ENPM673 Project5
Perception of Autonomous Robots
Visual Odometry

By (Group 2)
B. Sai Praveen(DirID: spraveen)
Preyash Parikh(DirID: pparikh)
Ajinkya Parwekar(DirID: ajinkyap)

May 4, 2020

1 INTRODUCTION

- Visual odometry is defined as the process of estimating the robot's motion (translation and rotation with respect to a reference frame) by observing a sequence of images of its environment. Visual Odometry is a particular case of a technique known as "Structure from Motion" (SFM) that tackles the problem of 3D reconstruction of both the structure of the environment and camera poses from sequentially ordered or unordered image sets. Visual Odometry is a method of finding a robot/camera pose with respect to world frame using camera data.

- Visual odometry mainly focuses on local consistency and aims to incrementally estimate the path of the camera/robot pose after pose, and possibly performing local optimization. In this project, camera pose attached to a car is estimated by reading consecutive pair of images from the dataset. The given parameters for this project is dataset and camera calibration matrix. The output is the trajectory plot of the car.

2 PREPROCESSING

1) The images which are provided are in mosaic form. Thus, we need to de-mosaic it first. We implemented in-built function $cv2.COLOR\_BayerGR2BGR$.

2) The next step is to undistort image. We used the given ReadCameraModel.py file for un-distortion lookup table. Undistorted image is crucial for feature detection.

3) Images are converted to Grayscale as feature detection requires Grayscale images.

3 DETERMINATION OF FEATURE POINTS

This is one of the most important step as we need feature points to calculate fundamental matrix. To detect feature points from the given frame we used SHIFT detector. To match these feature we used FLANN matcher. Once the tracker detects the corresponding points in the present frame, this data is used to calculate fundamental matrix and then rotation and translation between the frames.

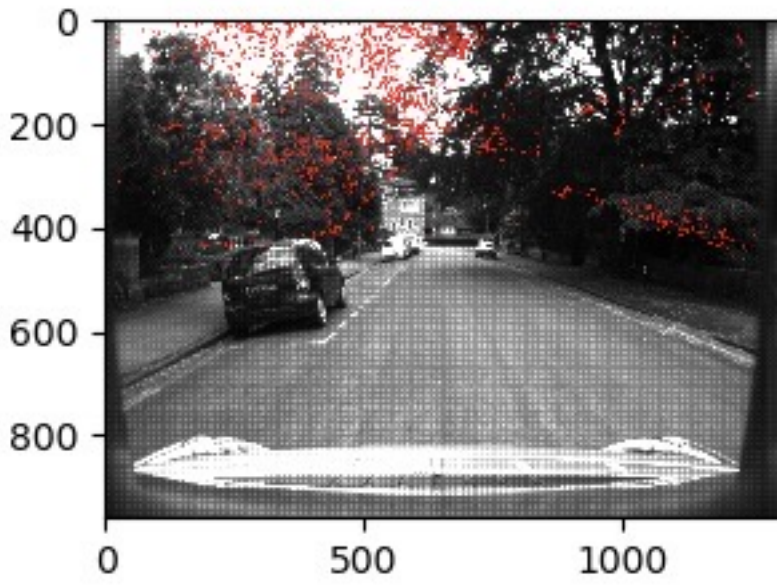

Figure 1: Feature detection

4 COMPUTING FUNDAMENTAL MATRIX WITH RANSAC

- The fundamental matrix is a 3×3 matrix which relates corresponding points in stereo images.
- The fundamental matrix is a 3×3 matrix with 9 unknowns and 7 degrees of freedom. It can be defined as,

$$x_1^T F x_2 = 0$$

where x_1 and x_2 are the coordinates of feature points in the two images. This equation was derived from epipolar geometry.
-The fundamental matrix has 9 unknowns but can be defined as one equation.

$$\begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = 0$$

After taking 8 feature point, the fundamental matrix is,

$$Af = \begin{pmatrix} x_1'x_1 & x_1'y_1 & x_1' & y_1'x_1 & y_1'y_1 & x_1' & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n'x_n & x_n'y_n & x_n' & y_n'x_n & y_n'y_n & x_n' & x_n & y_n & 1 \end{pmatrix} f = 0$$

where x_1, y_1, x_2, y_2 are the normalized corresponding points.
The solution to this problem is last column of v matrix generated by [USV] = svd(A)

After this, force the rank of fundamental matrix to be 2 even if it is not. Now, by using the transformation matrices generated by normalizing the points, re-transform back the fundamental matrix. Then divide fundamental matrix by its norm.

