ENPM673 Perception for Autonomous Robots

PROJECT 5

Visual Odometry

Nalin Das (116698290)
Aditya Khopkar (116911627)
Nidhi Bhojak (116787529)

Contents

List of Figures

1 Introduction

Visual odometry refers to the problem of recovering camera motion based on the images taken by it. Visual odometry operates by estimating relative motion of the camera between subsequent images by observing changes in them. Later, these estimates are combined into a single trajectory.

Here, we are given the input video sequence of camera mounted on a car as well as the camera calibration matrix. The data comprises of 3873 Bayer images. We then need to determine the camera trajectory from the given image sequences using various operations.

The output can be accessed from the following link:

Video Outputs Link

The general flow of the project can be inferred from the figure given below.

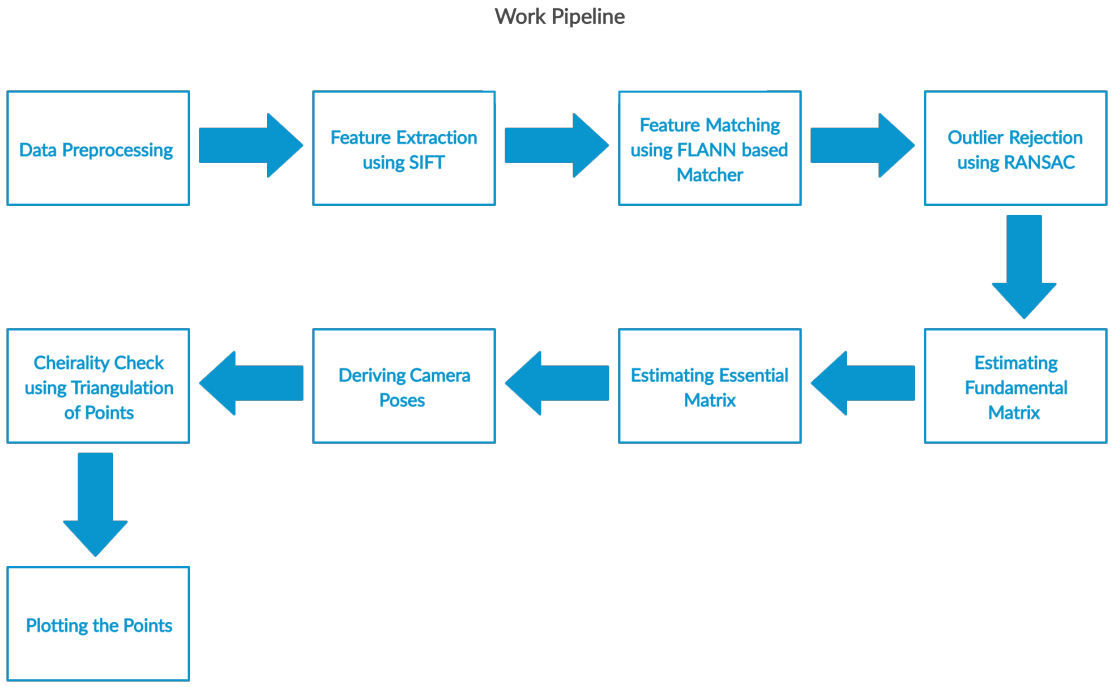

Figure 1: General Pipeline

2 Data Preparation

Before we process the data, we define a specific region of Interest to eliminate unwanted extra details.

- The images provided are in the Bayer format and hence needs to be converted in the BGR

colorspace.

- This can be done using the python function **cv2.cvtColor(image,cv2.COLOR_BAYER_GR2BGR)** of OpenCV.
- After the conversion, we need to determine the camera parameters from the given *ReadCameraModel.py*.

  The ReadCameraModel() function will provide us with the camera parameters as well as the LUT(Look Up Table) which will be utilized to undistort the image. The camera parameter matrix derived is in form,

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Now, we use UndistortImage(img, LUT) function of the UndistortImage.py. This step compensates for any lens distortion we may have.

