ENPM 673 - Perception of Autonomous Robots

Pradeep Gopal
Sahana Anbazhagan
Srikumar Muralidharan

May 2, 2020

1 Introduction

Visual odometry is used for estimating the trajectory of the robot. In this project we have been given a data set with frames of a driving sequence taken by a camera in a car. We are to estimate the 3D motion of the camera and provide a plot of the trajectory of the camera. We have been given the scripts to extract the intrinsic parameters using which the essential and Fundamental matrices can be calculated. Using this we compute the rotation and translation components thus estimating the 3D motion.

2 Data Preparation

The given data set has the images in Bayer format. As the first step in data preparation we have to recover colored images from the given data set. This can be done using the demosaic function with GBRG alignment. Now we have converted the Bayer encoded image into a normal color image. The next step is to extract the camera model parameters from the given ReadCameraModel.py file as fx, fy, `c_x`, `c_y`, `camera_img`, LUT in our case. The last step in the process of data preparation we have to undistort all the images. We have been given UndistortImage.py for the same.

3 Working and Methodology

3.1 Feature matching

The first step is to match the features. Feature matching is the task of establishing a correspondence between two images of the same object or the same scene.The common approach to do this is by detecting a set of feature points associated with each image descriptors from the image. Once the features have been extracted for two or more images, a preliminary feature match is established between the images. The general algorithm to perform the same is as follows:
1) Finding a set of distinct feature points.
2) Defining a region around each of the keypoints.
3) Extract and normalize the region.
4) Computing a local descriptor from the normalized region.
5) Match the local descriptors.
We have used SIFT keypoint and Flann based matching in our program for performing feature matching.

3.2 Estimation of fundamental matrix(F)

Fundamental matrix is a non-full rank matrix that relates to the corresponding points in two images from different views. This is used in epipolar geometry which is an intrinsic projective geometry between two views. This depends only on the camera's internal parameters that is the K matrix and the relative pose. Hence it is independent of the structure of the scene. The F matrix is a 3x3 matrix whose rank is made two by changing the diagonal elements of the matrix to make the linearly independent matrices. The s$_{33}$*elementismade*0*forcefullyandhencethematrixnowhasarankof*2.*Wehaveequations*1*and*2*togettheFmatrix.* $\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix}$

0(1)

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ x_mx'_m & x_my'_m & x_m & y_mx'_m & y_my'_m & y_m & x'_m & y'_m & m \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0 \tag{2}$$

We need at least 8 points to solve the given homogenous system. Hence this is also known as eight point algorithm. But since we have computed the point of correspondences using SIFT there are bound to be some noises and outliers. Thus in order to remove these outliers and maximize the inliers we have used RANSAC algorithm and obtained a better fundamental matrix. We have written a function called Fundamental_Matrix() which takes two parameters, each being the 8 features of two frames. Using all the above formulas and concepts we have computed the best F matrix stored in FinalFundamentalMatrix.

3.3 Estimation of Essential matrix(E)

We have the F matrix with us now and using this we can find the relative pose between two images. This can be found by calculating the essential matrix. This is another 3x3 matrix, but has some additional properties that relates the corresponding points with some assumptions about the camera unlike the F matrix.

$$E = K^T F K \tag{3}$$

The above equation is used to compute the essential matrix where K is the camera calibration matrix or is also known as the camera intrinsic matrix. From equation 3 it is evident that there is a direct dependency on the F and K matrices. The singular values of E may not necessarily be (1,1,0) due to noises in K. This can be corrected by reconstruction using the equation 4.

$$E = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T \tag{4}$$

The E matrix is defined in the normalized image coordinates which means the origin is at the optimal center of the image and F is defined in the original image space. Using the given equations and from the computed F matrix we have written a function called EssentialMatrix() with two parameters which are the K and F matrices.

3.4 Obtaining Rotation and Translation components

The camera pose will give us information about 6 DOF of which 3 are rotation, i.e., roll, pitch, yaw and 3 translation components along the x, y, z axes. We get 4 camera pose configurations from the E matrix. They are:

$$(C_1, R_1), \quad (C_2, R_2), \quad (C_3, R_3), \quad (C_4, R_4) \tag{5}$$

where C is the camera center and R is the rotation matrix. Camera pose (P) is given by equation 5.

$$P = KR \begin{bmatrix} I_{3x3} & -C \end{bmatrix} \tag{6}$$

From the E matrix in order to compute the 4 pose configurations we use:

$$E = UDV^T \tag{7}$$

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

So we get the 4 configurations as:

1. $C_1 = U(:,3)$ and $R_1 = UWV^T$
2. $C_2 = -U(:,3)$ and $R_2 = UWV^T$
3. $C_3 = U(:,3)$ and $R_3 = UW^TV^T$
4. $C_4 = -U(:,3)$ and $R_4 = UW^TV^T$

All this is done keeping in mind that det(R) = 1. If det(R) = -1 then C = -C and R = -R.

