Robot Localization and Navigation Project 2

Ali Rasteh
netID: ar7655

April 6, 2024

Problem Definition and Pre-requisites

In this project, we are supposed to use AprilTags as the landmarks detected in the environment by a camera and estimate the pose of the camera with respect to the world frame using these features. The second part of the project needs us to estimate the velocity of the camera with respect to the world frame using feature tracking.

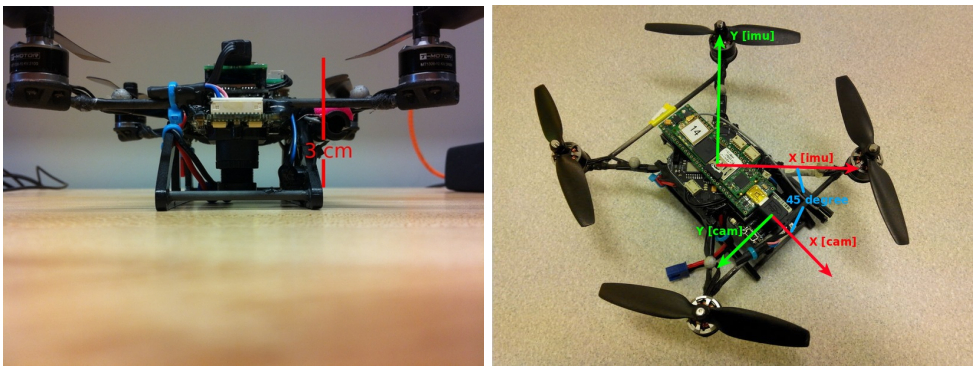
As shown in Fig. 1, the camera is attached under a drone and expressed with frame $\mathcal{C}$ while the drone's body frame coincides the IMU frame and is indicated with $\mathcal{B}$. Based on Fig. 1 and the provided parameters, the homogeneous transformation from the IMU to the camera is as follows:

$$ {}^{\mathcal{C}}\mathbf{T}_{\mathcal{B}} = \begin{bmatrix} cos(\pi/4) & -cos(\pi/4) & 0 & -0.04 \\ -sin(\pi/4) & -sin(\pi/4) & 0 & 0.0 \\ 0 & 0 & -1 & -0.03 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

Part1: Vision Based Pose Estimation

Models and Methods

Part 1 of the project relies on an AprilTag matrix pattern that enables us to estimate the homography transformation between the image plane and the matrix plane. The AprilTag library in Matlab is able to detect and correspond the corners of the image to the corners of the object in the world frame based on specific patterns. As shown in Fig.2, the spacing and the number of tags is known. So we first convert the id of the detected markers to the world frame coordinates. The function "getCorners(id)" in this project is exactly designed to do this task. In this function, we used the id to detect the position of the AprilTag and then computed the four corner coordinates using the dimensions shown in the figure.



(a) Side View (b) Top view

Figure 1: The drone and installed camera used to do this project.

We then construct this system of equations to get the elements of the homography matrix, using the correspondences between the image points and their world coordinates:

$$\begin{bmatrix} & & & & & \vdots & & & \\ x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \\ & & & & & \vdots & & & \end{bmatrix} \mathbf{h}_{9\times 1} = \mathbf{0} \tag{2}$$

The solution to the above linear system of equations is the right singular vector corresponding to the smallest singular value of $\mathbf{A}$ meaning $\mathbf{V}(:, 9)$. We reshape the solution vector into a 9×9 matrix $\mathbf{H}$ and resolve the sign uncertainty by multiplying the sign of the last element of $\mathbf{V}$.

$$\mathbf{H} = sgn(\mathbf{h}(9)) * \mathbf{H}; \tag{3}$$

The last step is to decompose the tomography matrix into the rotation and translation of the world frame with respect to the camera frame. A homography matrix is constructed as $\mathbf{H} = \mathbf{K}[\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}]$ where $\mathbf{K}$ is the camera intrinsic parameters, $\mathbf{r}_1, \mathbf{r}_2$ are the first and second columns of the rotation matrix, and $\mathbf{t}$ is the translation vector from the camera frame to the world frame. First, we multiply $\mathbf{H}$ by the inverse of the intrinsic parameter $\mathbf{K}$ and then use the first and second columns to construct a rotation matrix and derive the closest orthonormal matrix for it by using the SVD of the results.

$$\begin{aligned} \hat{\mathbf{R}} &= [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_1 \times \mathbf{r}_2] \\ \mathbf{USV^T} &= SVD(\hat{\mathbf{R}}) \\ \mathbf{R} &= \mathbf{U} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(\mathbf{UV}^T) \end{bmatrix} \mathbf{V^T} \end{aligned} \tag{4}$$

And the translation vector t is computed by normalizing the last column of the homography matrix.

$$\mathbf{t} = \frac{\mathbf{H}(:, \mathbf{3})}{||\mathbf{r}_1||_2} \tag{5}$$

Part 1 Results

The results for the first part of the project are shown in Fig. 3 and 4. As shown in the figure, in most of the cases the estimations are tracking the actual positions and orientations, but in some of them, we have insignificant errors.

