CMSC828T P2Ph2: Vision-based 3-D Velocity Estimator

Yash Manian

I. Introduction

This project builds on top of the Mapping and Localization project. The objective is to use the frames from the previous project and estimate the velocity for the trajectories based off of optical flow. The accuracy of the output is to be compared to the ground truth provided. This project involves optical flow calculation, feature tracking and outlier rejection.

II. Corner Extraction and Tracking

The method used to compute optical flow for a given set of frames is based around feature detection and tracking. The method of detection used here was detecting FAST features.

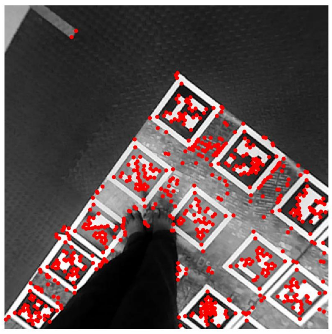
Fig. 1. FAST feature detection

These features once detected, are sent into a KLD tracker, so their corresponding features that have moved in the next frame, are correctly identified and tracked. Once these features are identified, it is a simple matter to calculate the flow using the following equations.

$$\vec{v} = \frac{x_i - x_{i-1}}{dt}$$

Where, $\vec{v}$ is the flow vector, x_i and x_{i-1} are the current and previous frame respectively, and dt is the time between frames. The resultant vector field is as follows.

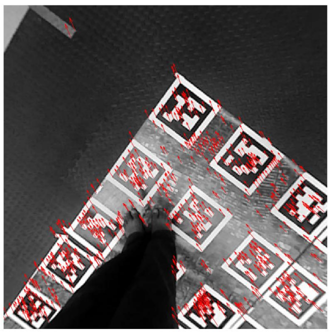
Fig. 2. Calculated vector field

III. Depth Calculation

To estimate 3D velocity, depth information on the features is required. This can be found by computing the Rotation and translation of the camera. This was done by computing homography for the known tag detections. The first step being conversion of the detected features to the camera frame as follows.

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = K^{-1} * \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{1}$$

After the homography is computed between two sets of points, the following decomposition is run to extract rotation and translation.

$$r_1 = K^{-1} * (\lambda * h_1) \tag{2}$$

$$r_2 = K^{-1} * (\lambda * h_2) \tag{3}$$

$$r_3 = r_1 \times r_2 \tag{4}$$

Here, r_1, r_2, r_3 are the columns of the rotation matrix, K is the intrinsic camera matrix, λ is the scale factor, which can be computed as follows.

$$\lambda = \frac{1}{norm(K^{-1} * h_1)} \tag{5}$$

Following this, a singular value decomposition is run on the matrix, and the two components are then transposed and multiplied to get the true rotation matrix. The translation can be achieved as follows

$$t = K^{-1} * (\lambda * h_3) \tag{6}$$

This rotation and translation are used to construct a transformation matrix for the rest of the features.

$$Rt = \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \tag{7}$$

This is then used to convert all the other feature points to the world frame coordinates.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = Rt^{-1} * \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \tag{8}$$

This depth Z is then de-normalized as follows.

$$Z = \frac{1}{Z} \tag{9}$$

IV. Velocity Calculation

Now that the depth is known, both rotational and translational velocity can be computed by six simultaneous equations as follows.

$$\begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix} = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & x*y & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{x}{Z} & (1+y)^2 & -x*y & -x \end{bmatrix} * \lceil V_x \rceil \tag{10}$$

The linear velocity is then transformed using the rotation matrix. This gives us the final linear 3D velocity. This was compared against the ground truth provided in VeliSAM2, to calculate velocity error.

