University of Maryland

Perception for Autonomous Robots

Project - 5

Visual Odometry

Pruthvi Sanghavi

4 May 2020

Contents

1 Introduction

Visual Odometry is a crucial concept in Robotics Perception for estimating the trajectory of the robot (the camera on the robot to be precise). The concepts involved in Visual Odometry are quite the same for SLAM which needless to say is an integral part of Perception. 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.

2 Data Preparation:

1. The first step in the project was the preparation of the given dataset and reading all the images in the dataset. The input images are in Bayer format on which demosaicing function with GBRG alignment was used. Thus, the Bayer pattern encoded image img was converted to a color image using the opencv function: **color_image = cv2.cvtColor(img, cv2.COLOR_BayerGR2BGR)**

2. The next step in the data preparation phase was to extract the camera parameters using ReadCameraModel.py as follows: **fx , fy , cx , cy , G_camera_image , LUT = ReadCameraModel ('./model')**

3. The images in the given dataset were further Undistorted using the current frame and next frame using UndistortImage.py: **undistorted image = UndistortImage(originalimage,LUT)**

3 The Basic Pipeline

To estimate the 3D motion (translation and rotation) between successive frames in the sequence, the following steps were followed.

1. Find point correspondences between successive frames using a keypoint algorithm of your choice.

There are a number of algorithms available to find the keypoints and the descriptors of the two images. The algorithm generally used are: 1) ORB (Oriented fast and Rotated Brief) 2) SIFT (Scale Invariant Feature Tranform) and 3) SURF (Speeded up Global Features). For the current project we tried the ORB and the SIFT Algorithms. **SIFT** algorithm uses the **sift.detect()** function to find the keypoints in the images. **ORB** is computationally faster and has a good matching performance as compared to SIFT and SURF algorithms. These algorithms provide with the key points in both the images. Once the keypoints and descriptor objects are obtained, a matcher is used to find the points which match in the two images for that we have used the **Brute force keypoint matcher that uses orb descriptors**.

Once the matching points were obtained they were accessed. The result of matches = bf.match(des1,des2) line is a list of DMatch objects. This DMatch object has following attributes:

DMatch.distance - Distance between descriptors. The lower, the better it is.

DMatch.trainIdx - Index of the descriptor in train descriptors

DMatch.queryIdx - Index of the descriptor in query descriptors

DMatch.imgIdx - Index of the train image.

Thus, the matching points are obtained which we stored in the list to access it for later use.

2. Estimate the Fundamental Matrix using the Eight-Point Algorithm within RANSAC

The fundamental matrix (F) is a 3x3 matrix that relates the matching point between the two images taken from different views. The concept of **epipolar geometry (The epipolar geometry is the intrinsic projective geometry between two views.)** is used to derive the fundamental matrix.