Figure 2: The picture of used AprilTags for the purpose of pose and velocity estimation.

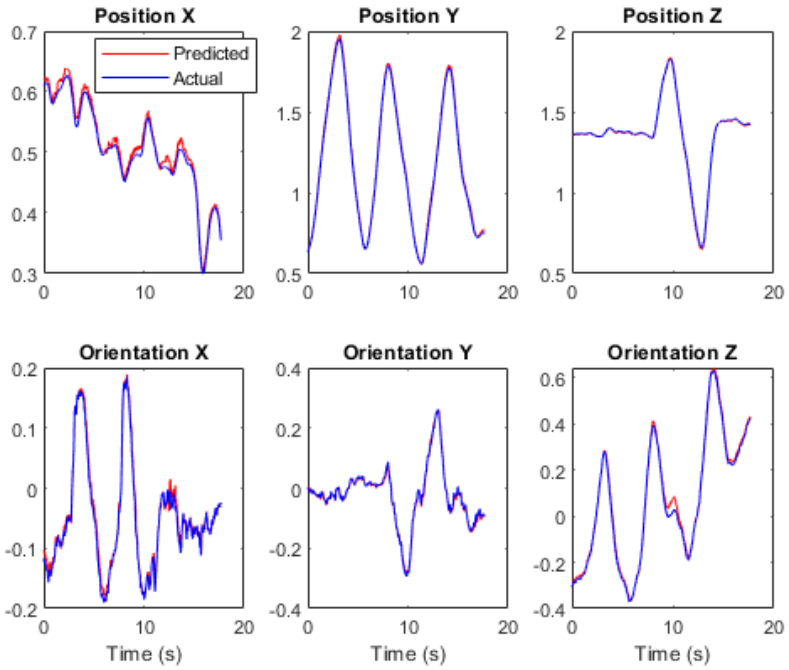


Figure 3: Results of part 1 of the project for dataset1. As shown in the figure in most of the cases the estimations are truly tracking the actual values.

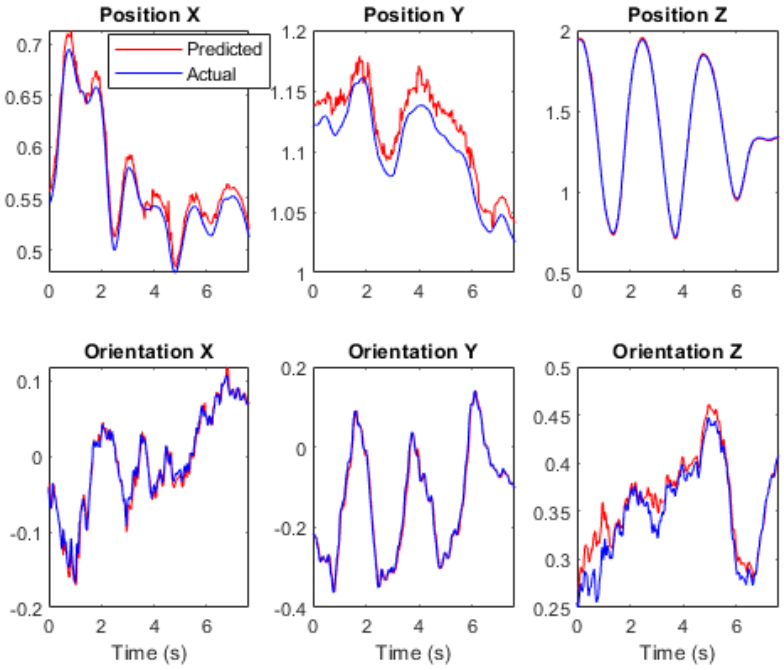


Figure 4: Results of part 1 of the project for dataset4. As shown in the figure in most of the cases the estimations are truly tracking the actual values.

