Perception For Autonomous Robots

Project 5

# VISUAL ODOMETRY

Raghav Nandwani (116321549) Sanket Goyal (116155144)

Varsha Eranki (116204569)

# Contents

# List of Figures

# 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 concepts of Visual Odometry are analogous to SLAM technique.

In the given project, we were given a set of camera frames from a camera planted on a car(in this context, analogous to a robot) 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 The Pipeline

## 2.1 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 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, we extracted the Camera Parameters from the given *ReadCameraModel.py* where the function *ReadCameraModel('path_to_model')* 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 Extraction of Feature points

The next step involves feature points matching using SIFT and FLANN based feature matching.

1. To detect and map the same points in the consecutive image frames for computing the Fundamental matrix, we need to firstly identify feature points in the two images and then match them.

2. Firstly, for detecting the feature points, we used *SIFT Detector* that gave us seven thousand feature points approximately for every frame.

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

4. Once feature points detection and matching was calculated, these points were computed for the next step of calculating the Fundamental matrix.

## 2.3 Computation of Fundamental Matrix using RANSAC

After extraction of these correspondences, the Fundamental matrix was computed by randomly selecting 8 points and computing the Fundamental matrix from those points and checked with all the correspondences for the inliers. The Fundamental Matrix is a 3x3 matrix which captures the relationship between points in two frames. We computed this as follows:

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 $x_1$ and $x_2$ are the image coordinates of the detected feature point from two successive frames.

3. This equation can 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:

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

5. The above equations has 9 unknowns in one equation. So, by taking 8 random correspondences from

consecutive image frames, we determined the components of the Fundamental matrix as follows:

$$\begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx_m' & x_my_m' & x_m & y_mx_m' & y_my_m' & y_m & x_m' & y_m' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

6. The above equation was then computed by obtaining the Eigen vector of the smallest Eigen value of the above Coordinate matrix , i.e. the last column of V matrix in Singular Value decomposition of the mentioned matrix.Thus, we determined the Fundamental matrix from above calculations.

7. To eliminate the noise in the point correspondences between image frames and to implement the rank 2 constraint of the above Fundamental matrix (3X3 matrix), we then further computed its Singular value decomposition and enforce the smallest Eigen value to be zero.

8. Because of the random correspondences we do this for some number of iterations and reject the outlier using $x_1^T F x_2 = 0$ and choose the fundamental matrix with most number of inliers.

9. It can also be further defined as, the epipolar line in one camera is the product of the Fundamental matrix and a point from the other camera.

Then this was repeated for 50 iterations and the fundamental matrix was selected on the basis of the most number of inliers.

## 2.4 Essential Matrix

The Essential matrix was determined from the Fundamental matrix and Camera calibration matrix. The essential matrix determines the relative camera poses between the image frames which is computed from the Camera calibration matrix and Fundamental matrix. The Essential matrix has 5 degrees of freedom.

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 was calculated from the above equation, its Singular value Decomposition

was calculated and S matrix is changed in the following way.

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

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

## 2.5 Determining Camera Pose for Frames

Then, we computed the camera poses in the form of (R,C) where R and C denote the Rotation and camera centre between two successive frames by solving for the most number of points that lie before the computed Camera pose from frame to frame. The Camera pose matrix defines the Camera position with respect to the world frame and thus, consists of three translation and three rotation vectors showing the six degrees of freedom. The camera pose is determined from the Essential matrix.

1. The Essential matrix gave us 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 $(C_1, R_1), (C_2, R_2), (C_3, R_3) and (C_4, R_4)$ 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 $W$ we solve for the following equations:

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

and the formulations of the possible Camera centres and Rotation matrices were computed from:

(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) $C_4 = -U(:,3) and R_4 = UW^TV^T$

5. Then it was computed for Camera pose whose equation is (where K denotes the camera calibration matrix):

$$P = KR[I_{3x3} - C]$$

6. From the above computations, we got the four possible permutations of the camera poses. 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 $R = -R$).

7. Linear Triagulation and Check Chlarity condition has been further used to calculate the correct R and C, which is explained in the further topics..

### 2.6 Plotting the points

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

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

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

$$p_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad , \quad p_1 = H \cdot p_0$$

## 3 Comparison with Predefined Functions

We can observe the difference in the output in the predefined function and the code we made. We can do the following to reduce the error

The accumulated drift, comparing the user defined function and the opencv function is: 1120880.096. It is calculated as

$$d = \sum_{j=1,2,....n} \sqrt{\left| \left(x_o^j\right)^2 - \left(x_c^j\right)^2 \right| + \left| \left(y_o^j\right)^2 - \left(y_c^j\right)^2 \right|}$$

here $(x_o, y_o)$ is the point plotted using predefined function, $(x_c, y_c)$ 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.