3 Feature Extraction

- To detect and map the same points in the consecutive images, we need to identify the feature points in both the frames and match them.
- Here, to detect the feature points we have utilized SIFT detector which gives out the keypoints and its descriptors.
- SIFT stands for Scale Invariant Feature Transform, which is a robust algorithm based on histogram of gradients. Here, Image content is transformed to local feature co-ordinates that are invariant to translation, rotation, scaling and other imaging parameters.
- After detection of the features from both the frames, we need to match the similar features using FLANN based Matcher.
- Once the features are matched then these points can be utilized to compute the Fundamental Matrix.

4 Computing the Fundamental Matrix

After the feature extraction, Fundamental matrix was computed by randomly selecting eight points from the points derived earlier. The Fundamental Matrix, denoted by F, is a 3×3 matrix with rank 2. It relates the corresponding set of points from two images. It can be said that it is the algebraic representation of epipolar geometry. We have the equation

$$x_1^T F x_2 = 0$$

where x_1 and x_2 are the image co-ordinates of the detected feature points from the two successive frames. The equation can be represented as,

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

It can be further can be simplified as ,

$$x_1x_2f_{11} + y_1x_2f_{21} + x_2f_{31} + x_1y_2f_{12} + y_1y_2f_{22} + y_2f_{32} + x_1f_{13} + y_1f_{23} + f_{33} = 0$$

Here, the above equation has nine unknowns in one equation. Simplifying the equation for m correspondences we get,

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y1 & x'_1 & y'_1 & 1 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ x_mx'_m & x_my'_m & x_m & y_mx'_m & y_my'_m & y_m & x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f12 \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

This system of equation can be answered by solving the linear least squares using Singular Value Decomposition. So, by taking 8 random correspondences we can estimate the Fundamental Matrix.

5 Eliminating Outliers

- After getting the eight points at random from the set of keypoints, we need to find the best inliers. For this, we obtain the distance for each point from the epipolar line.
- If the distance of the keypoint from the epipolar line is less than threshold value, we consider it to be a good match and account it be the inlier.
- If the distance is greater than that of the threshold set, we discard the keypoint. Here we have used threshold to be 0.01.

The above process is repeated for number of iterations (here, 50). Once we have all the inliers, we have the final fundamental matrix using these points.

6 Essential Matrix

Further, the essential matrix can be derived from the Fundamental matrix and the Camera Calibration Matrix. Essential Matrix determines the relative camera poses between the subsequent image frames.

- The essential matrix can be extracted using the equation given below, where K is the camera calibration matrix and F is the Fundamental Matrix.

$$E = K^T F K$$

- Once we compute the essential matrix, we calculate its Singular Value Decomposition, to compensate for the noise present in the K.
- To put the rank 2 constraint we change the value of S to (1,1,0). Hence SVD of E is given as,

$$E = USV^T$$

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

7 Extracting Corresponding Camera Poses

After getting the Essential Matrix, we need to determine the different camera poses that define the Camera center and angles from different frames. It relates the camera position with respect to the world frame giving out a 3×3 matrix consisting of rotation and translation yielding 6 DOF. The Camera Poses are estimated using the Essential Matrix in the form (R,C) where R and C are the Rotation and the Camera Centres.

- Inorder to determine the Camera Poses, we compute the Singular Value Decomposition of the E matrix using,
$$E = UDV^T$$
- The corresponding Rotation and Camera centres can be extracted using below mentioned equations,
$$C_1 = U(:,3) \; and \; R_1 = UWV^T$$
$$C_2 = -U(:,3) \; and \; R_2 = UWV^T$$
$$C_3 = U(:,3) \; and \; R_3 = UW^TV^T$$
$$C_4 = -U(:,3) \; and \; R_4 = UW^TV^T$$
where W is,
$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
- The corresponding four poses are of the form
$$(C_1, R_1), (C_2, R_2), (C_3, R_3), (C_4, R_4)$$
.
- Once we have the rotation matrices and camera centres, we estimate the camera poses using $P = KR[I_{3\times3} - C]$.
- To minimise the error in camera poses, the determinants of R and C were checked to be positive.
- Further the determination of correct pose from the four possible poses is to be done.