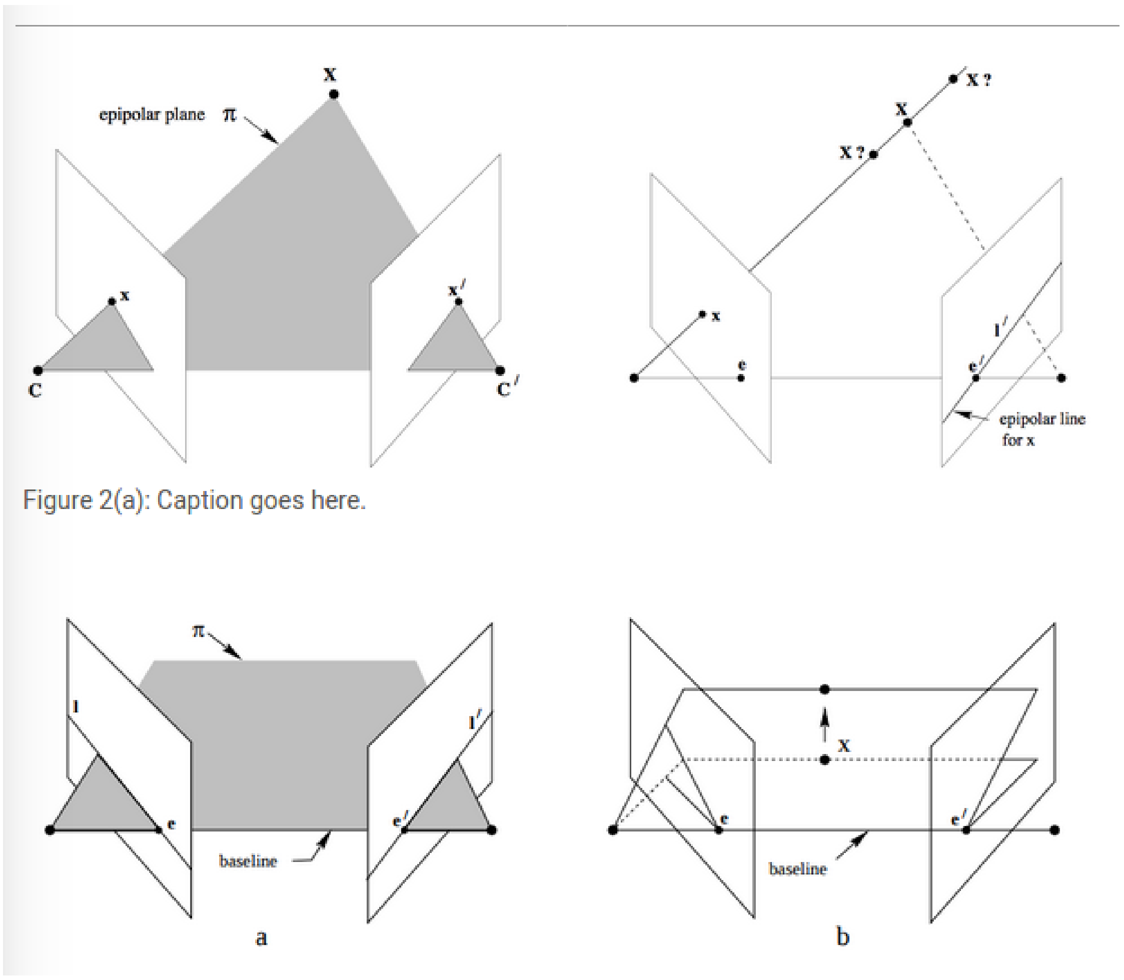


Figure 1: Concept of epipolar geometry

When a point X is viewed from two different angles in 3d, the same point appears as two different point x and $x\prime$ and they are the corresponding point. From the figure above,

Epipole is the point of intersection of the line joining the camera centers with the image plane. (see e and e in the Fig. 2(a))

Epipolar plane is the plane containing the baseline.

Epipolar line is the intersection of an epipolar plane with the image plane. All the epipolar lines intersect at the epipole.

Fundamental Matrix

$$\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Figure 2: fundamental matrix

As shown the F matrix is the algebraic representation of the epipolar geometry. For the points in the two images x_1 and $x_1^{'}$. **Fundamental Matrix**

$$\mathbf{x}_i'^{\mathbf{T}}\mathbf{F}\mathbf{x}_i = 0$$

Figure 3: epipolar constraint

The above condition is known as the epipolar constraint or the correspondence condition. Thus, a homogeneous transform can be setup to get the 9 values of the fundamental matrix.

$$\begin{bmatrix} x_i' & y_i' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

$$x_i x_i' f_{11} + x_i y_i' f_{21} + x_i f_{31} + y_i x_i' f_{12} + y_i y_i' f_{22} + y_i f_{32} + x_i' f_{13} + y_i' f_{23} + f_{33} = 0$$

Figure 4: homogeneous condition

Simplifying for the m matches,

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

Figure 5: correspondence equation

Eight Point Algorithm 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 above homogenous system. That is why it is known as Eight-point algorithm.

