University Of Maryland

College Park

ENPM - 673 Perception for Autonomous Robots

Project 5 Report

Visual Odometry

Akshay Bapat
116215336

Patan Sanaulla Khan
116950985

Kulbir Singh Ahluwalia
116836050

4 May 2020

Introduction

Data Preparation

The given data images are in the Bayer format and to apply the SIFT algorithm we need to convert the images to the Grey Scale. Hence to achieve this we first convert the images to the normal BGR(Blue Green Red) scale and then we undistort the image based on the camera parameters. Once the undistorted image is achieved we apply on the Grey scale filter on the image through the cvtColor function again to generate the required set of images.

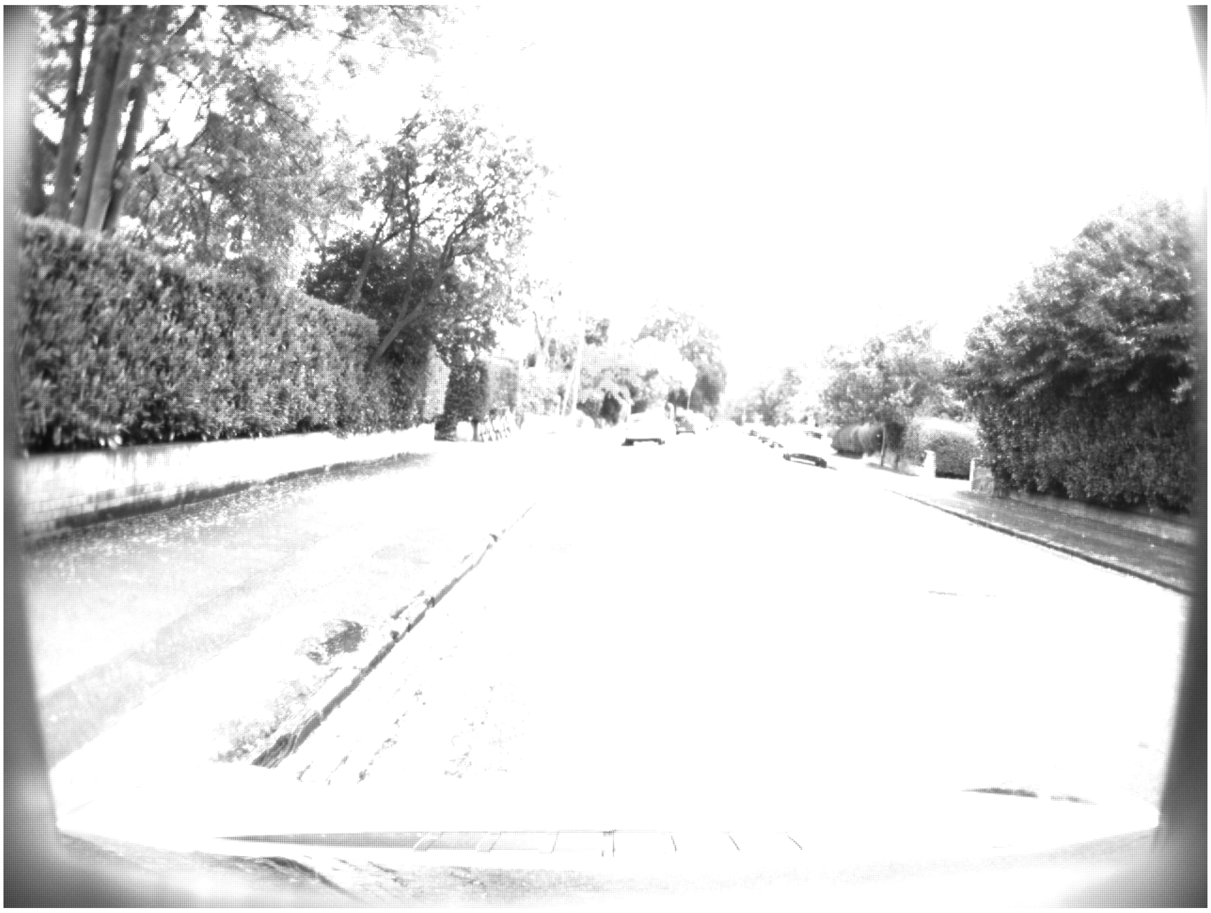

Figure 1: Image in the Bayer format

Listing 1: Code for generating the data

```
img = cv2.imread(os.path.join('./stereo/centre',fileName),0)
color_image = cv2.cvtColor(img,cv2.COLOR_BayerGR2BGR)
undistorted_image = UndistortImage(color_image, LUT)
greyScale = cv2.cvtColor(undistorted_image, cv2.COLOR_BGR2GRAY)
cv2.imwrite(os.path.join('./data',fileName), greyScale)
```

