CMSC828T (P2Ph2): Velocity Estimator

Rishabh Agarwal[1]

***Abstract*— This paper describes how the project2 phase2 of CMSC828T class was completed. The paper talks about the calculation of optical flow and analyzing motion flow based on the output.**

I. INTRODUCTION

Based on the extraction of corner extraction and KTL tracker to calculate the optical velocity, I was able to calculate the velocity of the images between consecutive frames and using the concept of 3D velocities from optical flow from Kostas Daniilidis's lectures and David J Heeger's notes, I was able to extract the velocity of the camera between each frames.

II. LANDMARKSCOMPUTATION

For the Landmarks Computation, I implemented the same GTSAM from Project 2 Phase 1[1]. Based on the DetAll provided for all the tags I computed the landmarks position in the global frame and saved them as LandMarksComputed.mat. This .mat file was called in all the files and the required tagID cordinates were extracted from the .mat.

The plot below shows the location of all the tags extracted.

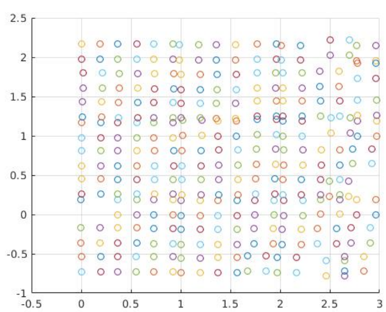

Fig. 1. Landmarks coordinates Side View

III. OPTICAL FLOW

For the calculation of optical flow, I extracted the corners from the jpg files given. First, I converted the jpg to grayscale and used detectMinEigenFeatures function of computer vision toolbox to extract the corners from the image. After the extraction of corners I used KTL tracker to track the corners in consecutive frames. It was important to make sure that we track the corners extracted in frame1 and frame 2 that are common otherwise the matrix imbalance error will occur due to different number of corner extraction [2].

*This work was completed with the help of Teaching Assistants of CMSC828T

IV. 3D VELOCITY EXTRACTION

The theory from the notes of motion estimation[3] was extremely helpful as it carefully explained the concepts of inferring the relative 3d motion between the camera and objects in the scene inferring the depth and surface structure of the scene and segmenting the scene using the motion information. I used the optical flow equation as described in equation 1 to calculate the velocity.

$$u(x,y) = p(x,y)A(x,y)T + B(x,y)\Omega \qquad (1)$$

where A, p and B are expanded in the cited paper [3].

The above equation required the Z value for the points with respect to the camera position and to extract the z I performed the homography between the absolute landmark position calculated from the GTSAM and the observed landmark position in the camera frame.

Velocity plots are shown the the figs below. The red line indicated the velGTSAM provided to us and the blue line indicates the velocity calculated.

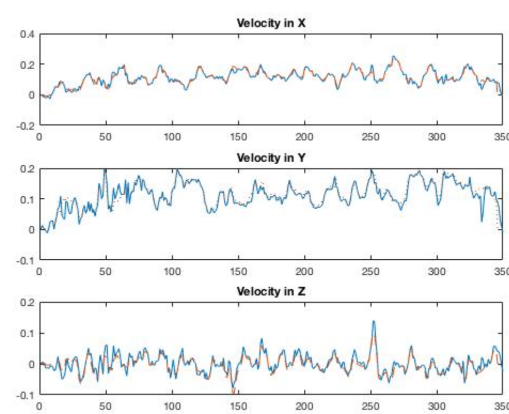

Fig. 2. Straight Line

Some visible lag between the velocities is due to the fact that I decided to skip the frames with empty DetAll whereas the answer provided considered them empty.

Error plots:

V. RANSAC

As explained in the project pdf, since the optical flow could include outliers implementing the ransac helped avoiding those and helped keeping the velocity uniform without any abrupt changes.

