University Of Maryland

College Park

Project Report

673-Perception for Autonomous Robots

Project 5

Aditya Goswami
116951968

Mahmoud Dahmani
116777896

Sukoon Sarin
116955522

May 4, 2020

Contents

List of Figures

1 Introduction

Visual Odometry is a technique used to find a robot/camera pose i.e translation and orientation of the camera on the vehicle/robot with respect to the the world frame using camera data. The study of the corresponding points of the sequential images helps in calculating the displacement of the camera on the vehicle/robot. In this project we are given frames of a driving sequence taken by a camera in a car, and the scripts to extract the intrinsic parameters. We implemented the different steps to estimate the 3D motion of the camera, and provide as output a plot of the trajectory of the camera. The output is also compared with output using pre-defined Opencv functions.

Visual Odometry is a very important concept in computer vision for estimating the trajectory of the vehicle/Robot (the camera on the vehicle/robot to be precise).Visual Odometry is very similar to SLAM, which is an integral part of Perception and localization in Robotics.

- 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.

1.1 Basic Pipeline

- Feature Matching and Outlier rejection using RANSAC
- Estimating Fundamental Matrix
- Estimating Essential Matrix from Fundamental Matrix
- Estimate Camera Pose from Essential Matrix
- Check for Cheirality Condition using Triangulation
- Further below is the detailed report of our steps for each problem.

2 Data Preparation

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 *UndistortImage.py* was used to finally have the Undistorted Images.

(a) Frame 1

(b) Frame 2

(c) Frame 3

Figure 1: Undistorted frames from processed dataset

3 Determination of feature points and matching

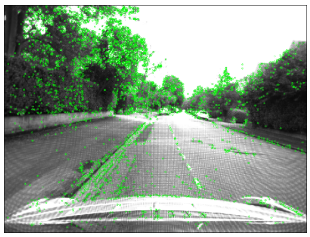
1. For computing the Fundamental matrix, we needed eight same feature points from two adjacent frames. These points were then formulated to solve for the Fundamental matrix.

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.

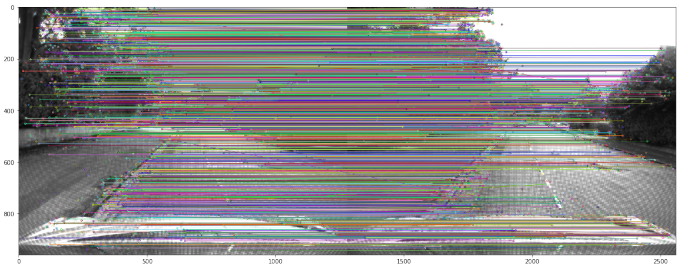
4. SIFT stands for Scale Invariant Feature Transform, which is a robust algorithm based on histogram of gradients. Here, Image content is transformed to local feature co-ordinates that are invariant to translation, rotation, scaling and other imaging parameters.

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



(a) Keypoints in frame 1

(b) Corresponding Keypoints in frame 2



(c) Best keypoints Match

Figure 2: Undistorted frames from processed dataset

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 x1 and x2 are the coordinates of a feature point in two images:

$$x_1^T F x_2 = 0$$

where x_1 and x_2 are the image co-ordinates of the detected feature points from the two successive frames.

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:

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

It can further be simplified as ,

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

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

$$\begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y1 & x_1' & y_1' & 1 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ x_mx_m' & x_my_m' & x_m & y_mx_m' & y_my_m' & y_m & x_m' & y_m' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f12 \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

5. Unlike homography, in F matrix estimation, each point only contributes one constraints as the epipolar constraint is a scalar equation. Thus, we require at least 8 points to solve the homogenous system. That is why it is known as Eight-point algorithm.

6. 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.

7. 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.

4.1 RANSAC

- 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.

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 and F is final fundamental matrix.

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 reducing to rank 2 constraint, the Essential matrix was determined.

$$E = USV^T$$

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

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:

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

5. The corresponding Rotation and Camera centres can be extracted using below mentioned equations,

$$C_1 = U(:,3) \; and \; R_1 = UWV^T$$

$$C_2 = -U(:,3) \; and \; R_2 = UWV^T$$

$$C_3 = U(:,3) \; and \; R_3 = UW^TV^T$$

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

and the four possible Camera poses were solved as follows for the four sets of camera centres *(C)* and rotation matrices *(R)*:

$$(C_1, R_1), (C_2, R_2), (C_3, R_3), (C_4, R_4)$$

6. Once we have the rotation matrices and camera centres, we estimate the camera poses using $P = KR[I_{3\times3} - C]$.

7. 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

7.1 Linear 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.
- Triangulation assumes known relative camera position and orientation and known intrinsic parameters. Here, we apply Linear Triangulation method.

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

- 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.

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

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

Here $\bar{X}$ 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

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

8.1 Non-Linear PnP

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

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

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

9 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.

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

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.

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

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

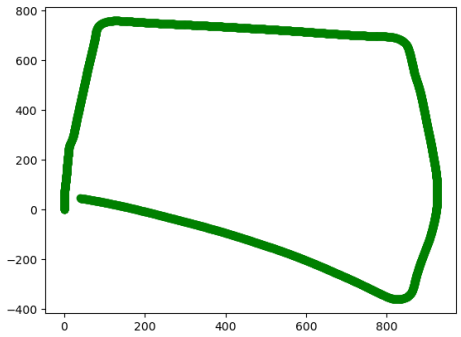


Figure 3: Output generated using pre-defined OpenCV functions

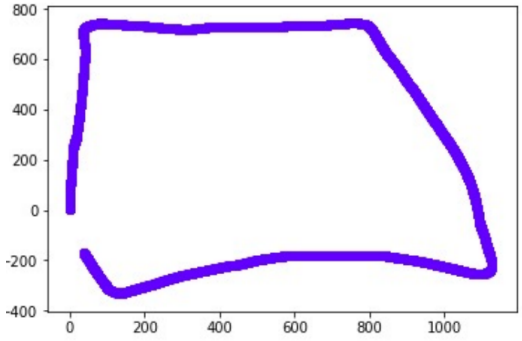


Figure 4: Output generated using self-defined functions

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

$$d = \sum_{j=1,2,\ldots,n} \sqrt{|\left(x_0^j\right)^2 - \left(x_c^j\right)^2| + |\left(x_0^j\right)^2 - \left(x_c^j\right)^2|}$$

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.
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.

Finally We had a great help from all these resources:

- Lecture notes and pdf.[1]
- Computer Vision, A Modern Approach [2]
- Buildings built in minutes - An SfM Approach -an article is written by Chahat Deep Singh [3]

11 Perspective-n-Points

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

11.1 Non-Linear PnP

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

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

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

References

[1] Prof. Charifa. *ENPM-673.*

- Lecture notes and pdf.

.

[2] Forsyth and Ponce. *Computer Vision, A Modern Approach.*

- Pdf - Computer Vision, A Modern Approach.

.

[3] Chahat Deep Singh. *CMSC 733 Computer Vision Class pdf.*

- Buildings built in minutes - An SfM Approach: CMSC733 Computer Vision.

.