RANSAC

In this project, since we did not get satisfactory result with 8 point Algorithm we implemented RANSAC. We make a 8×8 grid out of the image and as per their position we assign all key points from image to the grid. Then we take 8 cells at random out of 64 cells. Also, we take one point from these 8 cells. Therefore, we have 8 points at random out of all the key points detected in an image. Now we will have some inliers and outliers. If we obtain the distance of above points from epipolar line then we will have best inliers. It is considered as good match if, the distance between epipolar line and the keypoint is less than one pixel. Thus, it is an inlier. Once the inlier is selected, we get new fundamental matrix. We discard keypoint who's distance is greater than threshold.
This entire process is repeated several times. We have iterated through 50 times.

5 ESSENTIAL MATRIX, CAMERA POSE ESTIMATION AND TRIANGULATION CHECK

It is computed using fundamental matrix and the calibration matrix. The equation is given by,

$$E = K^T * F * K$$

where K is calibration matrix and F is fundamental matrix.
Essential matrix is also a 3×3 matrix with 5 degrees of freedom.

$$Essentialmatrix = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We equate eigen values with 1, 1 ,0. In order to reduce the noise, we compute svd of essential matrix.
In order to define the camera centers we need to have camera poses. The camera pose consists of 6 Degree of Freedom, Rotation (Roll,pitch,yaw) and Translation(x,y,z) of camera with respect to world. To find the camera poses we first need to find singular value decomposition, $E = UDV^T$. And then we solve for W, given by,

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The four camera pose configurations are (C1,R1), (C2,R2), (C3,R3), (C4,R4). The resulting configurations are,

1) C_1 = U(:,3) and $R_1 = UWV^T$
2) C_2 = -U(:,3) and $R_2 = UWV^T$
3) C_3 = U(:,3) and $R_3 = UW^TV^T$
4) C_3 = -U(:,3) and $R_4 = UW^TV^T$

Make sure that the determinants of Camera Centers and Rotation Matrices are positive. If they are not positive, they are corrected. After, set of camera poses and rotation matrices are computed, the camera pose are defined where,

$$P1, currentframe = [I_{3\times3}|0]$$
$$P2, nextframe = [R|-C]$$

LINEAR AND NON-LINEAR TRIANGULATION

If we have to remove disambiguity, we need to find the correct unique pose. To obtain optimal camera pose, we used cheirality condition. This states that the point projected in world frame using camera pose must be all in the front of camera i.e. the reconstructed points must be in front of the cameras. A 3D point X is in front of the camera if, $r_3(X - C)>0$ where r is the z-axis of the camera. Not all the points satisfy this condition due to noise. The combination of R,C that produces maximum number of points in front of camera is chosen to be optimal values. For non-linear triangulation, we have two camera pose and linearly triangulated points and we can refine the location of 3D world points that will help in reducing error of reprojection error. This error can be calculated by measuring the error between the measured and projected 3D points. It is given by,

$$min_x \sum_{j=1,2} \left(u^j - \frac{P_1^{jT}\tilde{X}}{P_3^{jT}X} \right)^2 + \left(v^j - \frac{P_1^{jT}\tilde{X}}{P_3^{jT}X} \right)^2$$

where j = index of each camera.
$\tilde{X}$ is the homogeneous representation of X (World Point).
P_1^{jT} is each row of the camera projection matrix.

We used scipy built in function " *scipy.optimize.least_squares*()" to minimise the error.

6 PERSPECTIVE AND POINT TECHNIQUES

Perspective-n-Point is the problem of estimating the pose of a calibrated camera given a set of n 3D points in the world and their corresponding 2D projections in the image. The camera pose consists of 6 degrees-of-freedom. For there to exist a solution, $n \geq 3$.
After calculating the projected 3D points and respective correspondence, we obtain the optimal camera pose by applying PnP technique.
Linear PnP - 2D points can be normalized by the intrinsic parameter to isolate camera parameters, (C, R), i.e., K^-1x. A linear least squares system that relates the 3D and 2D points can be solved for (t, R) where,

$$t = -R^T * C$$

Non linear PnP - Reprojection error that is geometrically meaningful error is computed by measuring error between measurement and projected 3D point. The formula is given by,

