Perception For Autonomous Robots

Project 5

VISUAL ODOMETRY

Nikhil Mehra (116189941) Pranali Desai (116182935)

Sayan Brahma (116309165)

Contents

List of Figures

1 Introduction

Visual Odometry is a method of finding a robot/camera pose i.e translation and orientation of the robot/camera with respect to the the world frame using camera data. The study of sequential images and its corresponding points helps in calculating the displacement of the robot/camera.

The consecutive pairs of the images were used to estimate the camera pose attached to the car in the data set. The set of images with the camera calibration matrix are the given parameters for this project and the output is the trajectory plot of the car/camera in the x and z axis. The output is also compared with output using pre-defined Opencv functions.

The basic concepts used for Visual Odometry can be compared to the results of SLAM which is an important topic in Image Processing and Perception.

2 Preparing the Data

1. In this project, the dataset has a set of 3873 Bayer format images, taken in sequence from a camera facing the path of a car moving forward and taking turns.
2. For data preparation, these images were first converted from Bayer format to BGR format by using cv2 function, *cv2.cvtColor(image, cv2.COLOR_BayerGR2BGR)*
3. The file *ReadCameraModel.py* is provided with a function *ReadCameraModel('path_to_model')* to extract camera parameters and LUT matrix for undistorting the image.

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

4. The function *UndistortImage(image,LUT)*, from file *UndistortImage.py* was used to finally have the undistorted image

3 Determination of Feature Points

1. For computing the Fundamental matrix, we needed eight same feature points from two adjacent frames. These points were then formulated to solve for the Fundamental matrix.
2. In a given frame, there were thousands of feature points that can be identified. To capture such feature points, the function from OpenCV, SIFT detector was implemented.
3. To match these feature points, another function called the FLANN based matcher was implemented. The FLANN based matcher matches the similar feature points that were determined using SIFT detector.

4. Thus, these set of feature points between two frames were used for calculating the Fundamental matrix mentioned below.

4 Computing Fundamental Matrix via RANSAC

1. The Fundamental matrix describes relationship between the correspondences in the stereo images.

2. It is a 3x3 matrix with 9 components that can be defined by the following equation where x_1 and x_2 are the coordinates of a feature point in two images:

$$x_1^T F x_2 = 0$$

3. The above equation was derived by implementing the epipolar geometry constraint equation. The Fundamental matrix denoted by F has 9 components, i.e 9 unknowns but can be defined as one equation as follows:

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

The above equation when simplified further to one equation:

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

4. So instead of one feature point, we take 8 feature point coordinates to calculate the Fundamental matrix.

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

5. To solve the above equation, we computed the SVD of the Coordinate matrix and the last singular value set to zero, in order to add the rank 2 constraint in the fundamental matrix. This determined the Fundamental matrix with one of its eigenvalue as zero.

6. Then rank of the Fundamental matrix was computed to 2 to eliminate noise conditions in point correspondences between the frames.

$$F = USV^T$$

Thus, we got the Fundamental matrix from the above equation by applying the mentioned condition.

5 Calculation of Essential Matrix

1. The Essential matrix, E, is also a 3x3 matrix which infers 5 degrees of freedom of the matrix parameters and motion.
2. It was computed using the Fundamental Matrix and Camera Calibration matrix that was already given to us.
3. The equation for the essential Matrix is:

$$E = K^T F K$$

where K is the Camera Calibration of the Camera.

4. To reduce the noise from frame to frame in point correspondences, we performed SVD on the Essential matrix and equated the eigen values to 1, 1 and 0. Thus, the Essential matrix was determined.

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

6 Camera Pose Estimation and Triangulation Check

1. After solving for the Essential matrix from the camera calibration matrix, we need to determine the camera poses that define camera centres and angles in the two frames.
2. The Camera calibration matrix shows 6 DOF.
3. To find the camera poses, we first found the Singular value decomposition given by:

$$E = UDV^T$$

4. After this, we solved for the camera poses from a vector W given by

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

and the four possible Camera poses were solved as follows for the four sets of camera centres(C) and rotation matrices(R) $\{(C_1, R_1), (C_2, R_2), (C_3, R_3)$ and (C_4, R_4) where $C)$

   (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. Once the four sets of Camera centres and Rotation matrices were computed, the poses were determined where camera pose of the current frame is $P_1 = [I_{3x3}|0]$ and camera pose of the next frame is $P_2 = [R|-C]$

6. To obtain the correct value, the determinants of Camera centres and rotation matrices were made sure that they were positive and otherwise were corrected to be positive.