Part2: Corner Extraction and Tracking

Models and Methods

In the second part of the project, we are going to estimate the velocity of the camera with respect to the world frame using the pose estimations from the last part and the movement of features in the sequential frame. We use a

general-purpose feature point extractor named FAST tracker and a tracker named KLT in this project. Then using the detected corresponding points in two frames we compute the optical flow of the images.
The first step is to compute the depth of features in the camera frame. This quantity is necessary for computing the optical flow later in this section. This is archived through the projective geometry equations as follows:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\mathbf{K}[\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}]}_{\mathbf{H}} \underbrace{\begin{bmatrix} X_w/\lambda \\ Y_w/\lambda \\ 1/\lambda \end{bmatrix}}_{\mathbf{X}_{norm}} \Rightarrow \mathbf{X}_{norm} = \mathbf{H}^{-1} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix}. \tag{6}$$

where u, v are the coordinates of the detected features in the images, X_w, Y_w are the coordinates of the feature points in the world frame, and λ is the scale of the projective geometry. We use the results from the previous section to reconstruct the $\mathbf{H}$ for this section and use it to find the scales of the projective geometry alongside the normalized world coordinates of the features.

In the next phase, we use the computed scale and find the feature points in the camera frame:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \Rightarrow \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \lambda \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{7}$$

From the above equations we can find the feature location in the camera frame $\mathcal{C}$ but still we need the values on the image place to compute optical flows. To do that we simply divide them by Z_C and define $x = X_C/Z_C$ and $y = Y_C/Z_C$. For the previous frame, we also perform the same computations and name the resulting values x', y'. The optical flow in this report is defined as the numerical time derivative of this image plane feature locations and is computed as:

$$\dot{x} = (x - x')/dt, \quad \dot{y} = (y - y')/dt \tag{8}$$

Where dt is assumed to be constant and is computed as the expected of time difference of the stamps corresponding to consecutive frames. With this, we can write the optical flow equations as follows:

$$\underbrace{\begin{bmatrix} \vdots \\ \dot{x}_i \\ \dot{y}_i \\ \vdots \end{bmatrix}}_{\dot{\mathbf{p}}} = \underbrace{\begin{bmatrix} \vdots & & & & & \\ -1/Z_{c,i} & 0 & x_i/Z_{c,i} & x_i y_i & -(1+x_i^2) & y_i \\ 0 & -1/Z_{c,i} & y_i/Z_{c,i} & -(1+y_i^2) & -x_i y_i & -x_i \\ \vdots & & & & & \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}}_{\boldsymbol{\xi}} \tag{9}$$

Assuming known depth (Z_c), the velocity of the body camera frame with respect to the world frame and expressed in the body frame ($\boldsymbol{\xi}$) may be computed with a pseudo-inverse as $\boldsymbol{\xi} = \mathbf{A}^{\dagger}\dot{\mathbf{p}}$. In this computation, we assume each row to be valid measurements and free of outliers. However, some of the rows of the matrix $\mathbf{A}$ are outliers and this could strongly bias the estimated velocity. To fix this, we use the RANSAC algorithm. We run this algorithm in the following steps:

1. if the maximum number of iterations (K=25) is not passed, randomly select 3 samples and compute the velocity as $\boldsymbol{\xi} = \mathbf{A}^{\dagger}\dot{\mathbf{p}}$
2. Use the estimated velocity to compute the error $||\mathbf{A}\boldsymbol{\xi} - \dot{\mathbf{p}}||_2$ for each sample
3. See how many of the errors fall below a certain threshold. If there are more than 200 samples, break the loop. otherwise, record the inliers and go to 1

After running the steps above, we recompute the velocity using the final set of inliers and output the result. In determining the number of RANSAC iterations in this project, we used Eq. 12 and set the probability of success to 0.99999999 and the probability of one point being an inlier (ϵ) to 0.8.