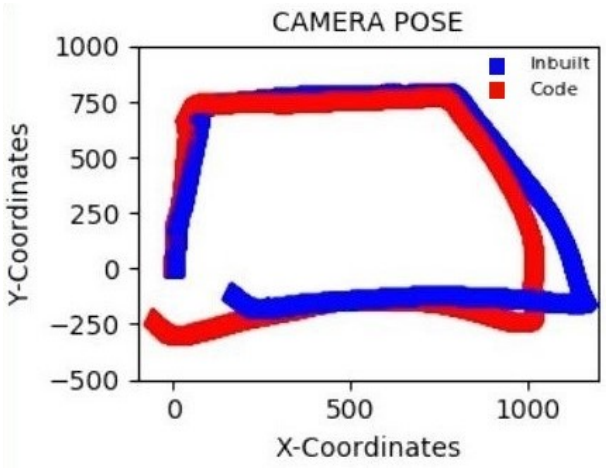


Figure 1: Comparison with inbuilt functions

# 4 Triangulation

After obtaining 4 possible camera poses using essential matrix to estimate the correct camera pose we triangulated 3D points.

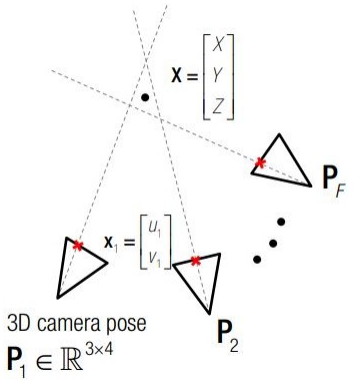


Figure 2: Triangulation of 3D points

## 4.1 Triangulate using linear least squares

To find the correct camera pose we check it 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. $r_3$ can be obtained from third row of the rotation matrix. So the pose (R,C) which produces maximum number of points satisfying cheirality condition.

### 4.2 Non-Linear Triangulation

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

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

Where $\tilde{X}$ 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.

## 5 Perspective-n-Points

It is the method of estimating the camera pose (6-DOF) by using set of 3D projected points, their projection on the image plane and the intrinsic parameter of the camera

### 5.1 Linear Camera Pose Estimation

Given the 2D correspondences with their respective 3D projection and the intrinsic parameters. Correspondences is normalized by intrinsic parameter i.e. $K^{-1}x$ in order to establish linear least square relation between 3D and 2D points to get value (t,R) i.e. $t = -R^T C$.

### 5.2 Non-linear PnP

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

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

Where $\tilde{X}$ is the homographic representation of point X and $P_i^T$ is the each row of camera projection matrix P , which is computed by $P = KR[I_{3X3} \quad -C]$. To minimize this error we used the scipy's optimization function i.e. *scipy.optimize.least_squares()*.