8 Triangulation

- In the previous section, we computed four possible camera poses for a pair of images using the essential matrix. Now, inorder to find the correct set of poses we triangulate the points.
- Triangulation assumes known relative camera position and orientation and known intrinsic parameters. Here, we apply Linear Triangulation method.

- To find the correct set of poses, we make sure that the point projected using the camera pose, must be in front of the camera. This is called "Cheriality Condition".
- Here we have 2 views with known camera intrinsic parameters and the corresponding pose.
- Hence to estimate the point in 3D, we compute a matrix A which is given by,

$$A = \begin{bmatrix} xM_3 - M_1 \\ yM_3 - M_2 \\ x'M'_3 - M'_1 \\ y'M'_3 - M'_2 \end{bmatrix}$$

  where M_i *and* $M'_i$are the i^{th} row of the extrinsic camera parameters at origin and the subsequent frame respectively
- Further, we compute the Singular Value Decomposition of the A matrix to get the point in 3D w.r.t the world frame.
- We impose the Cheirality condition satisfying $r_3(X - C) > 0$. Here r_3 is the z-axis of the camera rotation matrix.

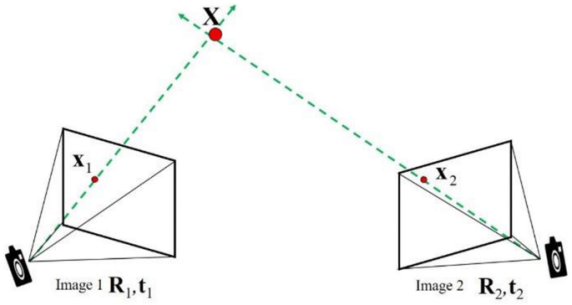

Figure 2: Triangulating 3D points

9 Plotting the Final Trajectory

To compute the final plot with respect to the first frame, we utilize the Homogeneous Transformation matrix from the first frame and multiply it with the homogeneous transform of the subsequent frames.
Homogeneous Transformation matrix is given by,

$$H = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$$

Using the above matrix, we plot the origin of every frame in order to generate the final visual odometry plot.