3.5 Obtaining the correct T and R and plotting

From the obtained 4 camera poses the depth of all points is estimated linearly using the cheirality equations. Now the R and T values are chosen based on whichever has the largest amount of positive depth values. Based on the rotation and translation parameters obtained between successive frames the position of the camera center can be plotted.

3.6 Triangulation and Cheirality condition

After obtaining the 4 camera pose configurations we are considering two poses and triangulating the 3D points using linear least squares. Now, we have to find a unique camera pose and this is done by using the Cheirality condition. This condition states that the reconstructed points must be in front of the camera. Not all the triangulated points will satisfy this condition due to the presence of noise. The best camera configuration will produce the maximum number of points that satisfy the condition. We have written a function called getTriangulationPoint() with 4 parameters, which are an identity matrix, features from the two images and the rotation and translation components obtained.

3.7 Non-Linear Triangulation

This is used to reduce the error in reprojection of the 3D points and the camera pose that was initially calculated linearly. This is possible because in case of linear triangulation the algebraic error is minimized. But we have geometrical error and in order to remove that we need to use non-linear triangulation which is given by the below mentioned equation. Here $\tilde{X}$ is the homographic representation of point X and $P^T{}_i$ is each row of the camera projection matrix P.

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

3.8 Non-Linear PnP

We know that linear triangulation helps in minimizing the algebraic error. Then the reprojection error or the geometrically meaningful error is computed by measuring the error between the measurement and the projected 3D point. The procedure is similar to non-linear triangulation and hence we can use the same equation as that. But a compact representation of the rotation matrix using quaternion is better to enforce orthogonality of the rotation matrix, R = R(q) where q is the four dimensional quaternion given by the following equation. This minimization is highly non linear because of the presence of the divisions and quaternion parameterization.

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

Using the above methodology we were able to plot and estimate the 3D motion of the camera. Adding the results obtained from our code below.

