ENPM673 - Perception for Autonomous Robots
Project 5: Visual Odometry

Cheng Chen, Haixiang Fang, Timothy Werder

Due May 4, 2020

Introduction

Visual Optometry and Stereo vision are powerful tools when it comes to the field of computer vision. Using a camera to infer where it moves can be used in many applications where other sensors are impractical or prone to heavy noise or failure. Stereo vision is the mathematical principal of taking images and finding the similarities between them. This can then be used to stitch together environments, track object locations or project objects into 3D space.
In this project we attempted to use a car's camera to track its locations and motion as it drives 'around the block'. We do this by analyzing the matching points using the Sift algorithm, computing the Fundamental matrix using RANSAC followed by the essential matrix, then disambiguate the essential matrix into a camera position, and determining the rotation and translation coordinates from the camera position trough triangulation and then plotting out results.

Output Video Link

https://www.youtube.com/watch?v=FuLNuOfDm2E&feature=youtu.be

Part 1: Correcting the Imagery

The first part of the project was to read in the imagery, find the camera intrinsic properties and correcting the images into usable images. This was as simple as running three commands. First the first frame intrinsic values were found. Second, each frame had was corrected to a BGR from a Bayer image and finally the images were transformed into gray scale. Because of the size of the data set we opted to do this as a prepossess and save off the corrected imagery.

Part 2: Finding Matches

The next step is the extraction of feature points by using SIFT and FLANN based feature matching. In order to extract the same points in two consecutive image frames for computing the Fundamental matrix, we need to detect features from the two frames and match those points.

We first tried both ORB and SIFT Detectors. Though ORB is faster than SIFT in terms of calculation time, the points caught by ORB are 10 times less than by SIFT for some specific frames. The difference is shown in the figure 1 and 2 below. Given the fact that more key points gave us a more accurate result when plotting the path, we decided to use SIFT algorithm in this case. After detection of key points from two consecutive frames, FLANN based Matcher was introduced to match the same features between two frames.

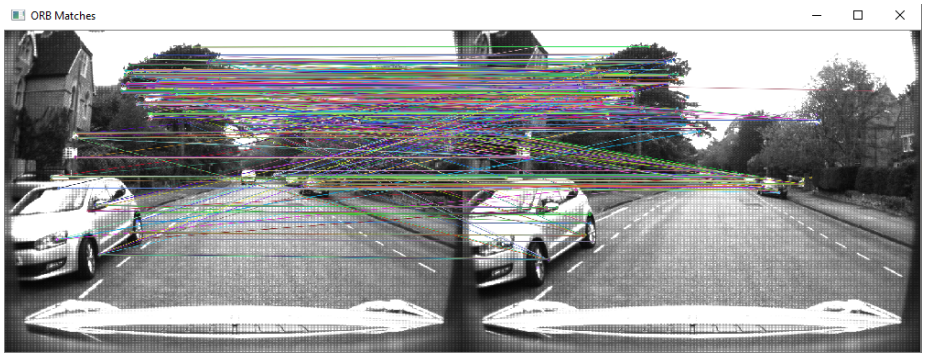

Figure 1: Matching between two consecutive frames using ORB features.

Figure 2: Matching between two consecutive frames using SIFT features.

Part 3: Creating the Fundamental Matrix and RANSAC

Once the matching pairs were found we then needed to compute the fundamental matrix. To do this in a loop, beginning our RANSAC, we select eight random matching points from the list of the matching points. We then normalize the points and use the eight-point algorithm to solve for the A matrix.

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx'_m & x_my'_m & x_m & y_mx'_m & y_my'_m & y_m & x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

Figure 3: A matrix of the 8 point Algorithm.

Then solving the linear Least Squares problem using SVD we obtain the three solutions to the Fundamental matrix. We then select the smallest solution and reshape it into the Fundamental matrix. We then compute the SVD of the Fundamental matrix and force the last element of the S matrix to be zero thereby enforcing a rank of two. Diagnalizing the S matrix and dotting it with the U and D matrix as seen below we form our F matrix.

$$F = (u \cdot diag(s) \cdot v)^T \tag{1}$$

Once we have the Fundamental matrix we then check all of the matching points with the Fundamental matrix using the formula below to find the error.

$$Error = \mid (X2^T \cdot F \cdot X1) \mid \tag{2}$$

If the error is below a selected threshold, per point, we count the point as an inlier. If the number of inliers is greater than previous runs, we save off all of the inliers and Fundamental matrix. This process is repeated for a selected number of times (200). Once all runs have been completed. We then take all of the inliers and compute one final Fundamental matrix.