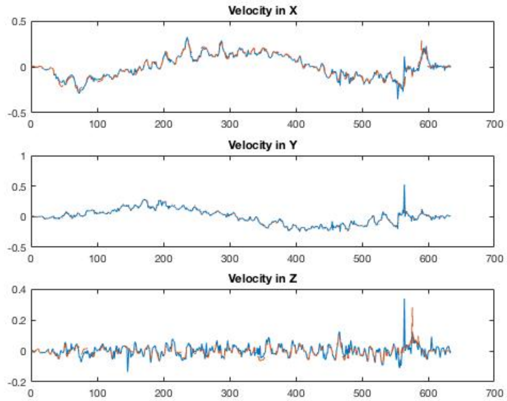

Fig. 3. Slow Circle

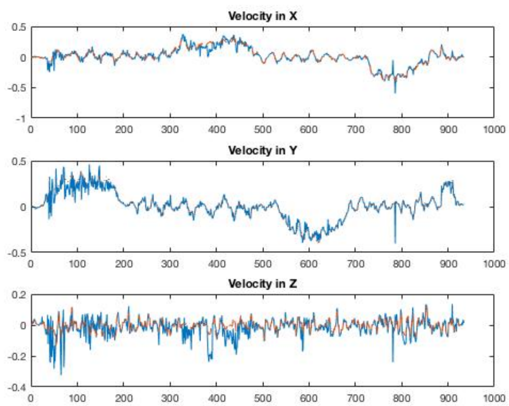

Fig. 4. Square

VI. Conclusion

As shown in the plots for the given trajectory the velocity calculated by optical flow closely follows the ground truth velocities as provided to us. The ransac still remains unclear as I am not sure how exactly the parameters are optimizing the calculation.

References

[1] Pfrommer, B., Sanket, N., Daniilidis, K., and Cleveland, J. (2017). PennCOSYVIO: A challenging Visual Inertial Odometry benchmark. 2017 IEEE International Conference on Robotics and Automation (ICRA). doi:10.1109 icra.2017.7989443

[2] https://www.mathworks.com/help/vision/examples/face-detection-and-tracking-using-the-klt-algorithm.html

[3] http://www.cns.nyu.edu/ david/handouts/motion.pdf

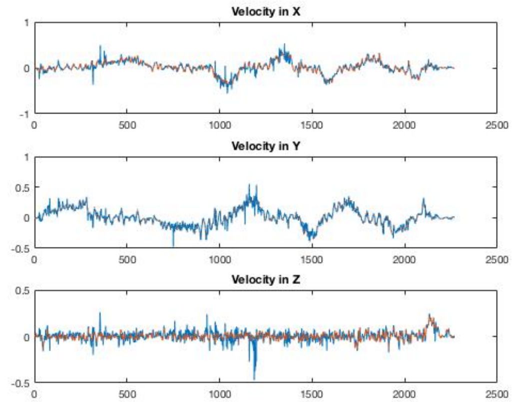

Fig. 5. Mapping

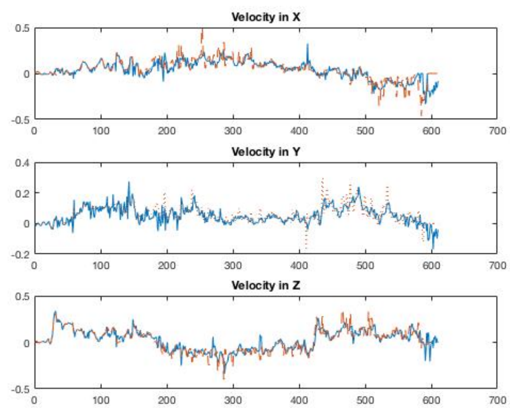

Fig. 6. Mountain

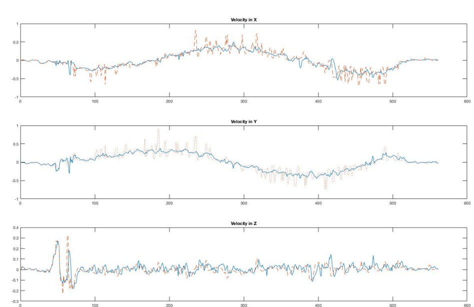

Fig. 7. fast circle

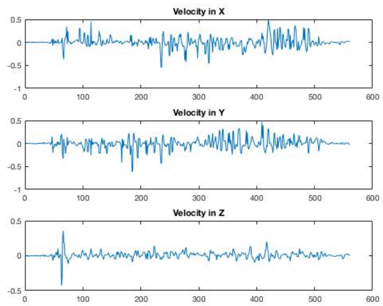

Fig. 8. Fast Circle Error

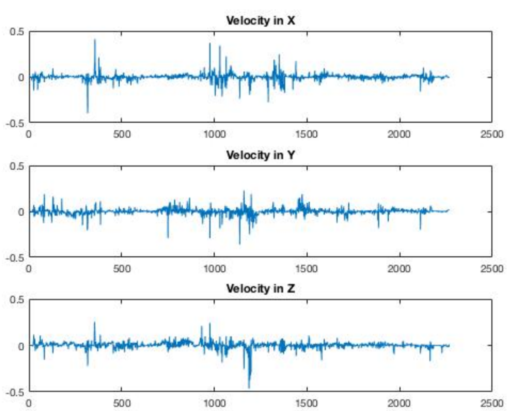

Fig. 9. Mapping Error