Figure 1: The corresponding xz plot for Frame 1399381484074456

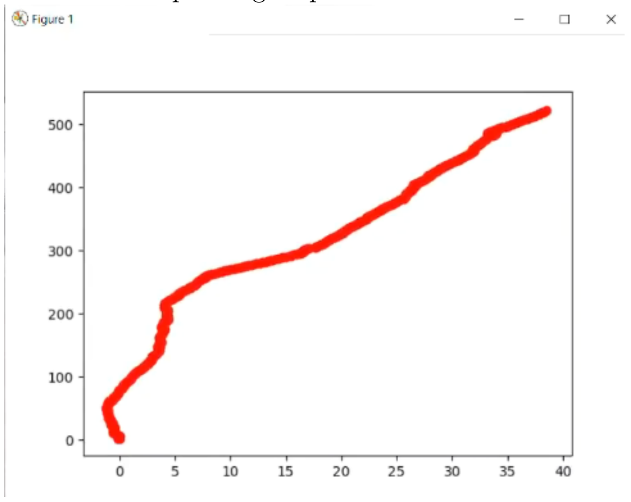


Figure 2: The corresponding xz plot for Frame 1399381522319395

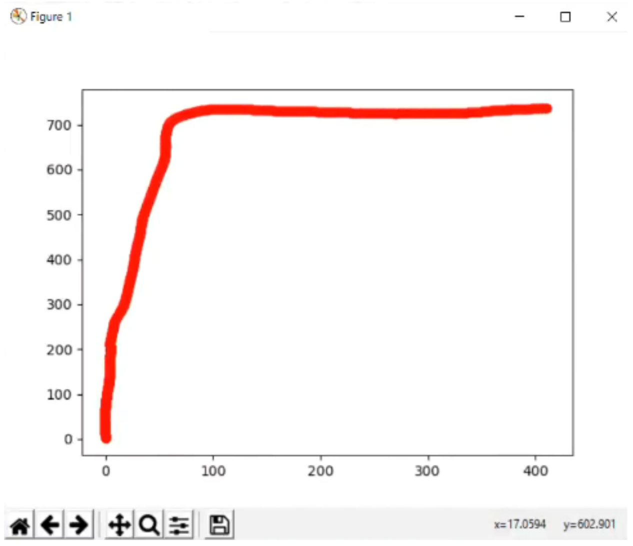


Figure 3: The corresponding xz plot for Frame 1399381652051744

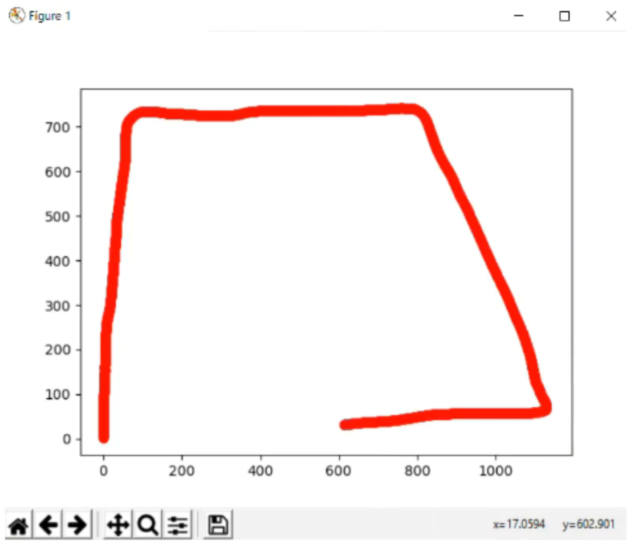


Figure 4: The corresponding xz plot for Frame 1399381672798998

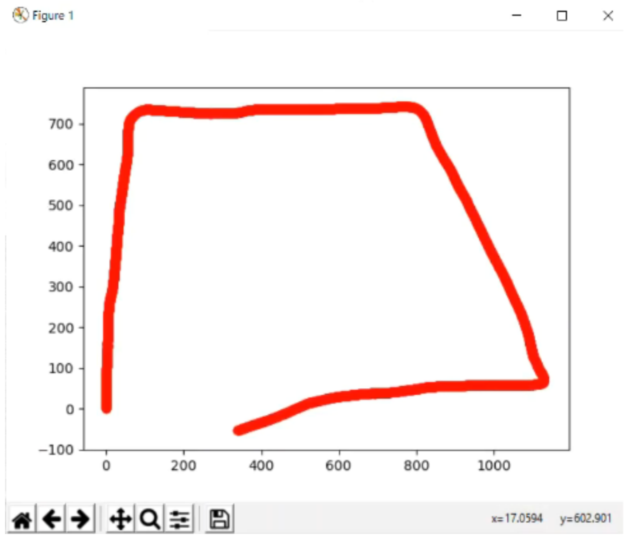
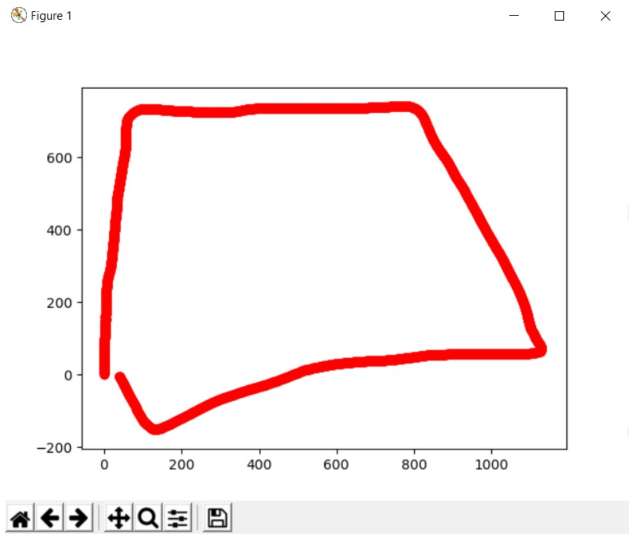
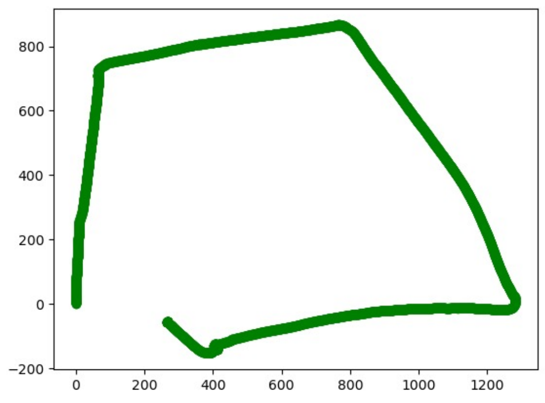


Figure 5: The corresponding xz plot for Frame 1399381701420103, which is also the final path traced



Attaching here the result obtained by using the inbuilt function.

Figure 6: The result obtained using inbuilt function



Hence we have inferred that our code works faster and computes the path quickly. It can be observed that the car starts and comes back to the same point after following a given path. The link to the output video is attached here.