Our results for the Fundamental matrix unfortunately were sub-par. Using OpenCv's findFundamentalMat function as our baseline we were unable to replicate the results coming from the function. We are uncertain as to why our performance was degraded in comparison.

Part 4: Finding the Essential Matrix

The Essential matrix is determined from the Fundamental matrix F and camera intrinsic matrix K. It is a 3X3 matrix with 5 degrees of freedom.

$$E = K^T F K \tag{3}$$

Once the Essential matrix was calculated from the above equation, its Singular Value Decomposition was calculated to deal with the noise in K. S matrix is enforced as the following way to reconstruct E.

$$E = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T \tag{4}$$

Part 5: Extracting the Camera Pose

The camera pose consists of 6 degrees-of-freedom (DOF) Rotation (Roll, Pitch, Yaw) and translation (x, y, z) of the camera with respect to the world coordinate system. From a single Essential, 4 camera pose configuration possibilities can be derived.

$$(R_i, C_i) \tag{5}$$

where i = [1, 4] After getting four rotation and translation combinations, first, Each combination is forced to have their rotation matrix determined of 1, which means, if det(R) ¡ 0, R = -R and C = -C. Second, the camera pose is given by a unique rotation and translation combination that pass the Cheirality Condition, shown in equation below

$$P = K R_i [I_{3x3}, -C] \tag{6}$$

The detial of Cheirality Condition and linear triangulation is explained in section 6.

Part 6: Disambiguation with Cheirality Condition

In previous section, four possibilities of camera poses are obtained from essential matrix. In order to estimate the camera pose, a correct unique rotation and translation is necessary. One way to remove this rotation and translation dis-ambiguity, is by checking cheirality condition of correspondence points in both rame. As we reconstructed 3D points form those correspondence points, the correct rotation and translation should make the 3D points be in front of both cameras, and the z component of the point coordinates in both camera coordinates are bigger than zero. We check this condition. In this project, the 3D points are reconstructed using linear triangulation. The ideal camera configuration, (C, R, X) is the one that produces the most 3D points satisfying the cheirality condition.

Part 7: Linear Triangulation

While imaging, A point in he 3D world coordinate is map into camera image coordinate, and the relationship is ruled by camera intrinsic and extrinsic matrix. With a single imaging, it's impossible to recover the depth of a 2D point. In another word, if there is a line from focal point to the 2D point on virtual image plane, there are infinite 3D points on the line, which can from this 2D point. However, with two imaging, it's possible to recover depth. As shown in Figure , a 3D point X projects on the left imaging has

$$x = PX \tag{7}$$

projects on the right imaging has

$$x^{'} = P^{'}X \tag{8}$$

where X is the 3D coordinates of point in world coordinate
x is the left image coordinate of the point projection
$x^{'}$ is the right image coordinate of the point projection
P is the projection matrix from world coordinate to left imaging coordinate
$P^{'}$ is the projection matrix from world coordinate to right imaging coordinate

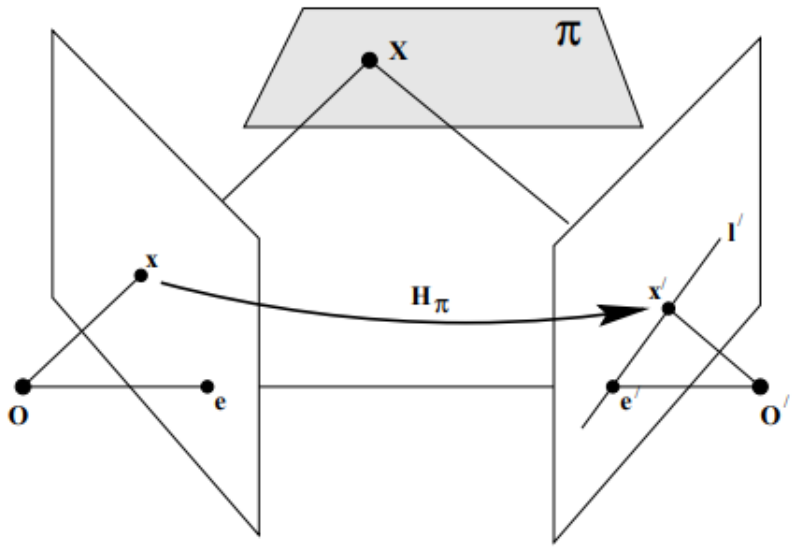

Figure 4: 3D points project on image plane.

