E9 208 Digital Video: Perception and Algorithms
Assignment 4

Vineeth S
M. Tech Artificial Intelligence
SR. No. 16543

Abstract

In this assignment, we explore the task of Visual Odometry using Nister's five point algorithm and eight point algorithm for essential matrix estimation. We implement RANSAC along with these methods for outlier rejection. Documented code will also be available at `https://github.com/vineeths96/Visual-Odometry` .

Problem 1: Visual Odometry

In this section, we discuss about the performance of the five point algorithm and the eight point algorithm for the estimation of essential matrix. We build off these algorithms based on a publicly available implementation of Visual Odometry[1].

We use the function provided by OpenCV for the estimation of the essential matrix using the five point method. Though OpenCV provides a function to estimate the fundamental matrix using eight point method (from which we can derive the essential matrix), it does not support RANSAC algorithm. Hence, we also develop our own implementation for achieving the same.

As one would expect, the OpenCV implementations are much more accurate and much faster than the implementations we develop. Especially, we find our implementations to be time consuming due to the naive sampling and implementation of RANSAC. We can trade off the accuracy and time by reducing the number of iterations for RANSAC. However, due to the cumulative nature of odometry, an error at one step adversely affects the estimate at all the successive steps.

We test our implementations and OpenCV implementations on a couple of sequences from KITTI dataset. We specifically use Sequences 03 and 10, since they are of relatively smaller size. From figures 1 2 3 and figures 4 5 6, we can observe that the OpenCV Nister 5 point method is the most accurate method. In figures 2 and 3, we can see that our implementation of 8 point algorithm outperforms that of the OpenCV inbuilt method.

One of the most important factor which we can observe is the "chain effect" of growing errors. The rotation and translation matrix update at one time instance depends on the values of these matrices at the previous time instance. Hence, an error at one time instance causes the trajectories to be different from the original trajectories — albeit maintaining a similar profile.

We use FAST feature detection algorithm to detect the keypoint locations instead of SIFT feature detection algorithm which we have discussed during the lectures. We keep track of atleast 2000 keypoints in every frame, and when the tracked keypoints in frames fall below that count, we trigger a new keypoint detection. For the OpenCV five point method, there is an efficient RANSAC algorithm that performs really well. For the OpenCV eight point method, we run RANSAC to select the inlier points and then use an eight point estimation over these points to obtain the fundamental matrix. Otherwise, due to noisy samples, the trajectories turn out to be quite different from the original trajectory. For our implementation of the eight point method with RANSAC, we randomly sample more than eight points to account for noisy data and select the inlier points. We also enforce the singular values of the essential matrix to the required values.