7 Triangulation

With the help of the calculated essential matrix and the 4 solutions for the camera pose to obtain the correct camera pose we triangulated 3D points.

7.1 Linear Triangulation

The points are triangulated by solving the equation using SVD:

$$\begin{bmatrix} [x]_x P_1 \\ [x']_x P_2 \end{bmatrix} X = 0$$

To obtain optimal camera pose we used cheirality condition, which states that the point projected in the world frame using camera pose must all be in front of the camera. i.e. $r_3\,(X - C) > 0$. where r_3 is the z-axis of the camera rotation matrix. So the combination of (R,C) that produces maximum number of points in front of the camera is chosen to be the optimal values of R and C.

7.2 Non-Linear Triangulation

Through the linear triangulation we minimized the algebric error in the triangulation, to minimize the geometric error we can use the non-linear triangulation by using the obtained camera pose and the linearly triangulated 3D projections. The error equation is given by

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

Here $\tilde{X}$ is the homographic representation of point X and P_i^T is the each row of camera projection matrix P. We used the scipy's inbuilt function i.e. *scipy.optimize.least_squares()* to minimize this error.

8 Perspective-n-Points

After we calculated the projected 3D points and their respective correspondences we obtain the optimal camera pose using the PnP technique.

8.1 Linear PnP

By using the 3D projection points and their respective 2D correspondences in combination with the camera matrix.we obtained the solution between 3D and 2D points to obtain value of t and R. i.e. $t = -R^T C$.

8.2 Non-linear PnP

By using the 3D triangulated points and linearly calculated pose we try and minimise the error by using following equation

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

Here $\tilde{X}$ is the homographic representation of point X and P_i^T is the each row of camera projection matrix P.

9 Code Pipeline

1. The images from the dataset folder were first loaded and then converted from Bayer format to BGR format. The Camera calibration matrix was computed followed by undistorting of the images.

2. The feature points are extracted from consecutive frames to compute the Fundamental matrix, using OpenCV functions of SIFT Detector for detection and FLANN based Matcher for matching the detected feature points.

3. Then after extracting these feature point correspondences, the Fundamental matrix was computed by randomly selecting 8 feature points.

4. Also, after computing the Fundamental matrix from those points, a RANSAC check is used for getting the count of the inliers(corresponding feature points).The Fundamental matrix with the most number of inliers is our optimal fundamental matrix.

5. The above check was done for 50 iterations to get the optimal Fundamental matrix. The solution be can improve by increasing the number of iterations at an expense of more computation time.

6. With the final Fundamental matrix, we further computed the Essential matrix using the Camera calibration matrix.

7. Then the possible Camera poses were determined forming four sets of camera centres and rotation matrices.

8. We assumed that the + x axis points towards right, + y axis is coming out of the page and + z axis points downwards initially. The camera poses with extreme rotation in z-axis are then neglected manually as rotation about z-axis does not have any physical relevance. These gave us the possible values of the camera poses and were further singled out based on cheirality check criteria.

9. We then computed the position of the camera of the current frame with respect to the previous frame by calculating Homogeneous transformation and multiplying it with previously calculated Homogeneous matrix.

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

10. From the calculated homogeneous transformation matrix the center of the camera frame for every frame is found plotted.

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

11. Our output is finally matched with the output using Opencv pre-defined function such as cv2.ndEssentialMat and cv2.recoverPose.

