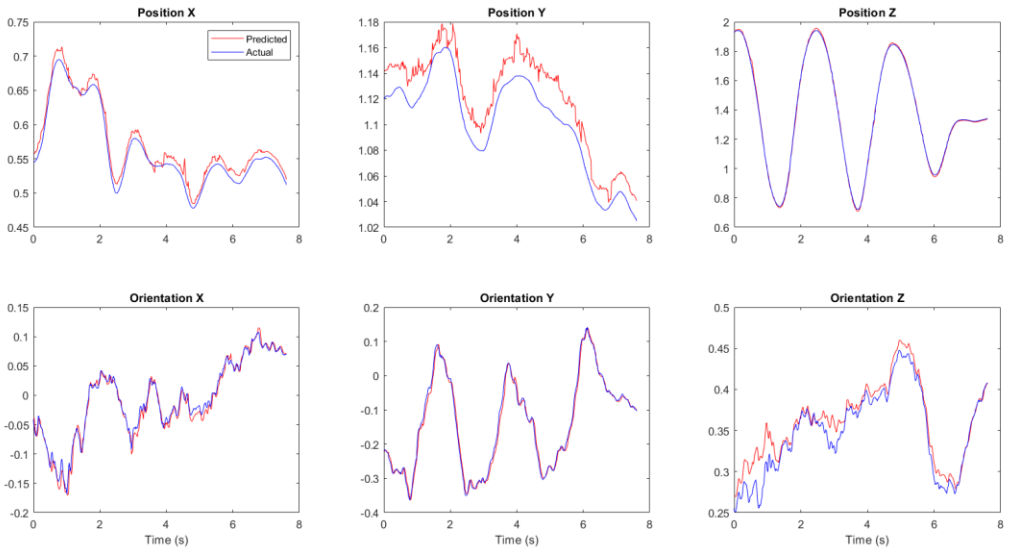
3. Reject outliers using RANSAC

I have created a separate file that is using RANSAC with optical flow.

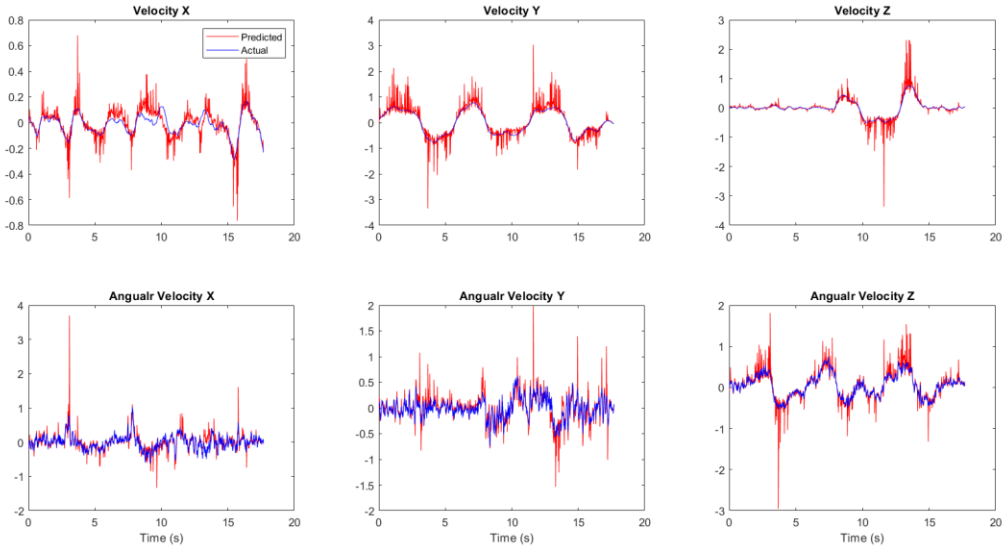
The velocities will be calculated in velocityRANSAC.m function. You can see the parameters for Probability of Hitting Inliers in the files, m = 3, psucees = 99%, e = 0.8. Now, we have to choose those points in the random algorithm which has maximum inliers. We are removing the points which are outside of the particular threshold.