Trajectory 1

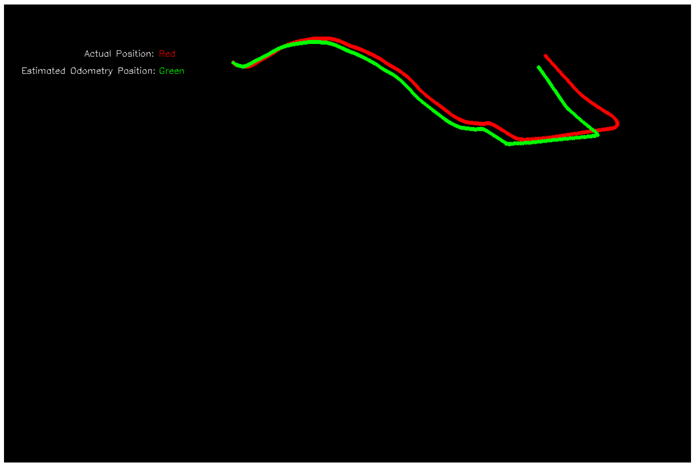

Figure 1: OpenCV Nister 5 Point Method

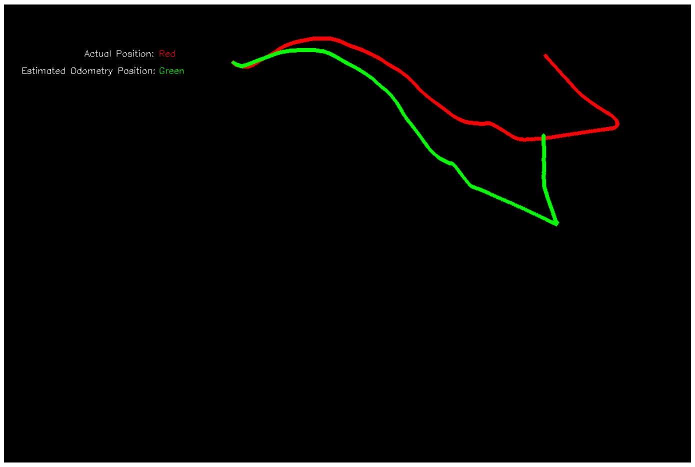

Figure 2: Own implementation of 8 Point Method

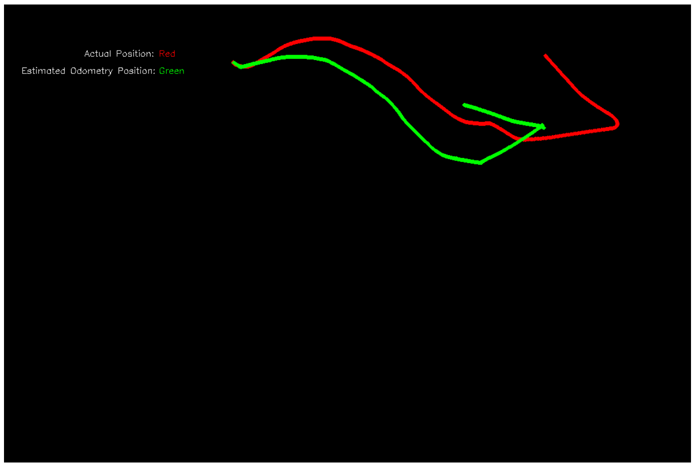

Figure 3: OpenCV implementation of 8 Point Method

Trajectory 2

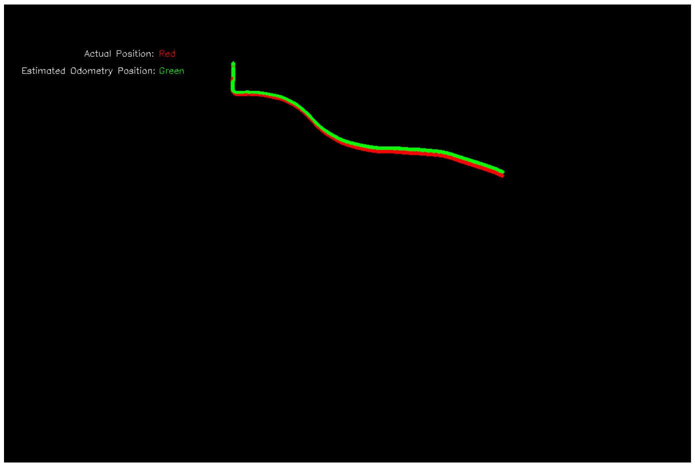

Figure 4: OpenCV Nister 5 Point Method

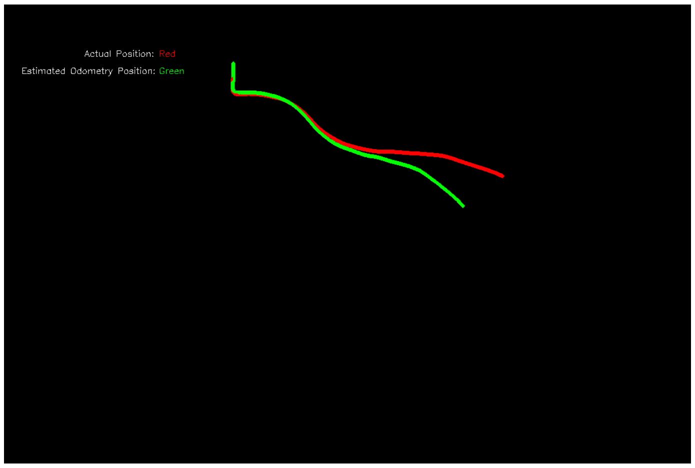

Figure 5: Own implementation of 8 Point Method

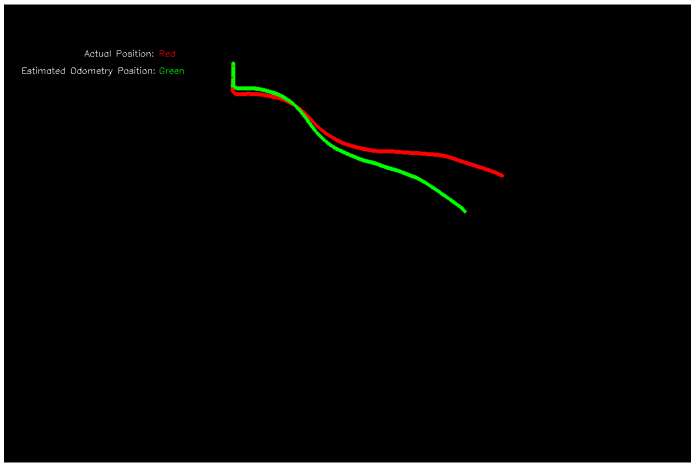

Figure 6: OpenCV implementation of 8 Point Method

References

[1] Ali Shobeiri. *Monocular Video Odometry Using OpenCV*. `https://github.com/alishobeiri/Monocular-Video-Odometery`. 2019.