**New York University**

# Vision Based 3-D Pose Estimator of Quadrotor

## ROBOT LOCALISATION AND NAVIGATION

ROB-6213 Spring 2021

PROJECT-2

Professor: Giuseppe Loianno

Student name: - Rishi Eswar Varma Vegesna (N17479594)

Student Netid: - rv2210

Student Name:- Rishi Eswar Varma Vegesna
Student Netid:- rv2210

# INTRODUCTION:

The project was vision based 3-D pose estimator of Nano+ quadrotor that was flown through a prescribed trajectory over a mat of AprilTags. The project was divided into 2 parts, in the first part from the given data the 3D pose which was position and orientation of the quadrotor in the world frame was estimated. The second part of the project was optical flow estimation. There was also a sub-part in part 2 of the project which was RANSAC implementation. The main functions of both parts of the project were GetCorner Function in which we calculate the coordinates of each April tag on the April tag mat, estimate pose function in which the position and orientation of the quadrotor was calculated with respect to world frame. In part 2 of the project, we also need to implement a low pass filter to reduce the noise between actual and predicted 3-D pose plot. Each of these parts and main functions will be discussed more clearly in the next sections.

# PROJECT PART 1 – POSE ESTIMATION

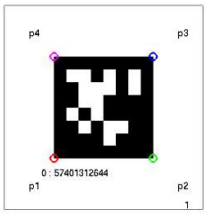
In this part of the project the position and orientation of the quadrotor in world frame were calculated and compared in the plot with predicted position and orientation of the quadrotor. The pose estimation code contains 5 sections

## Init function

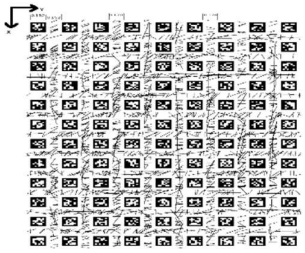
This function was used to initialise the data and read the sample Data, Vicon Data and times from the given data. This was already given as the part of the code and should not be modified

## GetCorner function

This function was used to calculate the coordinates p0,p1,p2,p3,p4 of April tags. The point p4 was calculated and with p4 each coordinate were calculated. The dimensions of each April Tag and the distance between each April tag in the April tag mat were mentioned in the handout. Initially a loop was given to read and calculate the coordinates of each April tag that was detected by the quadrotor during that particular time stamp. Once the coordinated are calculated they we made into a matrix of order 10 x 1 double. Then this process is repeated for every April tag in each time stamp and all the resulting coordinate matrices are concatenated horizontally and the final coordinate matrix was given as output of the function.




P4 is the reference point with which we Calculating coordinates



April tag mat

## Estimate pose Function

The main objective of this function was to estimate the pose of the quadrotor in world frame. In this function initially the camera intrinsic matrix was defined. This matrix was given along with handout for the project. From the given data p0 dash, p1 dash,p2 dash ,p3 dash,p4 dash values were calculated and the resulting coordinated matrices are concatenated for every time step present in the data. The next step was to calculate the H matrix for this we need the $9^{th}$ column of V matrix which can be obtained by applying singular value

decomposition to coordinate matrix. Once the 9th column of V matrix was estimated we have to make the column matrix into a 3x3 matrix which was our H matrix. Once we have the H matrix, we have to normalise the H matrix to reduce the noise in our calculation. For this the H matrix should be divided by the singular value. This Singular value can be obtained by applying singular value decomposition to the H matrix. Once the H matrix was normalised the sign of H matrix should be corrected. We then have to multiply the inverse of camera intrinsic matrix to the normalised sign corrected H matrix to get the projective transformation rotation matrix. Now the rotation matrix of and translation matrix of world with respect to camera frame were calculated from the formulas below. From the top view and side view pictures given along with the handout the transformation matrix of camera frame with respect to body frame can be calculated. By multiplying the two transformations in proper order and inversing the resultant matrix gives us the transformation matrix of body with respect to the world frame in current frame. The required position and orientation for the pose estimation can be calculated from this transformation matrix and given as outputs of the function.

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} {}^{c}R_{w} & {}^{c}T_{w} \end{pmatrix} \begin{pmatrix} X_{w} \\ Y_{w} \\ Z_{w} \\ 1 \end{pmatrix}$$

where x, y were pixel coordinated
$X_w, Y_w$ were world image coordinates
K was Camera intrinsic matrix

## Pose Estimation

This is the main program of the project part 1 which initialises the functions init, GetCorner, Estimatepose, plot functions and calls them for every time step.