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

# 6 Results and Analysis

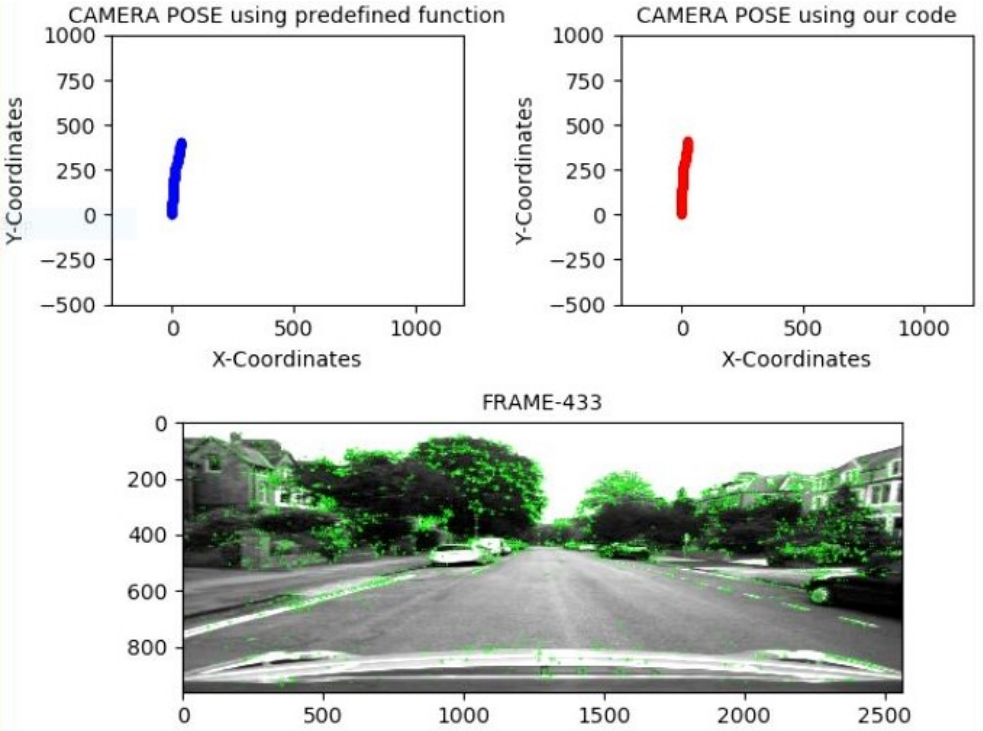


Figure 3: Output - Frame 433

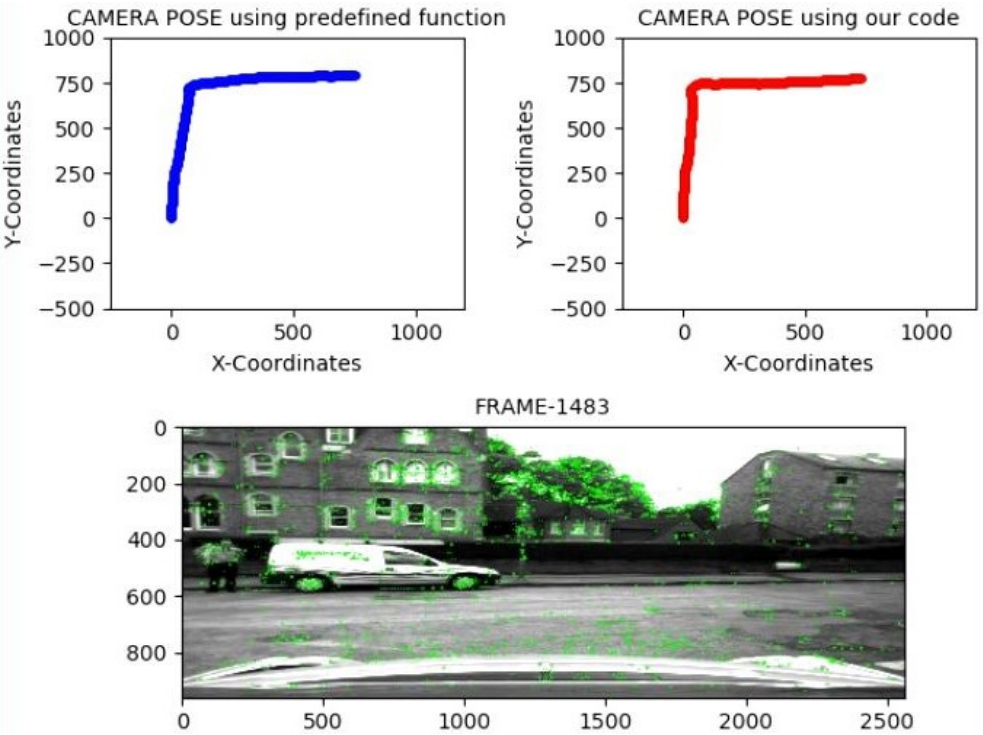


Figure 4: Output - Frame 1483

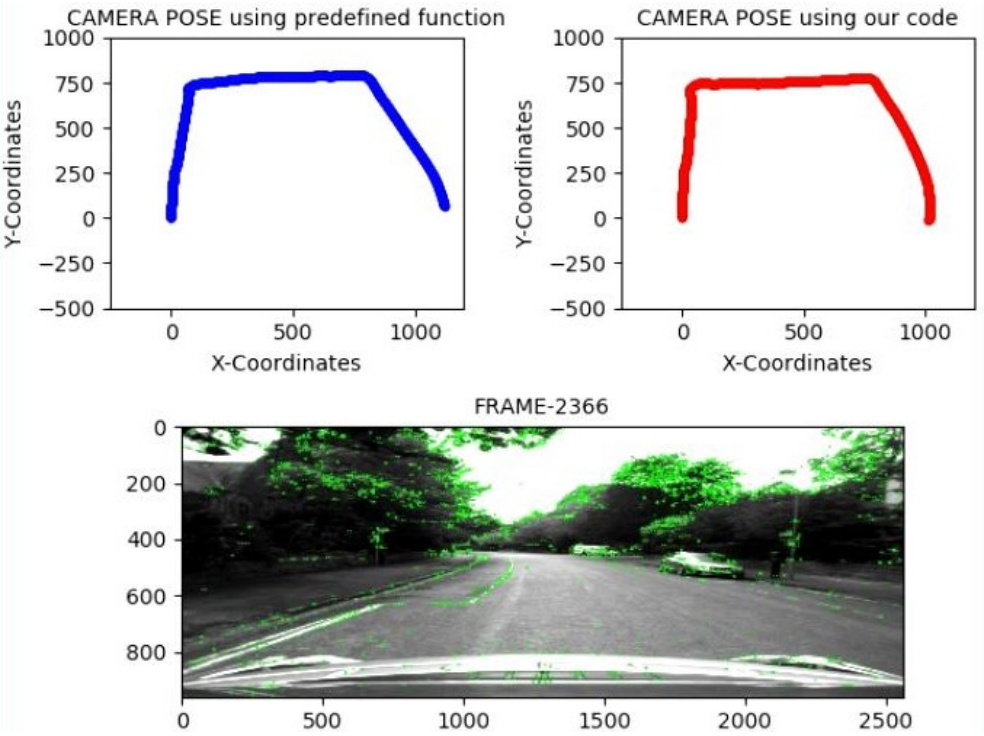


Figure 5: Output - Frame 2366