The linear triangulation that solving 3D coordinate of the point is given by

$$\begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x^{'}p^{'3T} - p^{'1T} \\ y^{'}p^{'3T} - p^{'2T} \end{bmatrix} X = 0 \tag{9}$$

where p^i is i-th row of projection matrix
x is the x compound of left image coordinate of the point
y is the y compound of left image coordinate of the point
$x^{'}$ is the x compound of right image coordinate of the point
$y^{'}$ is the y compound of right image coordinate of the point
X is the 3D coordinates of point in world coordinate

In this project, based on the image coordinates of point correspondences, their 3D coordinate is recovered under a specific camera poses (R, C). Using the method of linear least squares, X can be solved.

Part 8: Creating the Homogeneous matrix

Since $[R \mid t]$ is a 3X4 matrix. We added one row as [0 0 0 1] to make it a homogeneous matrix.

Part 9:Visualization

Once the coordinates of the homogeneous matrix are found we append the x and y coordinates to a list. Once all desired frames have been run and the list is complete, the list is saved off as a pickle object. We then take the object into a visualization that plays the video in sync with the OpenCV run (previously saved off) and the run that was most previously completed.

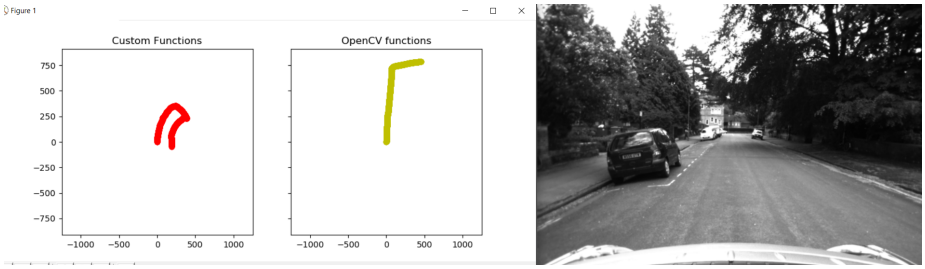

Figure 5: Animation of points be simultaneously plotted as the car drives.

Part 10: Comparison to Open CV [Extra Credit]

As mentioned we used OpenCV as a baseline to compare to for our code. Below are some examples of OpenCV vs our code.

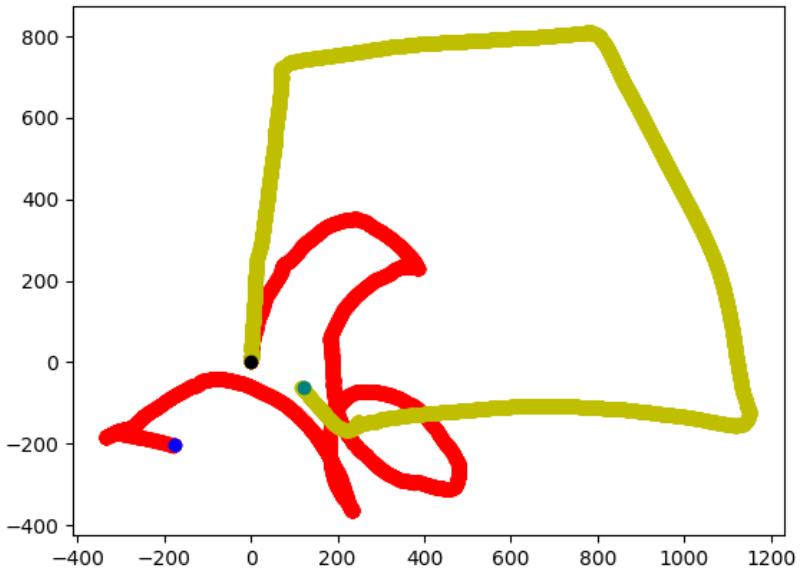

Figure 6: All frames plotted: Red: Custom code. Gold: Open CV

The next two images show relative motion of the car as it goes around the first turn and the second image shows the second turn. Note that the relative motion is correct, but because the OpenCV code is referenced from the 3700+ set and our code is built only using those points, the starting position and orientation will be different.