$$min_{R,C} \sum_{j=1,2} \left(u^j - \frac{P_1^{jT} X}{P_3^{jT} X} \right)^2 + \left(v^j - \frac{P_1^{jT} X}{P_3^{jT} X} \right)^2$$

where, all the notations are same as non-linear triangulation and $P = KR[I_3 \times 3 - C]$ is the camera projection matrix. However since the rotation matrix had not been enforced with orthogonality, we use a compact representation using a quaternion (R = R(q)). Hence the equation becomes,

$$min_{C,q} \sum_{j=1,2} \left(u^j - \frac{P_1^{jT} X}{P_3^{jT} X} \right)^2 + \left(v^j - \frac{P_1^{jT} X}{P_3^{jT} X} \right)^2$$

7 CODE PIPELINE

The images were converted from Bayer format to BGR format. Then the camera calibration matrix is calculated followed by undistorting of images. The detection and matching of the feature points is done with the help of OpenCv function, SHIFT detector and FLANN based matcher. Feature points from consecutive points are extracted to compute Fundamental Matrix by randomly selecting 8 features points. We will have a lot of fundamental matrix, to select the optimal one we look at the most number of inliers. The count of inliers is done through RANSAC check. This check is done till 50 iterations. If we increase the number of iterations, we will have more accurate result but the computation time will also increase. Now after having the final fundamental matrix, we calculate the Essential Matrix with the help of camera calibration matrix. By forming four sets of camera and rotation matrices, the possible camera poses are estimated. We assumed that the +x axis is facing towards right, +y axis is facing outwards of a page and +z axis points downwards. The camera poses in direction of extreme +z axis is manually phased out and then the camera poses are signalled out by cheirality check. Now the position of camera of current frame is calculated with respect to previous frame by calculating homogeneous transformation and then multiplying it with homogeneous matrix. The center of camera frame for every frame is plotted. We also, compared user defined functions output with the predefined functions such as - cv2.FindEssentialMat and cv2.recoverPose.

8 CONCLUSION AND RESULTS

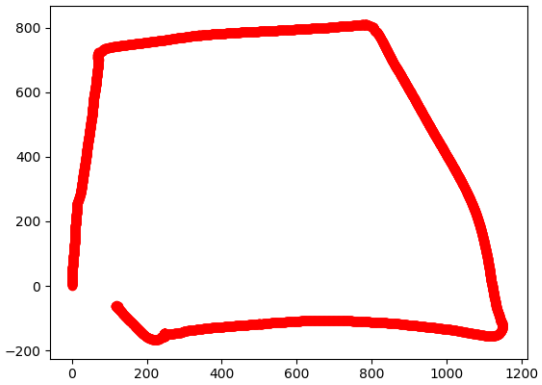

Figure 2: Pre defined function

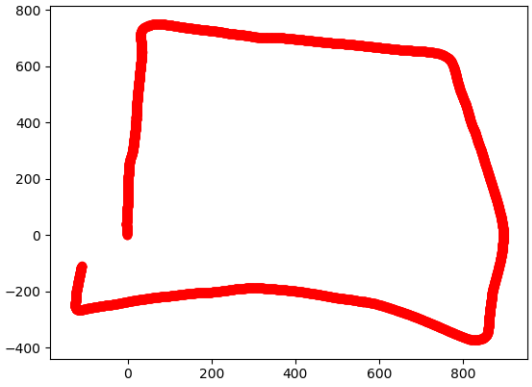

Figure 3: User defined function

The trajectory obtained is slightly different. The user defined function is not smooth as compared to predefined function. You can see that there is variation between the orientation of both the trajectories. The error could have been less if we used more number of iteration for RANSAC. We could also reduce error by using Zhang's 8 point Algorithm as it more robust.

The output google drive link is as follows.
https://drive.google.com/drive/folders/1OyYpgK7W9riYEMGYQ6BtSloxAM_dxho9?usp=sharing

9 CHALLENGES FACED

1) Long run time.
2) Shift detector is too slow. Also, we need to change the computer vision version as SHIFT is patented for version 4.
3) Error in optimizing the functions.

10 REFERENCES

1) https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html
2) https://link.springer.com/article/10.1007/s40903-015-0032-7
3) https://www.cis.upenn.edu/~cis580/Spring2015/Projects/proj2/proj2.pdf
4) https://cmsc733.github.io/2019/proj/p3/#pnp
5) Lecture Notes.