The above homogeneous equation which is of the form $Ax = 0$ can be solved using Singular value decomposition. Thus, on applying SVD the decomposition of the matrix A USV^T can be obtained, here U and V are the orthonormal matrices and S is the diagonal matrix containing the singular values.

If F has a full rank then it will have an empty null-space i.e. it won't have any point that is on entire set of lines. Thus, there wouldn't be any epipoles.

Estimate the Fundamental matrix F via RANSAC from point correspondences. The algorithms such as SIFT and ORB are noisy. Some of the matching points are incorrect which are called the **outliers**. Thus, in order to get better results, we need to perform outlier rejection. Various algorithms are available for the rejection of outliers such as least squaring, RANSAC etc. Here we have used RANSAC algorithm to find the inliers and get the best fundamental matrix. The algorithm can be shown as follows.

```
n=0;
for $i = 1{:}M$ do
    // Choose 8 correspondences, $\hat{x}_1$ and $\hat{x}_2$ randomly
    F = EstimateFundmentalMatrix($\hat{x}_1$, $\hat{x}_2$);
    $\mathcal{S} = \emptyset$;
    for $j = 1{:}N$ do
        if $| x_{2j}^T F x_{1j} | < \epsilon$ then
            $\mathcal{S} = \mathcal{S} \cup \{j\}$
        end
    end
    if $n < | \mathcal{S} |$ then
        $n = | \mathcal{S} |$;
        $\mathcal{S}_{in} = \mathcal{S}$
    end
end
```

Figure 6: RANSAC Algorithm

3. Estimation of Essential Matrix from the Fundamental matrix

The essential matrix is used to determine the relative camera poses between image frames. The essential matrix has 5 degrees of freedom and in this case it is computed from the fundamental matrix and the camera calibration matrix.

The Essential matrix E is expressed as :

$$E = K^T F K$$

Where, K is the camera calibration matrix and F is the Fundamental matrix.

Next, the rotation and translations have to be calculated from the essential matrix. However, the E matrix should be constrained to have a rank 2. This is achieved through singular value decomposition in the following way, E can be expressed as:

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

By constraining the value of the diagonal matrix and setting the last value of the diagonal to 0, this will constraint the rank of the Essential matrix E to 2. This in turn will provide the corrected Essential matrix that can be used to determine the camera poses.

4. Camera Pose computation from the Essential Matrix

Camera pose consists of Rotation and translation of the camera with respect to the world frame. The camera pose has 6 degrees of freedom,both the rotation and the translation have 3 degrees of freedom(R-(Roll,Pitch,Yaw) and T-(X,Y,Z).

The camera configurations can be obtained from the the Essential Matrix E. Four possible camera configurations can be obtained from the essential matrix E, which are expressed as:(C_1, R_1),(C_2, R_2),(C_3, R_3),(C_4, R_4). Here, C is the camera center and R is the Rotation matrix. These are calculated by decomposing E matrix through SVD($E = UDV^T$) and using the W matrix instead of the diagonal matrix D. W is $\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

The solution for the Translation and it is obtained from the last column of the U matrix, there are two possible solution for T. The configurations of the camera pose can be expressed 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 $R_4 = UW^TV^T$

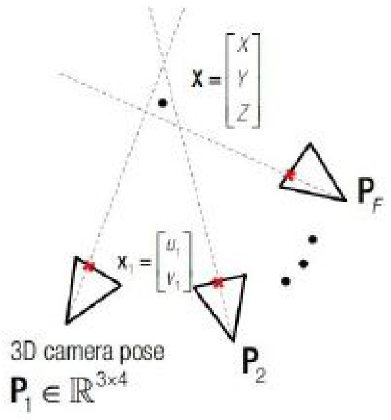
Figure 7: camera pose configurations

In addition, the determinant $det(R)$=1,If $det(R)$=1, the camera pose must be corrected i.e. C=C and R=R, this is done to reduce error. Finally, the camera pose P is calculated using the following equations:

$$P = KR[I_{3\times3} - C]$$

The next step would be to determine and select the best camera pose from the 4 different camera poses computed.