10 Comparison with Predefined Functions

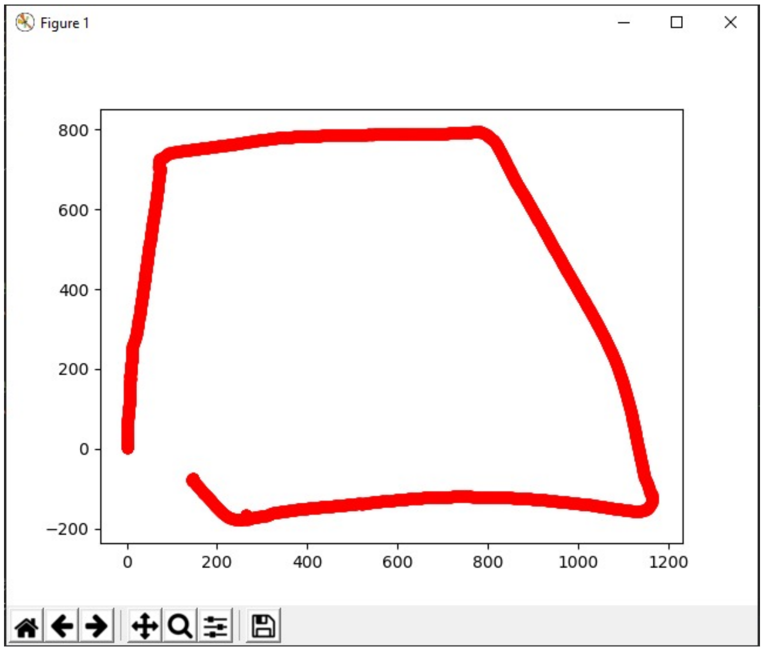


Figure 1: Output generated using pre-defined function

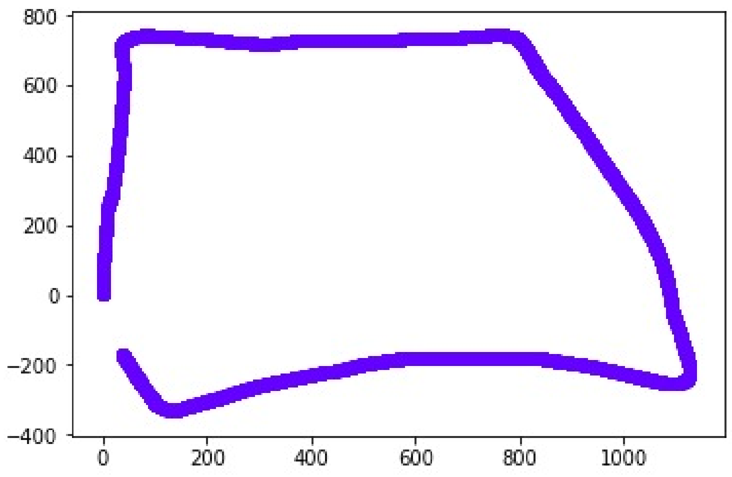


Figure 2: Output generated using the code

The accumulated drift from the predefined and user defined function is: 1114980.0958634. As we can see from the histogram that the drift per frame is very less till 2000 frames.

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

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.