# Project part-2 Optical Flow

In this part the pose of the quadrotor was calculated and the optical flow from one image to next image was also calculated. This part also consists of two sub parts that were RANSAC implementation and low pass filter implementation.

## GetCorner

This function was same as the GetCorner function in part 1. This function calculates the coordinated of each April tag present on the April tag mat.

## Estimate pose

This function was also same as the Estimate Pose function in part 1 of the project. The pose of the quadrotor was calculated by using the GetCorner coordinate matrices, H matrix was found by applying SVD to the 9th column of Coordinate matrix. Then the H matrix was normalised, and the sign of H was corrected. Finally, the transformation matrix of quadrotor body with respect to body was calculated.

## Optical Flow

This was the main function of project part 2. This function implements GetCorner, Estimate pose, plot, Velocity RANSAC function. Firstly, in this matrix we have to declare the camera intrinsic matrix and initialize loop load images i.e., we have to get pixel coordinated data from previous and current images. Once the images pixel

coordinated were loaded the good points that are visually distinguishable points in both images should be detected. Initially for this purpose the best 150 good points were selected using an inbuilt function but selecting only 150 good points may sometimes not contain the necessary image date for optical flow so to avoid this all the good points were calculated. These good points were detected using vision.PointTracker function in computer vision Toolbox in this Max bidirectional error was used for better reliability this Max bidirectional error calculates the error in coordinated of good points in forward track and backward track. After getting good points the corners were estimated using the detectFASTFEATURES and the location data of this corners were stored in a variable called POS. Once we have this, we have to initialize the tracker to the last frame and find the location of next points in the current image.

Now we have to calculate the velocity, this can be done by finding the change in the position of X and Y in current frame and last frame and dividing that with change in time. Now, we have to calculate the height of the quadrotor for this we reverse calculate the translation matrix of quadrotor in world frame with respect to camera frame. Now once Z was calculated the Motion Field equations were developed from this and store in u_v matrix in the code. And finally, we have to fix the linear velocity i.e., to change the frame of computed velocity to world frame.

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

Where, u, v are velocity of x and y of previous and current image

x, y are pixel coordinates

$V_x$, $V_Y$ are velocity of camera points

## VelocityRANSAC Function

Initially in this function we have to calculate the number of iterations for this we have to define probability of success, probability constant and minimum number of points for the model. So once we have the number of iterations we then have to initialise a for loop in which 3 random inliers are picked and the H matrix was calculated. After this optical velocity was calculated and the frame of the velocity was adjusted by multiplying it with inverse of H matrix. Finally, from this function the velocity matrix was given as output to optical flow which then plots the graphs.

## Low pass Filter

A low pass filter was implemented to reduce the noise in the actual plots. From the plots the actual values at peaks are overshooting so to reduce the noise i.e., the overshooting a low pass filter was implemented. This low pass filter was an inbuilt function Savitzky-Golay filter

# Plots and results

There were total of 6 plots 2 plots for part1 of the project and 4 plots for project part 2 with and without RANSAC. The plots are as follows