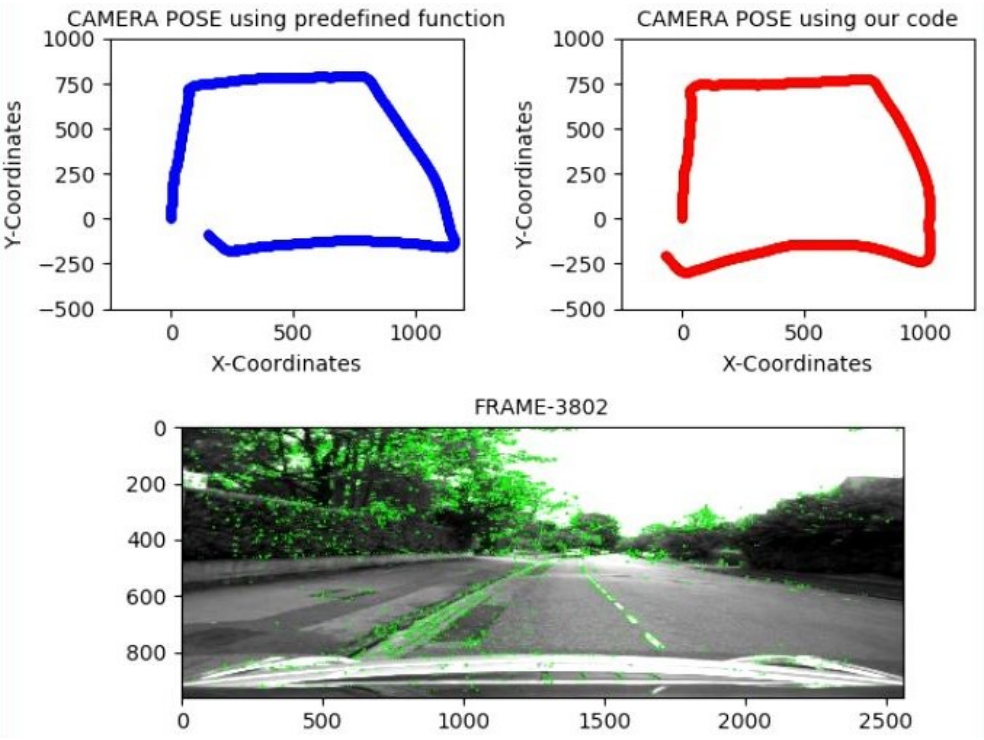


Figure 6: Output - Frame 3802

# 7 References


1. http://cis.upenn.edu/ cis580/Spring2015/Lectures/cis580-17-SfM_lecture.pdf
2. https://docs.opencv.org/3.4/dc/dc3/tutorial_py_matcher.html
3. Coursework material
4. https://www.youtube.com/watch?v=K-j704F6F7Q
5. https://cmsc733.github.io/2019/proj/p3/