Project 5 Report
ENPM 673 - Perception for Autonomous Robots
Team 5

Prof. Dr. Mohammed Charifa

UNIVERSITY OF MARYLAND

Nikhil Lal Kolangara (116830768)
Kartik Venkat (116830751)
Kushagra Agrawal(116700191)

Contents

1 Introduction **3**

2 Pipeline **3**

2.1 Step 1: Data Preparation 3

2.2 Step 2: Extraction of Feature Points 3

2.3 Step 3: Calculation of Fundamental Matrix 3

2.4 Step 4: Essential Matrix 4

2.5 Step 5: Determining Camera Pose 4

2.6 Step 6: Triangulation 5

2.7 Step 7: Plotting of points 5

3 Comparison with Predefined Functions **6**

4 Additional Steps in the Pipeline **7**

4.1 Non-linear Triangulation 7

4.2 Non-linear PnP 7

5 Output **8**

6 References **10**

1 Introduction

Visual Odometry can be defined as the process of determining the Robot's position and orientation by analyzing the associated camera frames. The concept of Visual Odometry is analogous to SLAM technique. In the given project, a set of camera frames from a camera planted on a car and the Camera Calibration matrix is provided. We then perform the necessary operations to output the trajectory plot of the Camera from these image frames. This process involves plotting of feature points on the camera frames and determining their variation for every consecutive pair of frames to estimate the trajectory of camera pose.

2 Pipeline

2.1 Step 1: Data Preparation

The images were converted to BGR colorspace, then the Camera parameters were extracted from the given python functions followed by undistortion of images.

1. Data for this project is a set of 3873 Bayer format images, captured in sequence from a camera placed on a car facing in the forward direction.

2. Firstly these images were converted into BGR format by using cv2 function cv2.cvtColor(image,cv2.COLOR BayerGR2BGR)

3. After conversion to BGR format, the Camera Parameters were extracted from the given ReadCameraModel.py where the function ReadCameraModel extracts camera parameters and LUT matrix used for undistorting the images taken. The Camera matrix is derived as follows:

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

4. After deterining the Camera Matrix, Undistortion is done by implementing the function UndistortImage(image,LUT), from the given python script file UndistortImage.py.

2.2 Step 2: Extraction of Feature Points

1. Firstly, for detecting the feature points, SIFT Detector is used.

2. After detection of feature points for two consecutive frames, FLANN based Matcher was used to match the same features between two frames.

3. Once feature points detection and matching was calculated, these points were computed for the calculation of the the Fundamental matrix.

2.3 Step 3: Calculation of Fundamental Matrix

1. The fundamental matrix being a 3x3 matrix has nine components. It has seven degrees of freedom(three rotations,three translations and one camera parameter between successive frames).

2. By considering the epipolar constraint equation of the Fundamental Matrix, we have:

$$x_1^T F x_2 = 0$$

where x1 and x2 are the image coordinates of the detected feature point from two successive frames.

3. This equation can also be written as:

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

4. The above equation can be simplified as:

$x_1 x_2 f_{11} + y_1 x_2 f_{21} + x_2 f_{31} + x_1 y_2 f_{12} + y_1 y_2 f_{22} + y2 f_{32} + x_1 f_{13} + y_1 f_{23} + f_{33}$

5. The above equations has 9 unknowns in one equation. So, by taking 8 random correspondences from consecutive image frames, the components of the Fundamental matrix are determined as follows:

$$\begin{bmatrix} x_1 x_1' & x_1 y_1' & x_1 & y_1 y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_m x_m' & x_m y_m' & x_m & y_m y_m' & y_m & x_m' & y_m' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

6. The fundamental matrix can be estimated by solving the linear least squares Ax=0.

2.4 Step 4: Essential Matrix

1. Essential Matrix is calculated as:

$$E = K^T F K$$

where F is the fundamental matrix and K is the Camera Calibration matrix.