A. Square trajectory

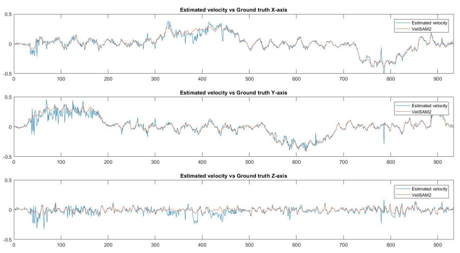

Fig. 3. Computed velocity v/s Ground truth velocity for square

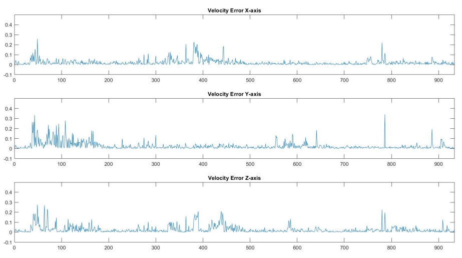

Fig. 4. Error computed between Estimated velocity and ground truth

B. Fast Circle trajectory

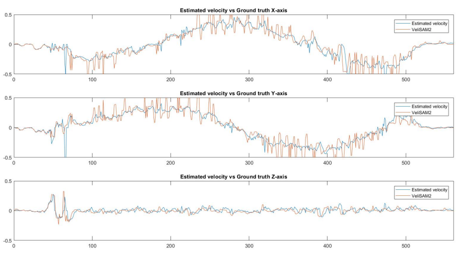

Fig. 5. Computed velocity v/s Ground truth velocity for fast circle

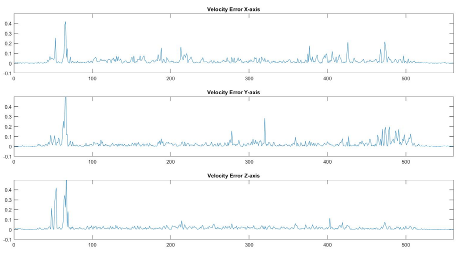

Fig. 6. Error computed between Estimated velocity and ground truth

C. Slow Circle trajectory

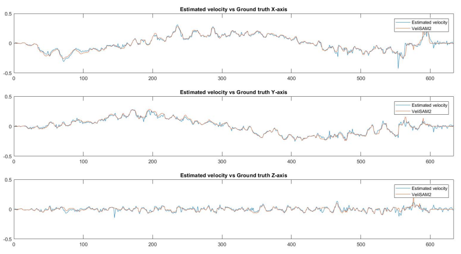

Fig. 7. Computed velocity v/s Ground truth velocity for Slow Circle

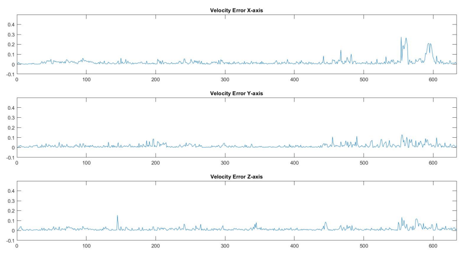

Fig. 8. Error computed between Estimated velocity and ground truth

D. *Mountain trajectory*

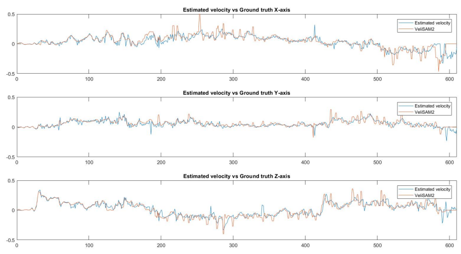

Fig. 9. Computed velocity v/s Ground truth velocity for Mountain

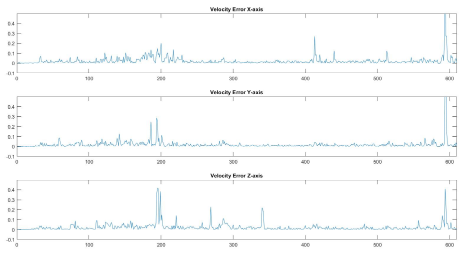

Fig. 10. Error computed between Estimated velocity and ground truth

E. *Mountain trajectory*

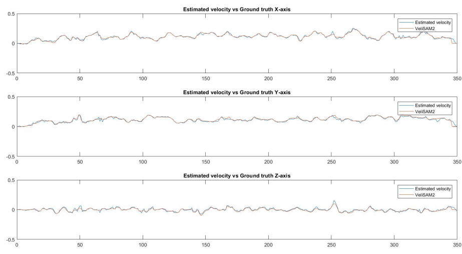

Fig. 11. Computed velocity v/s Ground truth velocity for Straight Line

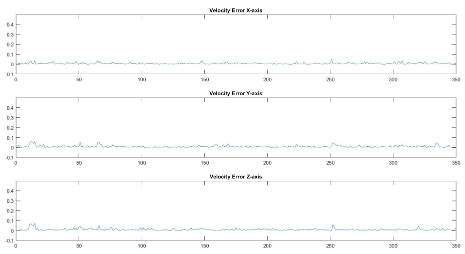

Fig. 12. Error computed between Estimated velocity and ground truth

V. RANSAC

RANSAC was to be used to remove outliers and generate a smoother velocity profile. However, using a third party library for RANSAC, and passing in the flow vector along with the matrix F, the results are almost exactly the same as the estimated 3D velocity.

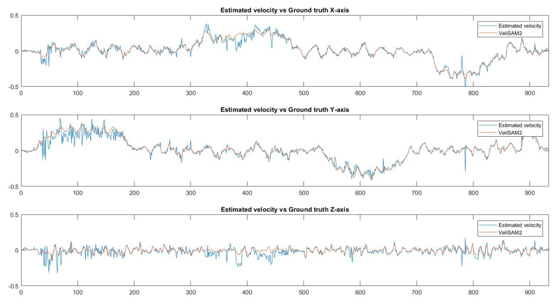

Fig. 13. RANSAC velocities for square

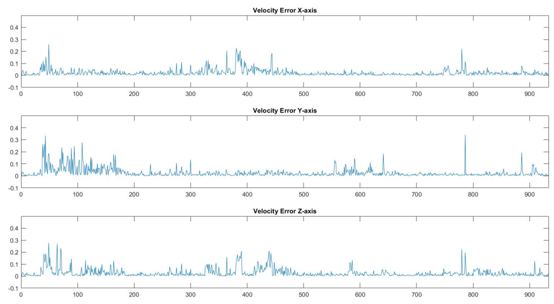

Fig. 14. Error computed between RANSAC velocity and ground truth