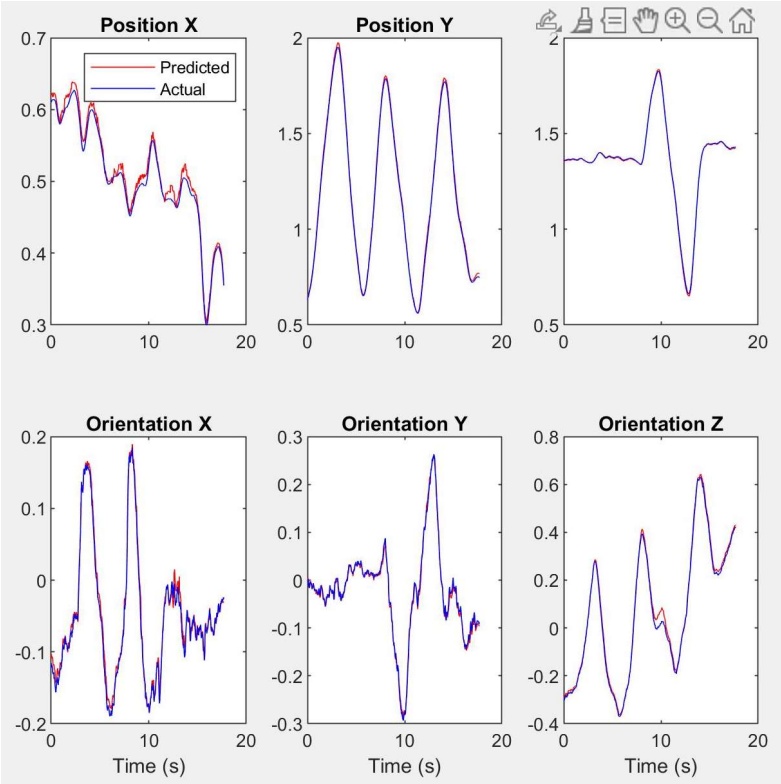



*Figure 1: Project part1 pose estimation DATASET 1*

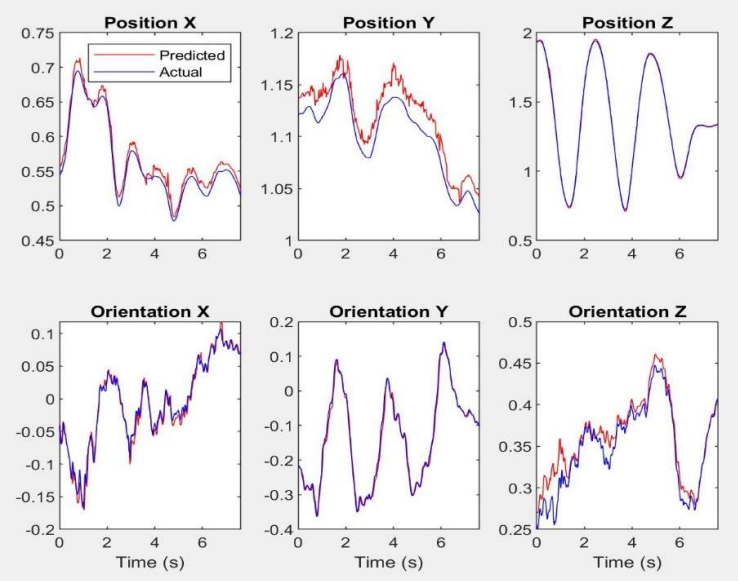



*Figure 2: Project part1 pose estimation DATASET 4*

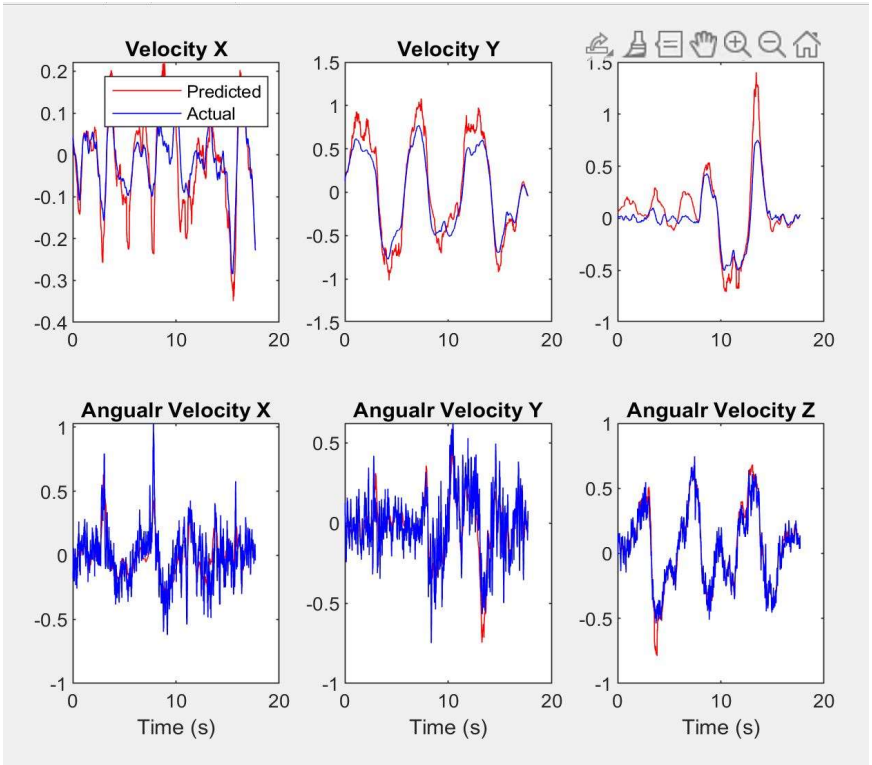



*Figure 3: Project part2 without RANSAC  DATASET 1*

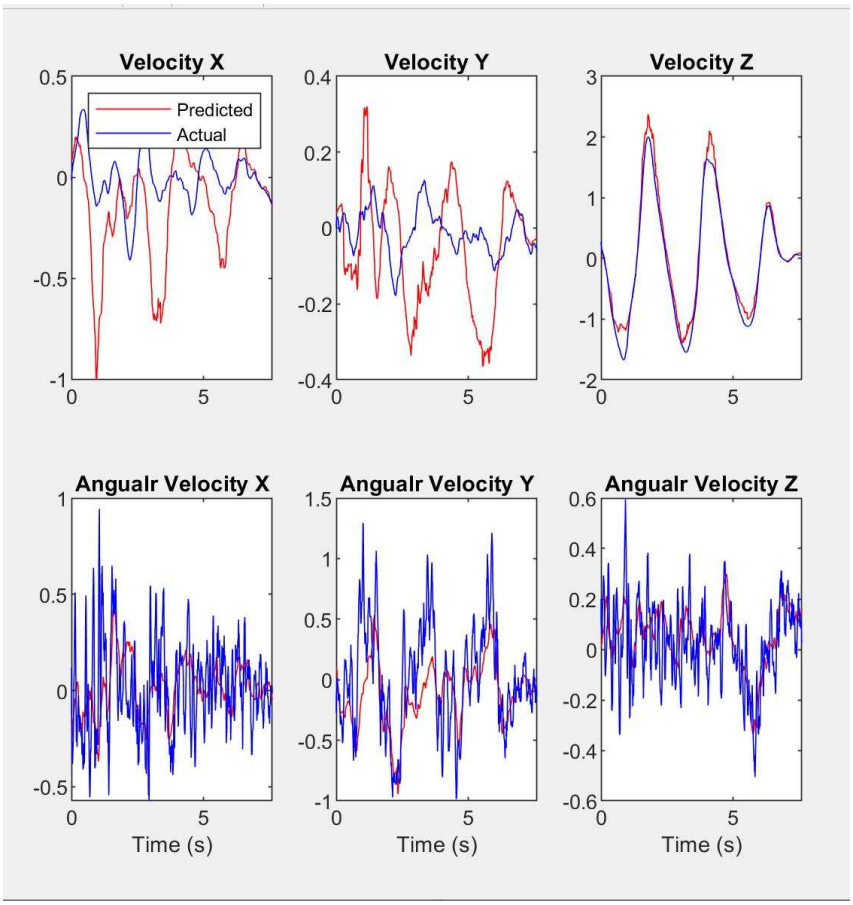



*Figure 4: Project part2 without RANSAC  DATASET 4*

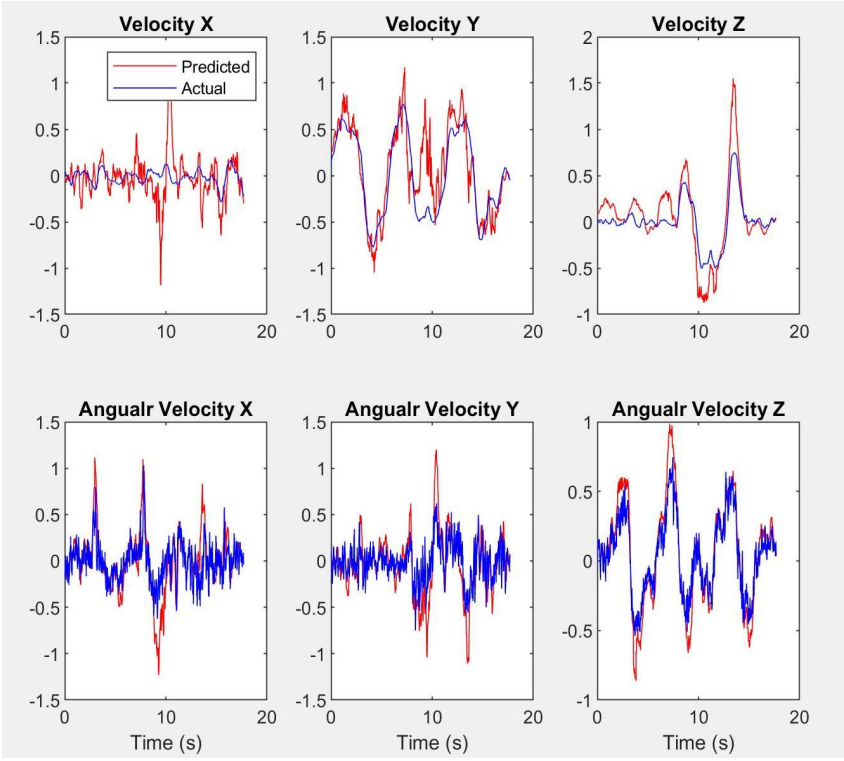



*Figure 5: Project part2 with RANSAC DATASET 1*

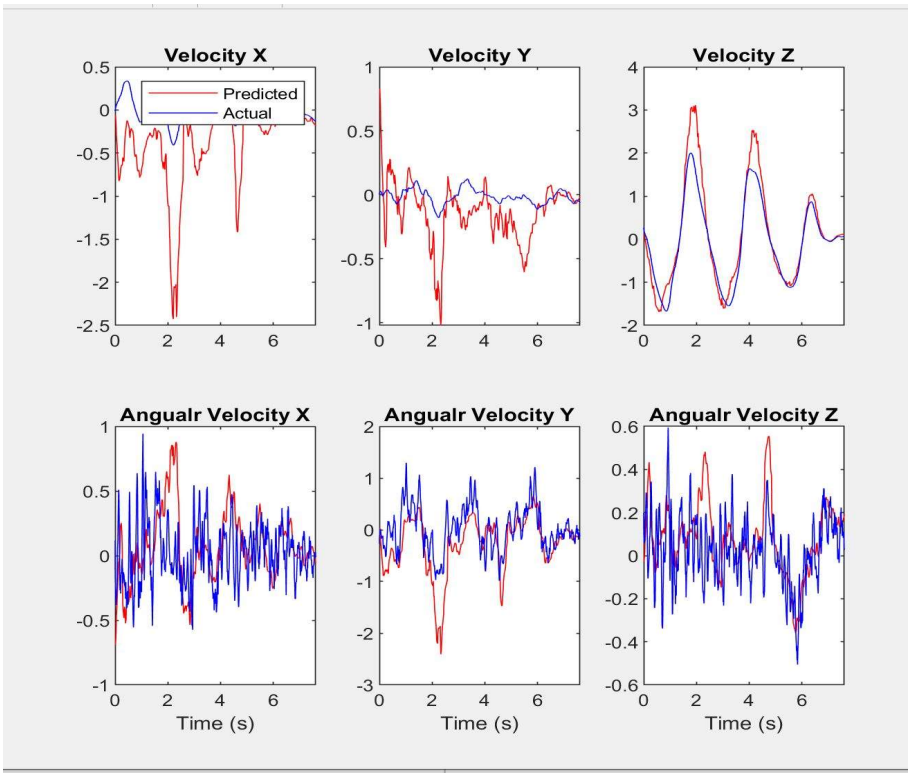



*Figure 6: Project part2 with RANSAC DATASET 4*

The plots of vision based 3-D pose estimator were as above. For part1 of the project Pose estimation the plots of DataSet number 1 were perfectly coinciding with the predicted plots whereas the plots of DataSet number 4 were slightly offset from the predicted plots. This may be due to inconsistencies in the sampled data or due to precision array lacking in the SVD functions. For part2 of the project Pose estimation the plots of DataSet number 1 were coinciding with the predicted plots but at the peaks the values were having more noise. This noise at peaks can be minimised by using a low pass filter which avoids large error values and minimises them. For this project the low pass filter available in MATLAB named Savitzky-Golay filtering was implemented. This Low pass filter reduced the noise in the plots and the graphs were implemented. The plots of Dataset number 4 were also similar to Dataset number 1. The RANSAC implemented plots were offset from predicted plots when compared to RANSAC not implemented plots. This offset was due to various factors in RANSAC code implementation i.e., the number of inliers, the number of iterations performed, the probability constants. Overall, the plots were almost accurate with some noises at peaks, which could be improved by using low pass filter.

# REFERENCES

1. Professor Giuseppe Loianno's Lecture notes
2. State Estimation for Robotics Chapter 3
3. Chapter 4- Y. Ma, S. Soatto, J. Kosecka, S. Shankar Sastry, "An Invitation to 3D Vision", Springer