2. Once the Essential matrix is calculated from the above equation, its Singular value Decomposition is calculated and S matrix is changed in the following way:

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

3. To eliminate the noise in Camera calibration, the last value of the diagonal matrix is set to zero from Singular value decomposition and a final Essential matrix is computed.

2.5 Step 5: Determining Camera Pose

1. The Essential matrix gives four camera pose configurations, each of these matrices showing the possible camera positions and angles.

2. These four possibilities of the Camera pose Matrices were denoted as (C1, R1),(C2, R2),(C3, R3)and(C4, R4) where C is the camera centre and R is the Rotation matrix.

3. In order to determine these four possibilities of Camera pose, the Singular value decomposition of Essential matrix is computed firstly.

$$E = UDV^T$$

4. After computing the above mentioned Singular value decomposition for the Essential matrix, and denoting a matrix W , solve for the following equations:

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

5. Then the camera pose is calculated whose formula is:

(a) $C_1 = U(:,3) and R_1 = UWV^T$
(b) $C_2 = -U(:,3) and R_2 = UWV^T$
(c) $C_3 = U(:,3) and R_3 = UW^TV^T$
(d) $C_4 = -U(:,3) and R_4 = UW^TV^T$

6. From the above computations, the four possible permutations of the camera poses is obtained. To further eliminate the error in camera poses, we implemented the that the determinant of R and C should be positive (if det(R) = 1, correct it to C = C and R = R).

2.6 Step 6: Triangulation

To find the correct camera pose, it is checked by using cheirality condition i.e. the point projected using the camera pose must be in front of the camera . So to check the cheirality condition we triangulate the 3D points using linear least squares and check for the depth of those 3D points i.e. So the combination of (is the z-axis of the camera. r3 can be obtained from third row of the rotation matrix. So the pose (R,C) which produces maximum number of points satisfying cheirality condition.

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

2.7 Step 7: Plotting of points

To compute the plot with respect to first frame Homogeneous transformation is calculated and multiplied with previous computed Homogeneous matrix.

$$P = KR[I_{3*3} - C]$$

Using this homogenous transformation, plot the origin of every frame in order to generate the visual odometry plot.

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

3 Comparison with Predefined Functions

The difference in the output in the predefined function and the code we made. The accumulated drift, comparing the user defined function and the opencv function can be calculated as:

$$\sum_{j=1,2}\left(u^j - \frac{P_1^{jT}\widetilde{X}}{P_3^{jT}X}\right)^2 + \left(v^j - \frac{P_2^{jT}\widetilde{X}}{P_3^{jT}X}\right)^2$$

here (xo, yo) is the point plotted using predefined function, (xc, yc) is the point plotted using the user's code and j in the respective frame.