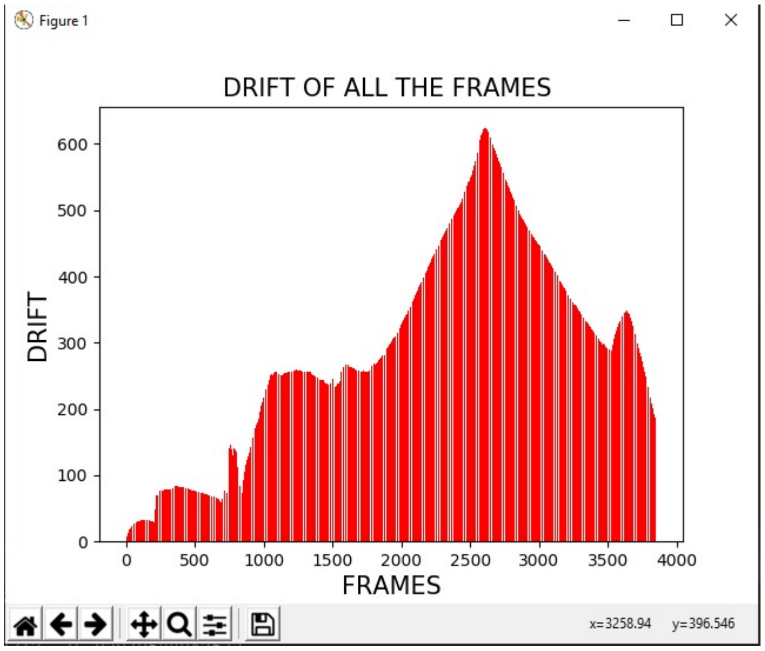


Figure 3: Drift Per Frame

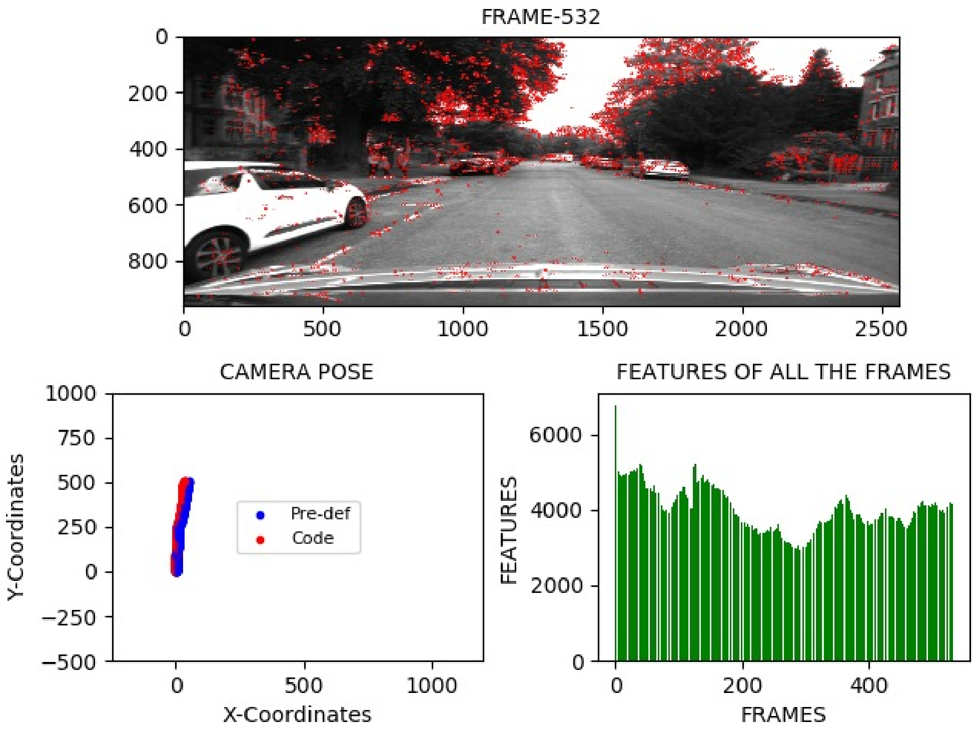
We can see the difference in the output in the Opencv function and the code we made. We can reduce the error by the following ways.

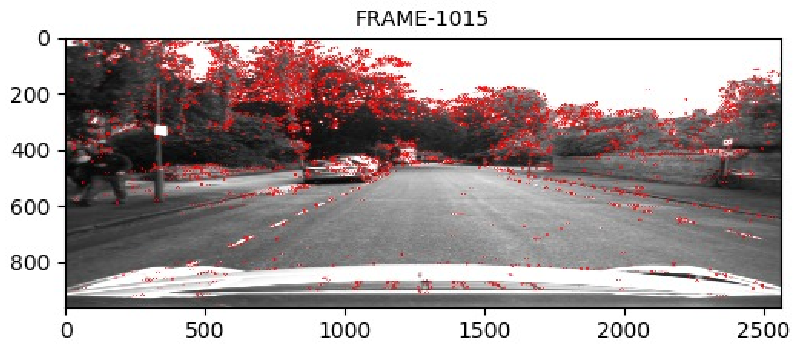
1. Increasing the number of iteration of RANSAC to compute Fundamental matrix.
2. We can use Zhang's 8-point algorithm, instead of randomly selecting correspondences between two images.
3. The image could be normalized to get everything with respect to optical center thus having better results.