10 Comparison with the OpenCV Function

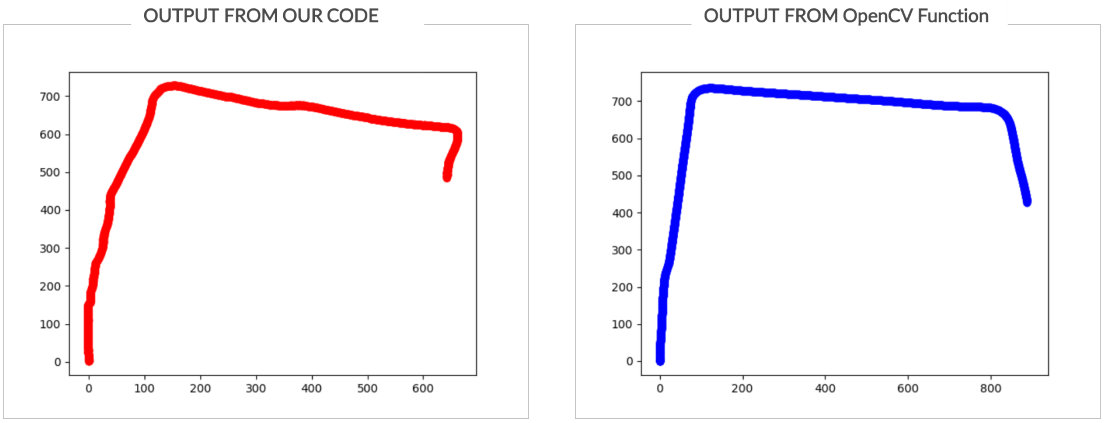

Figure 3: Output for the frame 1711

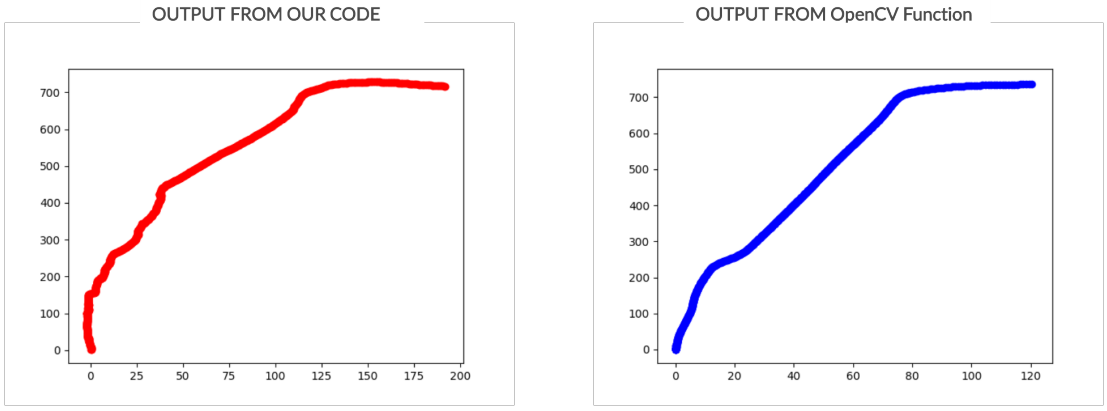

Figure 4: Output for the frame 883

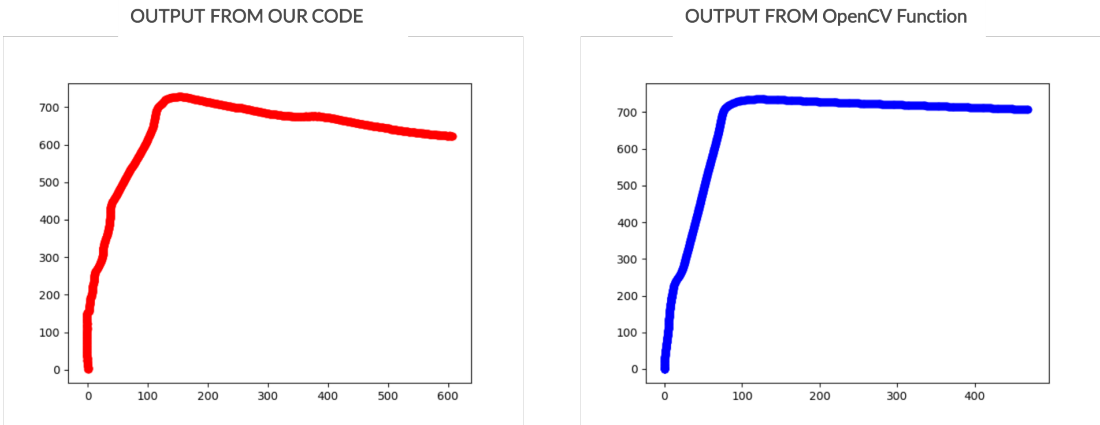

Figure 5: Output for the frame 1458

11 Challenges and Observation

- In the last leg of the video sequence, there was drift observed because of the non-uniform nature the car velocity. Hence, the linear triangulation gives fidgeting output which was corrected using a correction factor.
- The output can be made more robust by increasing the number of iterations used for RANSAC.
- Error in optimizing the functions to get the near correct values.
- Longer run times for the implementation.

12 References

- https://cmsc733.github.io/2019/proj/p3/
- Youtube Video on Structure for Motion By Prof. Mubarak Shah
- https://docs.opencv.org/master/da/de9/tutorial_py_epipolar_geometry.html
- https://www.youtube.com/watch?v=ggzJsYuCOeYt=774s