1. Increase the number of iteration of RANSAC to compute optimal value of Fundamental matrix.

2. Instead of randomly selecting correspondences we can use a robust algorithm i.e. Zhang's 8-point algorithm.

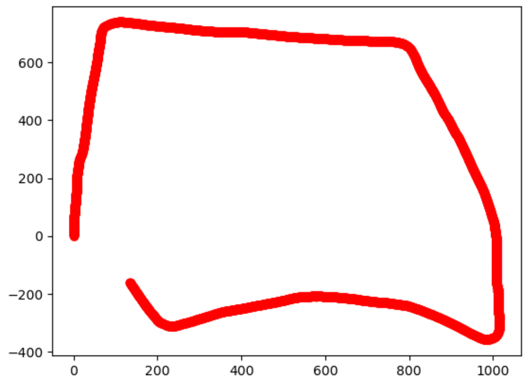


Figure 1: Output using our code

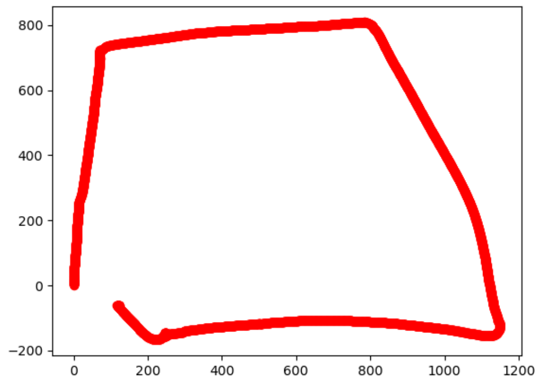


Figure 2: Output using predefined function

4 Additional Steps in the Pipeline

4.1 Non-linear Triangulation

By using the camera poses and 3D triangulated point that was calculated linearly, the error in reprojection of 3D points can be reduced using non-linear triangulation. The linear triangulation only minimizes the algebric error, This method minimizes the geometrical error given by

$$\sum_{j=1,2}\left(u^j - \frac{P_1^{jT}\widetilde{X}}{P_3^{jT}X}\right)^2 + \left(v^j - \frac{P_2^{jT}\widetilde{X}}{P_3^{jT}X}\right)^2$$

Where X˜ is the homographic representation of point X and P_i^T is the each row of camera projection matrix P.

To minimize this error we used the scipy's optimization function i.e. scipy.optimize.least squares() and obtain the 3D triangulated points.

This has not been implemented but if implemented will yield in better results.

4.2 Non-linear PnP

By using the camera poses and 3D triangulated point that was calculated linearly the reprojection error can be reduced. The linear PnP minimizes algebraic error. Reprojection error i.e. geometrical error is minimized by the following equation:

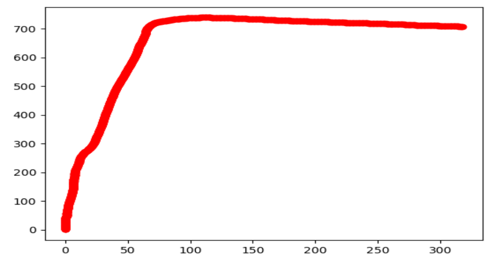
$$\min_{C,R}\sum_{i=1,J}\left(u^j - \frac{P_1^{jT}\widetilde{X_j}}{P_3^{jT}\widetilde{X_j}}\right)^2 + \left(v^j - \frac{P_2^{jT}\widetilde{X_j}}{P_3^{jT}X_j}\right)^2$$

Where X˜ is the homographic representation of point X and P_i^T is the each row of camera projection matrix P , which is computed by $P = KR[I_{3\times3}C]$.

To minimize this error scipy's optimization function i.e. scipy.optimize.least squares() is used.

This has not been implemented, but if implemented will yield better results.

5 Output


(a) Camera Pose using our code

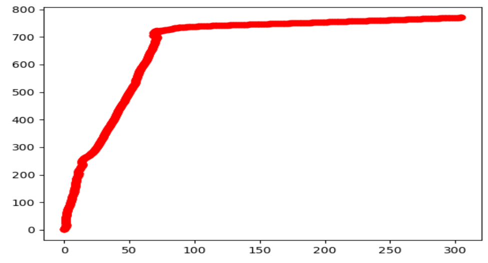
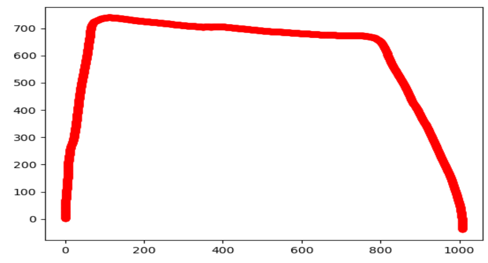