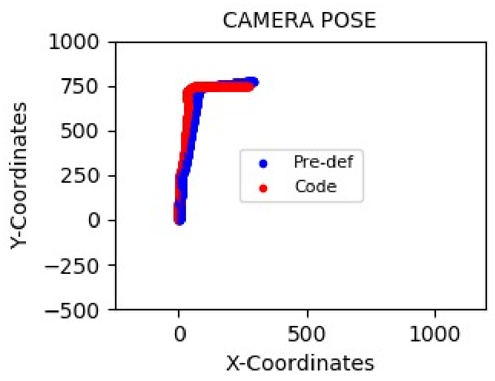
11 Output Link

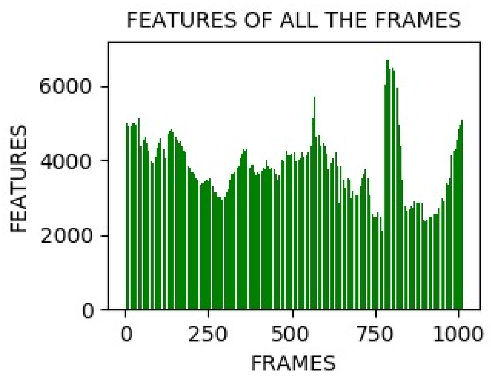
https://drive.google.com/file/d/1nyGLV_Q8ZoWQW-UUYyufVe5YsIr6On2x/view