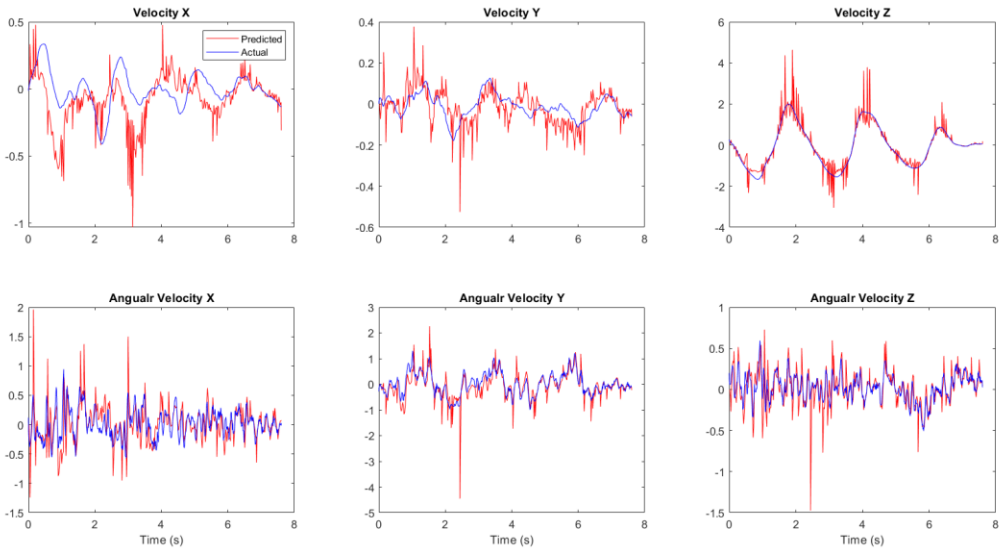
Pose Estimation for Dataset 1



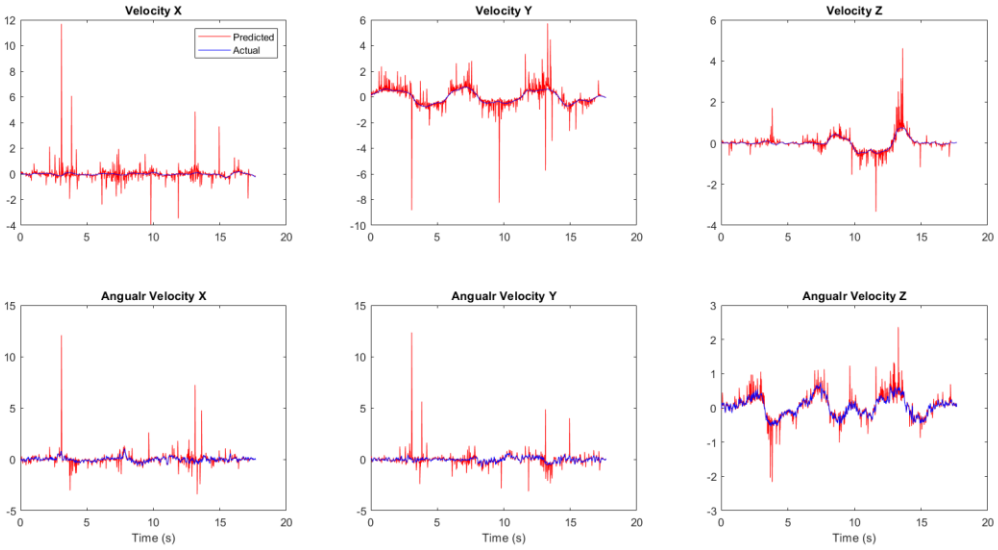
Pose Estimation for Dataset 4



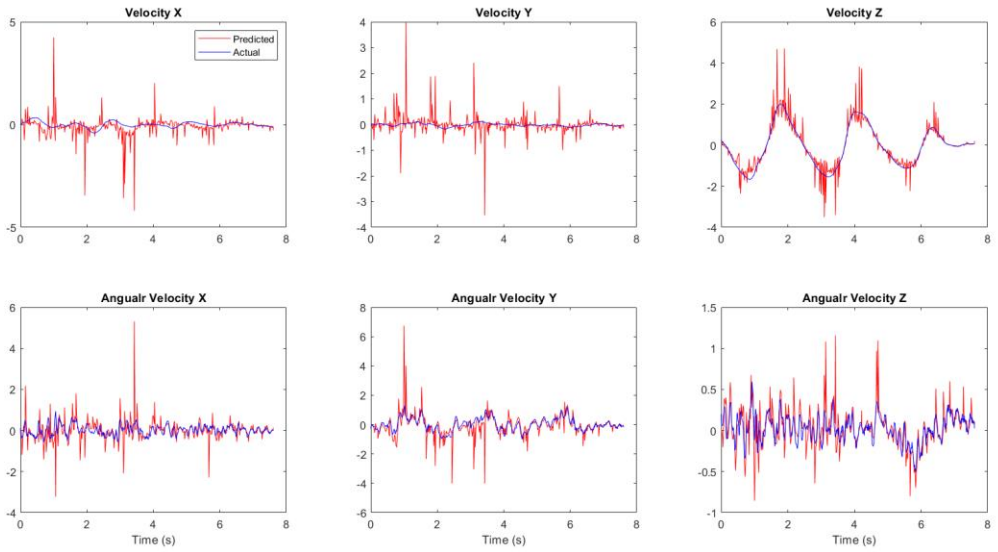
Optical Flow Velocity Estimation of Camera for Dataset 1



Optical Flow Velocity Estimation of Camera for Dataset 4



Optical Flow Velocity Estimation with RANSAC for Dataset 1



Optical Flow velocity estimation with RANSAC for Dataset 4

Results and Conclusions – As you can see the results from the graphs mentioned above. RANSAC is matching much with the ground truth, however, every graph has spikes. These spikes can be removed using filters.