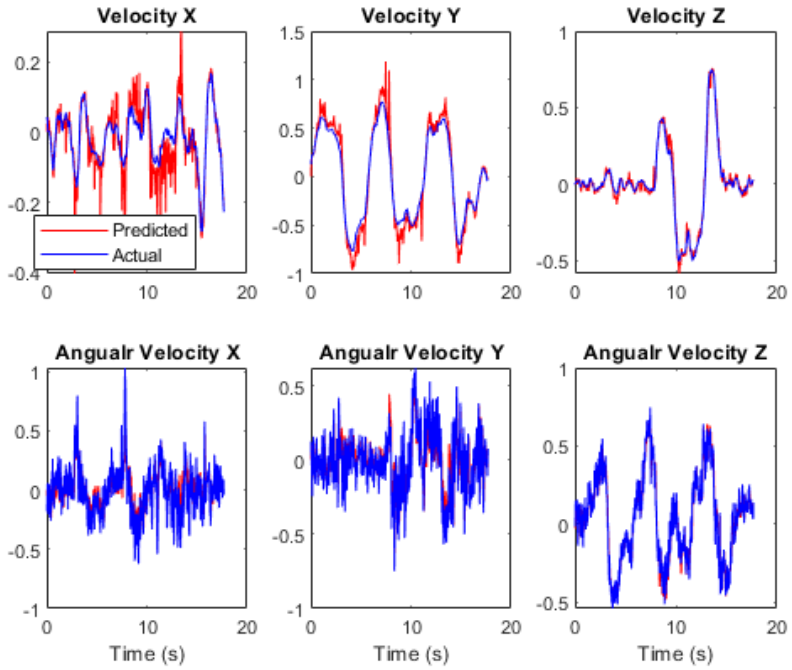


Figure 5: Results of part 2 of the project for dataset1 without using Ransac. As shown in the figure in most of the cases the velocity estimations are truly tracking the actual values.

$$K = \frac{\log\left(1 - p_{\text{success}}\right)}{\log\left(1 - \epsilon^{M}\right)} \tag{10}$$

Also, we need to use the adjoint map to transfer the velocities from the camera frame to the body frame and then use the rotation of the body with respect to the world computed from the previous section to express the result in the world frame:

$$\begin{aligned} \mathbf{v} &\triangleq \begin{bmatrix} {}^{\mathcal{B}}\dot{\boldsymbol{p}}_{\mathcal{B}}^{\mathcal{W}} \\ {}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}}^{\mathcal{W}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\mathcal{C}}^{\mathcal{B}} & -\boldsymbol{R}_{\mathcal{C}}^{\mathcal{B}} \boldsymbol{S}\left(\boldsymbol{r}_{\mathcal{C}\mathcal{B}}^{\mathcal{C}}\right) \\ \mathbf{0} & \boldsymbol{R}_{\mathcal{C}}^{\mathcal{B}} \end{bmatrix} \begin{bmatrix} {}^{C}\dot{\boldsymbol{p}}_{\mathcal{C}}^{\mathcal{W}} \\ {}^{C}\boldsymbol{\omega}_{\mathcal{C}}^{\mathcal{W}} \end{bmatrix} \\ {}^{\mathcal{W}}\mathbf{v} &= {}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}\mathbf{v} \end{aligned} \tag{11}$$

To remove the effect of high-frequency elements we use a very simple low-pass filter as follows. This is just a moving average that acts as a low pass filter in the frequency domain.

$$\begin{aligned} &\mathbf{v}[t] = \alpha \mathbf{v}[t-1] + (1-\alpha)\mathbf{v}[t] \\ &\alpha = 0.8 \end{aligned} \tag{12}$$

Part 2 Results

Figures 5 to 6 show the results of velocity estimations without using the RANSAC algorithm. As can be seen, even though the computation is faster, there were major spikes in the output. The results with the RANSAC algorithm which computes the velocity only on the detected inliers are shown in Fig. 7 to 8. As can be seen, the velocity estimations and accuracy are much better but we need to devote more computational power in order to run the RANSAC algorithm.