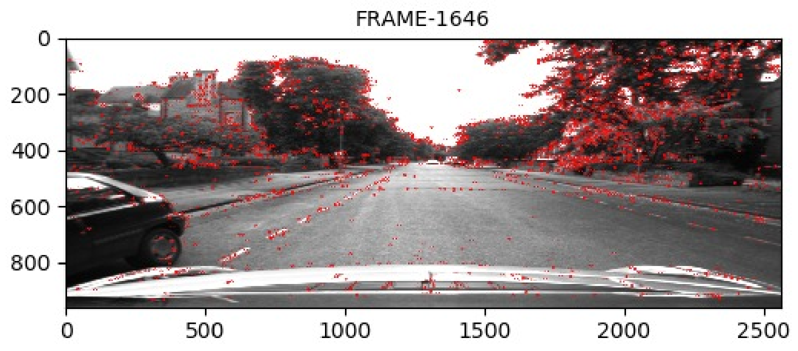
12 Results

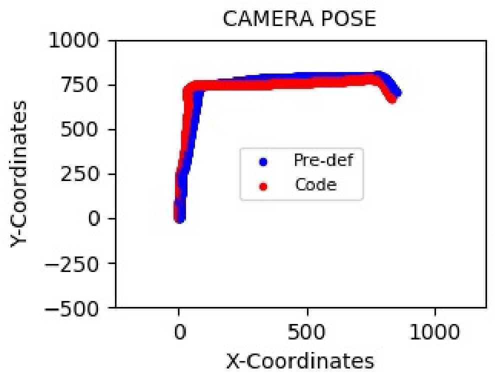


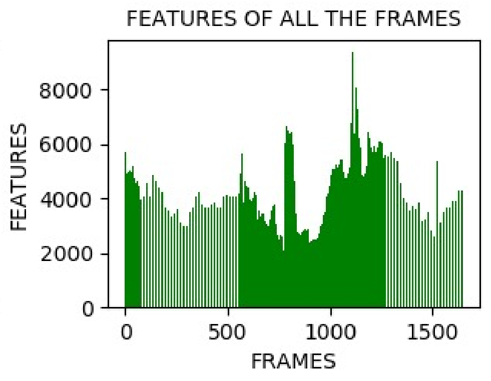

FRAME-1015

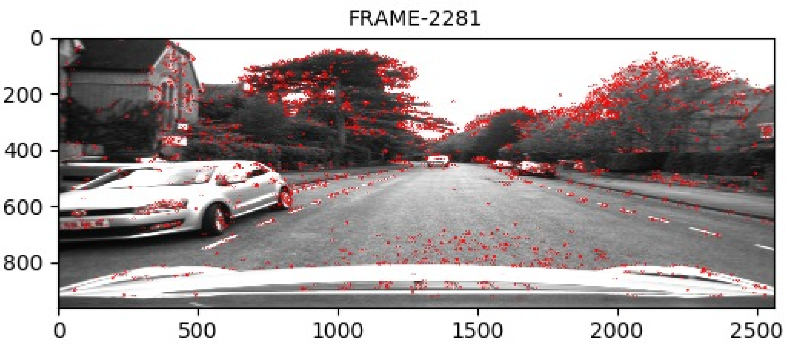

CAMERA POSE
Y-Coordinates
X-Coordinates
Pre-def
Code

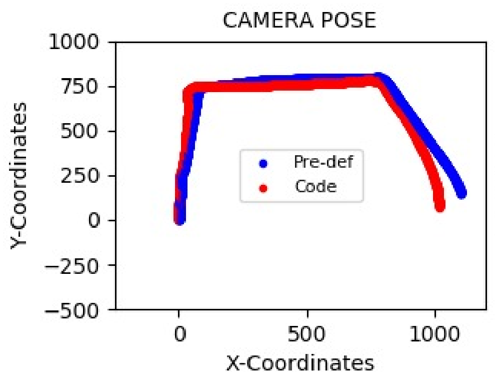

FEATURES OF ALL THE FRAMES
FEATURES
FRAMES

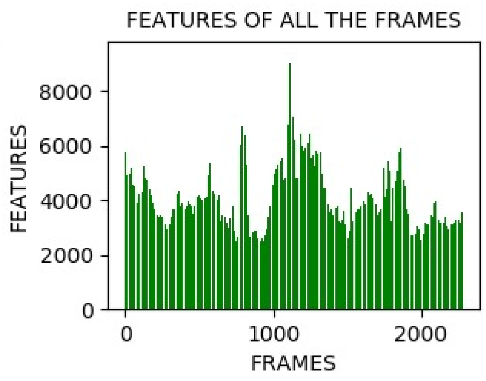

FRAME-1646

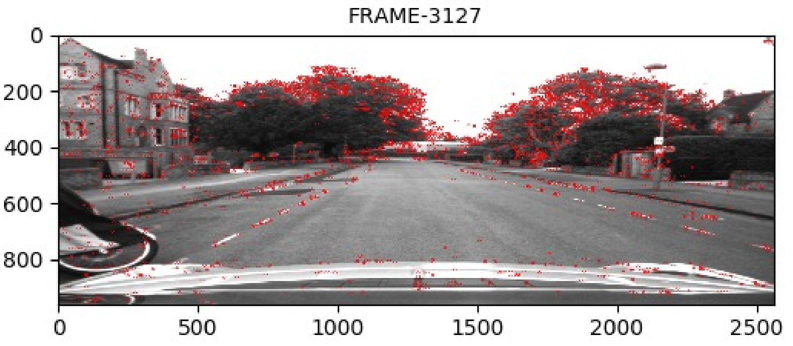

CAMERA POSE
Y-Coordinates
X-Coordinates
Pre-def
Code

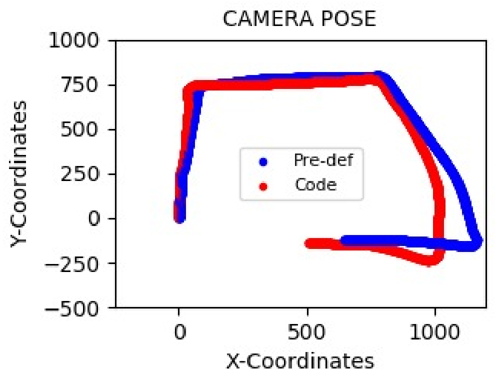

FEATURES OF ALL THE FRAMES
FEATURES
FRAMES

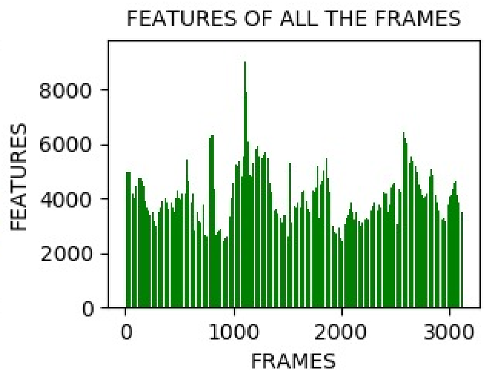

FRAME-2281
0
200
400
600
800
0
500
1000
1500
2000
2500

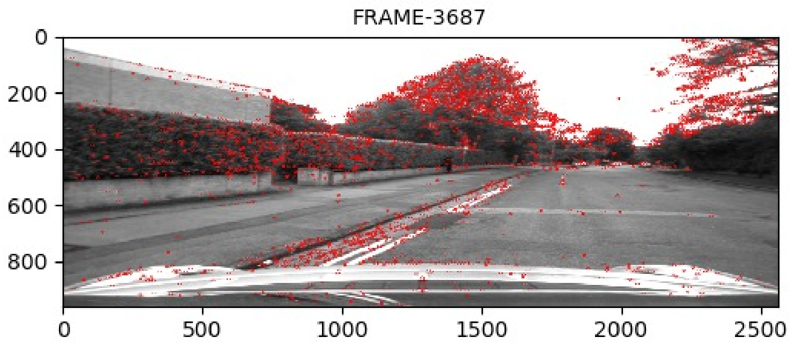

CAMERA POSE
Y-Coordinates
X-Coordinates
Pre-def
Code
1000
750
500
250
0
−250
−500
0
500
1000

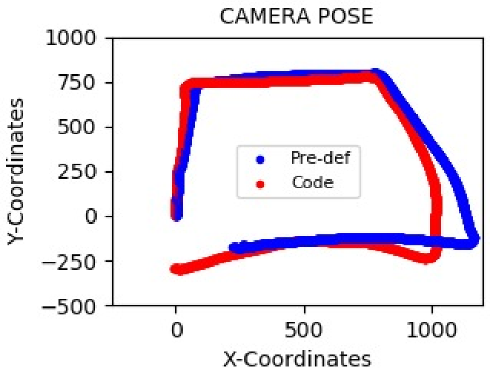

FEATURES OF ALL THE FRAMES
FEATURES
FRAMES
8000
6000
4000
2000