(b) Camera Pose using inbuilt function

Figure 3: Camera pose for frame 1026


(a) Camera Pose using our code

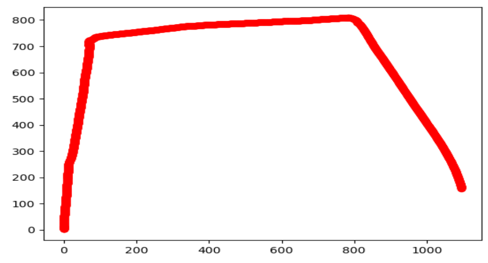
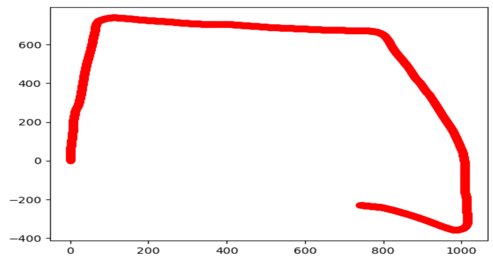

(b) Camera Pose using inbuilt function

Figure 4: Camera pose for frame 2273


(a) Camera Pose using our code

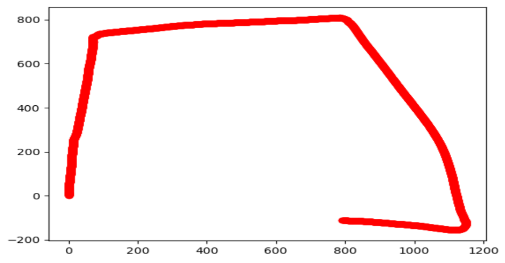
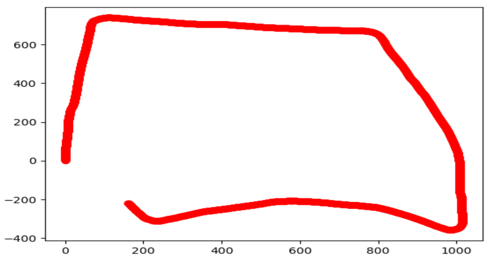
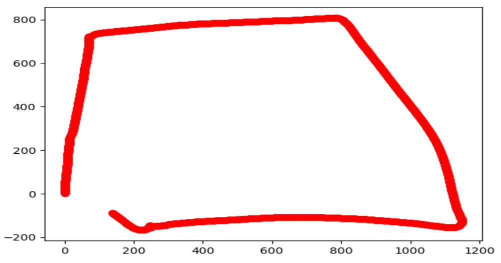

(b) Camera Pose using inbuilt function

Figure 5: Camera pose for frame 2960

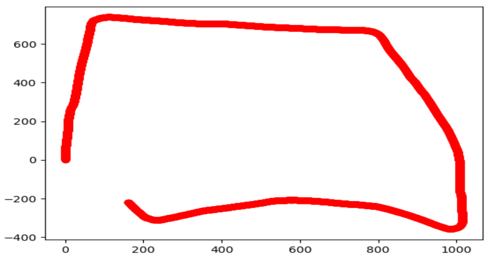


(a) Camera Pose using our code

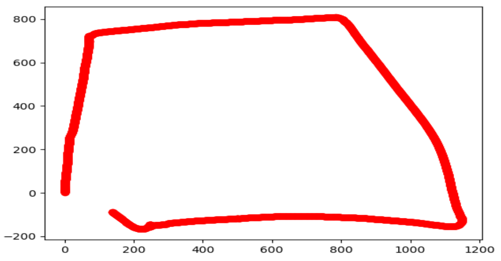


(b) Camera Pose using inbuilt function

Figure 6: Camera pose for frame 3785



(a) Camera Pose using our code



(b) Camera Pose using inbuilt function

Figure 7: Final Output

`https://drive.google.com/drive/folders/17OhooFbGob1AK6oTMPOyEKYiSf5e7nNW?usp=sharing`

6 References

1. Coursework material

2. https://cmsc733.github.io/2019/proj/p3/

3. https://web.stanford.edu/class/cs231a/course$_{n}otes/04 - stereo - systems.pdf$

4. http://www.cs.cmu.edu/ 16385/s17/Slides/11.1$_{C}amera_{m}atrix.pdf$