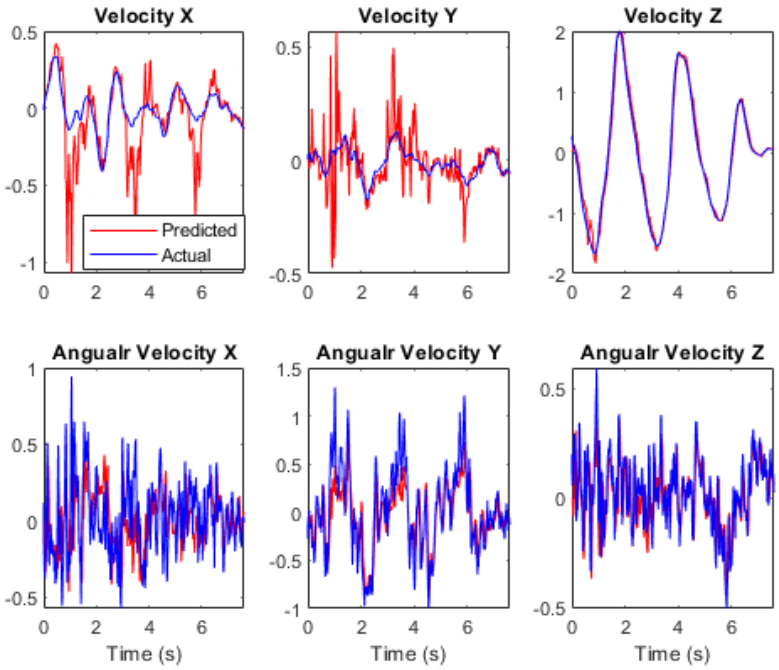


Figure 6: Results of part 2 of the project for dataset4 without using Ransac. As shown in the figure in most of the cases the velocity estimations are truly tracking the actual values.

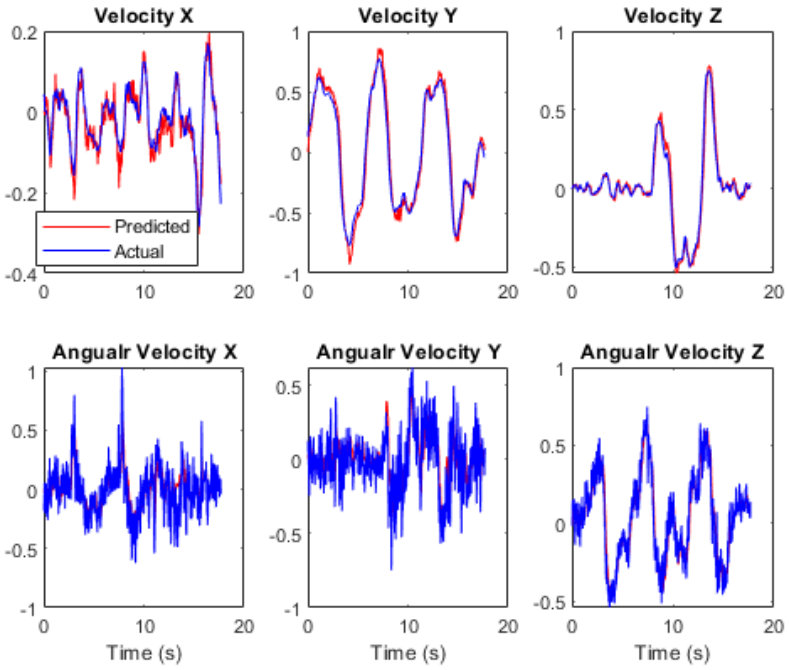


Figure 7: Results of part 2 of the project for dataset1 with Ransac. As shown in the figure in most of the cases the velocity estimations are truly tracking the actual values.

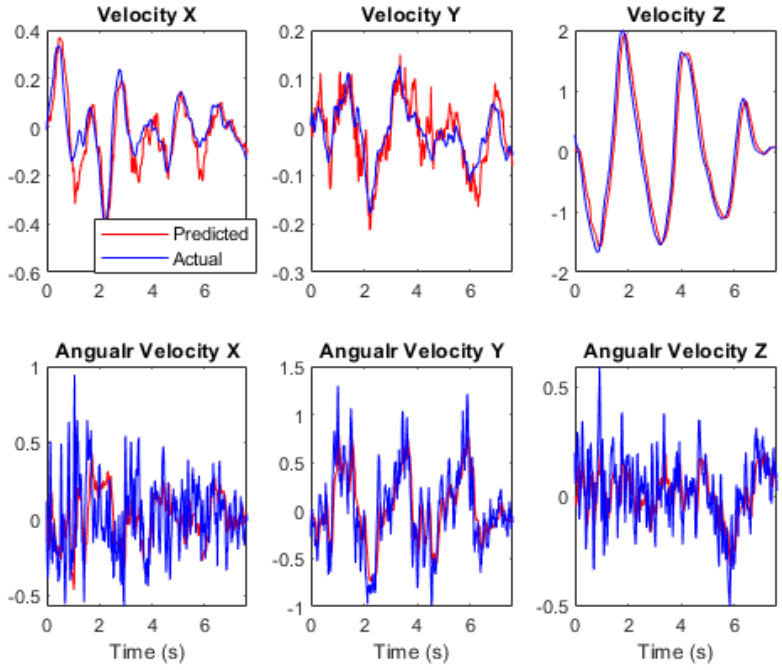


Figure 8: Results of part 2 of the project for dataset4 with Ransac. As shown in the figure in most of the cases the velocity estimations are truly tracking the actual values.