5.Linear Triangulation



$X = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$
P_F
$x_1 = \begin{bmatrix} u_1 \\ v_1 \end{bmatrix}$
3D camera pose
$P_1 \in \mathbb{R}^{3\times4}$
P_2

Figure 8: Triangulation of 3D points

Linear triangulation is used to determine the correct camera pose, this is achieved by checking for the Cheirality Condition. Which basically means that the points are in front of both the cameras. This is done using linear least squares method to check for depth positivity Z of the points with respect to the camera center. A 3D point X is in front of the camera if the following condition: r3(XC)¿0 where r3 is the third row of the rotation matrix (z-axis of the camera) is met. Since, not all points satisfy the this condition, this can be used to determine the correct pose.

Non-linear Triangulation For two camera poses the linearly triangulated points X, the locations of the 3d points are refined tom minimize reprojection error which is computed by measuring the error between measurement and projected 3d point and can be given as below,

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

Figure 9: non linear triangulation

X is the homogeneous representation of X and P^T, is each row of camera projection matrix P.
The minimization is highly nonlinear due to the divisions.

Non-linear PnP The linear PnP minimizes algebraic error. In non-linear prespective-n-points, the camera pose is refined to minimize the reprojection error between the measurement and the projected 3d point by enforcing orthongonality of the rotation matrix $R = R(q)$ and hence represented as below.

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

Figure 10: Non linear PNP

We compute P such that $P = KR[I_{3X3} - C]$, X is the homogeneous representation of X and q is the 4 dimensional quaternion.
The minimization is highly non-linear owing to the divisions and quaternion parameterization.

4 Operation with inbuilt functions

Using the built-in functions cv2.findEssentialMat and cv2.recoverPose from opencv, the following output was obtained.

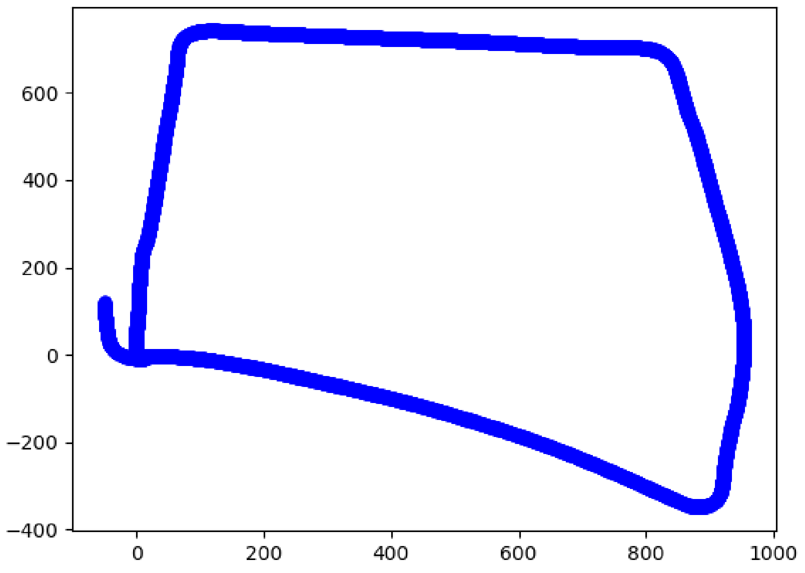


Figure 11: In-built function Output

5 Problems encountered

Following are the problems, we encountered while working on this project.

1) During the fundamental matrix calculation the values of some of the elements of the elements came out to be larger than is is recommended.

2) We found out that the matching points obtained using the SIFT and ORB algorithm came out to be different.

3) We tried plotting the camera centre position using different computers but every time the curve came out to be different.