0
0
1000
2000

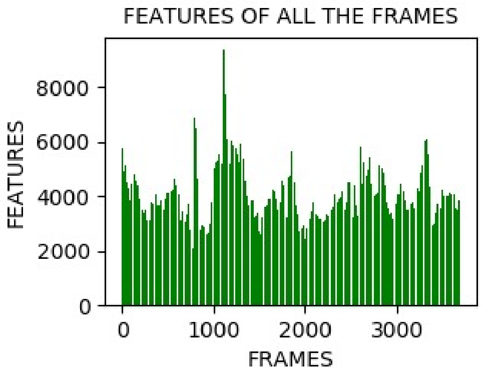

FRAME-3127
0
200
400
600
800
0
500
1000
1500
2000
2500


CAMERA POSE
Y-Coordinates
X-Coordinates
Pre-def
Code
1000
750
500
250
0
−250
−500
0
500
1000


FEATURES OF ALL THE FRAMES
FEATURES
FRAMES
8000
6000
4000
2000
0
0
1000
2000
3000


FRAME-3687
0
200
400
600
800
0
500
1000
1500
2000
2500


CAMERA POSE
Pre-def
Code
Y-Coordinates
X-Coordinates
1000
750
500
250
0
−250
−500
0
500
1000


FEATURES OF ALL THE FRAMES
FEATURES
FRAMES
8000
6000
4000
2000
0
0
1000
2000
3000

13 References

1. Lecture notes
2. https://cmsc733.github.io/2019/proj/p3/
3. https://web.stanford.edu/class/cs231a/course_notes/03-epipolar-geometry.pdf
4. https://www.uio.no/studier/emner/matnat/its/UNIK4690/v16/forelesninger/lecture_7_2-triangulation.pdf
5. https://www.youtube.com/watch?v=K-j704F6F7Q
6. https://docs.opencv.org/3.1.0/da/de9/tutorial_py_epipolar_geometry.html