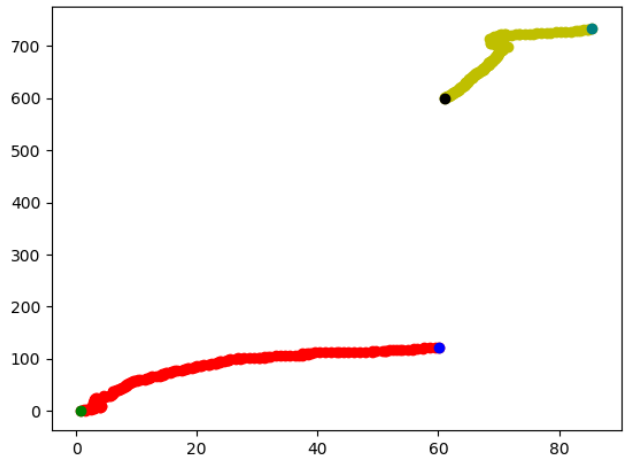

Figure 7: Turning on the first corner: Red: Custom code. Gold: Open CV

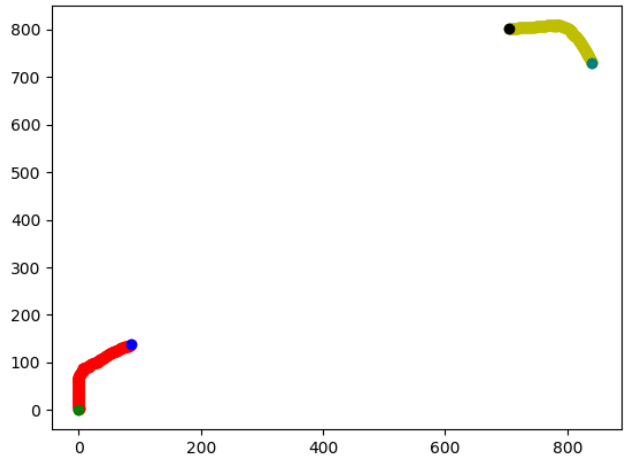

Figure 8: Turning on the second corner: Red: Custom code. Gold: Open CV

Additionally the first 500 frames also showed promise; this quickly degrades after 500 frames as the car moves back into traffic.

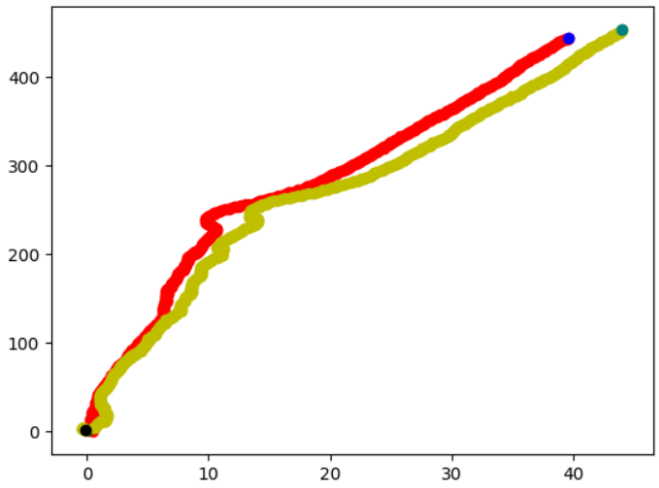

Figure 9: First 500 frames: Red: Custom code. Gold: Open CV

Part 11: Conclusion and Lessons Learned

Over all this project was very difficult for a multitude of reasons. First is the continual isolation of the team among COVID-19. This was a particularly difficult aspect with the data set size, though this was mitigated by learning of Github's gitignore feature.

Another difficult aspect of this project was the time it took to render the project to test out our code. To mitigate this we selected specific frames to test against. However this could still take up to an hour and a half to render out 500 frame severely reducing the amount of debugging that can be done.

In order to match the plot using built-in function more accurately, we could increase the number of iteration of RANSAC to compute optimal value of Fundamental matrix, though this would significantly increase the time required to produce results (Which was out of scope for the time available for this project). Additionally, instead of randomly selecting correspondences we could also use a robust algorithm i.e. Zhang's 8-point algorithm.

Additionally we noticed two additional aspects. One, that small batches of points had better accuracy than when the entire 3700+ frames were all run continuously, though we did not have enough time to compute a full set of all batches. Two, the program was hyper-sensitive to right movement when the car was moving straight. We believe that this is the main cause for our program's inaccuracy.

We believe that with additional testing, debugging and time our results would more closely mirror that of the in-built functions. We conclude, that through the results of our program do not mirror that of the in-built func-

tions, our program was well underway with and did produce some results with accuracy.