6 Results

Link to results: `https://drive.google.com/drive/u/0/folders/1n6yKQ3y4bpfgSGLTlOYE-0WESuEAebIu`

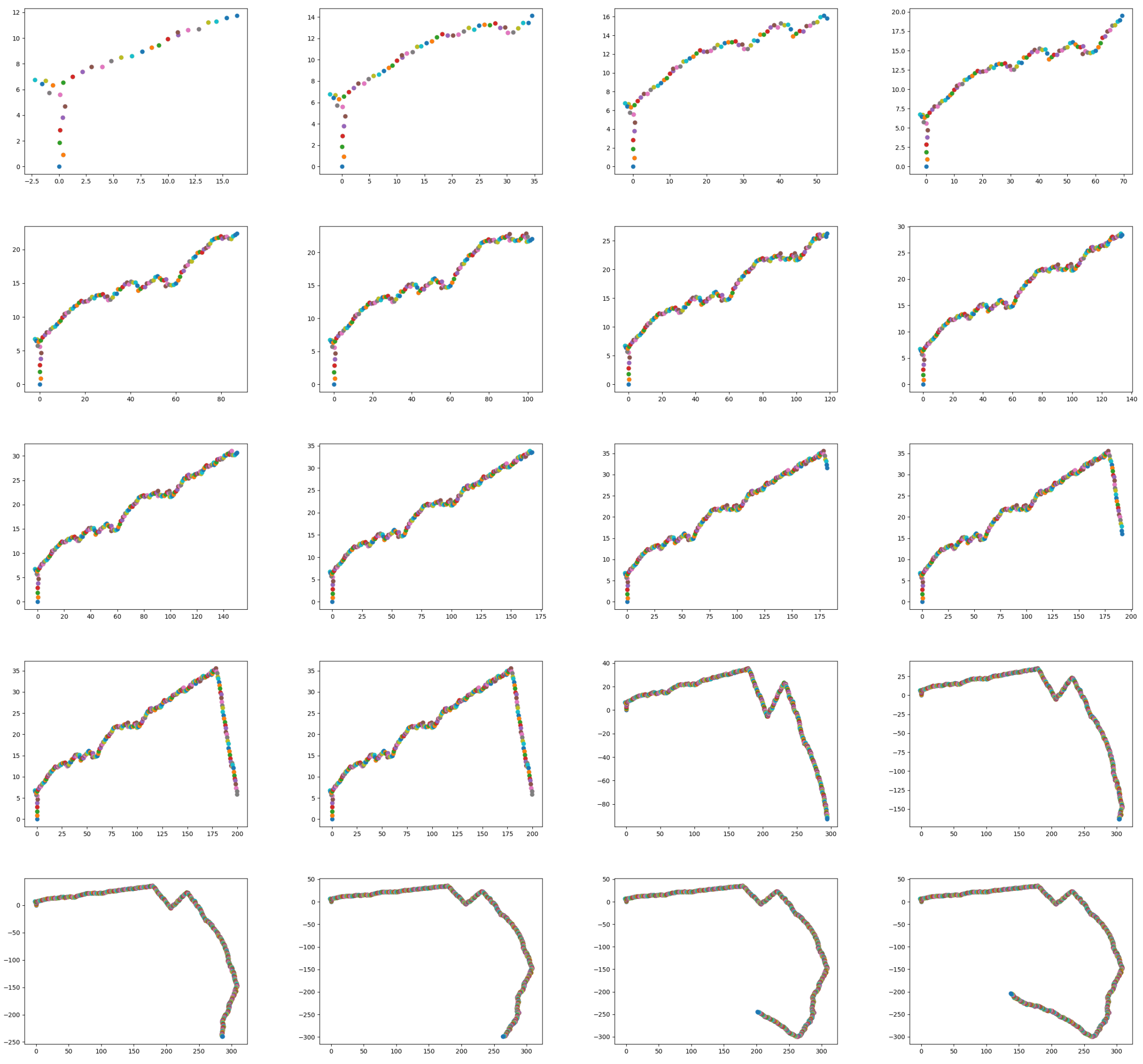


Figure 12:

7 References

1. Course Material
2. Lecture on Fundamental matrix: `https://www.youtube.com/watch?v=K-j704F6F7Q`
3. `https://cmsc733.github.io/2019/proj/p3/`
4. Opencv library