Figure 2: Same as Figure:1 but undistored and converted to Grey Scale

Step 1: Feature Matching and RANSAC

Feature Matching Using SIFT

To select the features form 2 consecutive images, we use the SIFT alogrithm provided by openCV as the function ORB_create(), where ORB algorithm uses FAST in pyramids to detect stable keypoints, selects the strongest features using FAST or Harris response, finds their orientation using first-order moments and computes the descriptors(2). The output for this algorithm generates a list of the best features for both the chosen images which are then matched based on the feature matching algorithm.

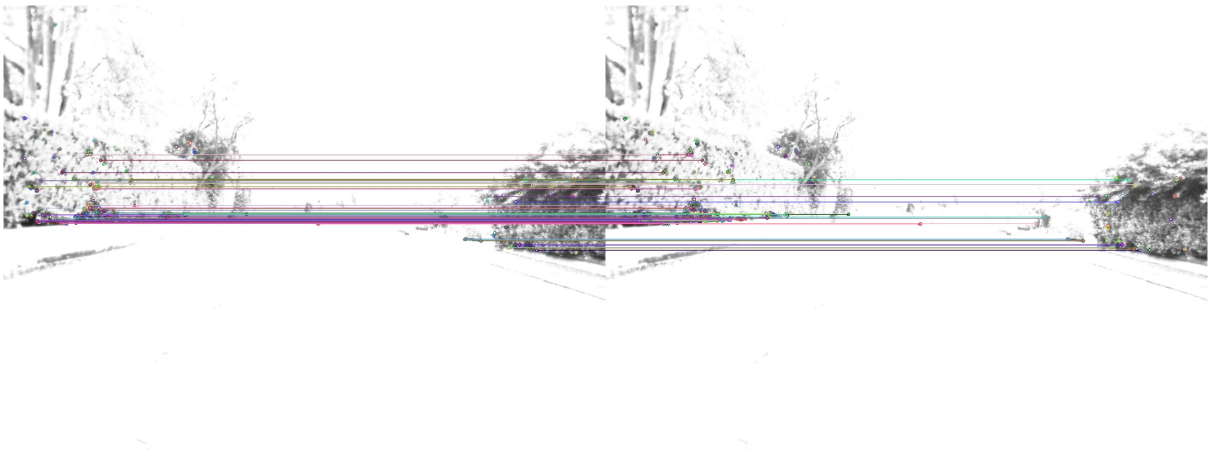

Figure 3: Feature matching using SIFT between 2 consecutive frames

RANSAC

Now from the generated feature points we select 8 random points using the random.sample() and then calculate the Fundamental matrix based on the algorithm. We find the fundamental matrix using the SVD operation where the equation is give as,

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

Now for a given 8 points we find the Fundamental Matrix (f) that satisfies the equation 1. This process is repeated for each randomly selected 8 points.

Step 2: Estimating Fundamental Matrix

Each of these randomly selected points generate a different value for the (f) matrix and using the equation given below where we select 2 points from the feature points list. The transpose of the second points is multipled with the Fundamental matrix and then with the first point (1), if the value of the product is less than the threshold given ϵ we consider that point is an inlier. We check this for every such possible point and maintain the count of the inliers. The randomly generated F Matrix with maximum inliers is considered as the final Fundamental Matrix.

$$x_2^T F x_1 < \epsilon \tag{2}$$

Step 3: Estimating Essential Matrix from Fundamental Matrix

After computing F, we compute the essential matrix "E".

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

where **K** is the camera calibration matrix or camera intrinsic matrix.

After we find the singular value decomposition of E, we observe that the sigma matrix is not $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$.

So, we do a SVD of E and then replace the sigma matrix with $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ while keeping U and V same.

After that, we multiply all three matrices to the new corrected E. Corrected E is calculated using:-

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

Step 4: Estimate Camera Pose from Essential Matrix

After we get E matrix, we find out 4 camera poses $(C_1, R_1), (C_2, R_2), (C_3, R_3)$ and $(C_4, R4)$ where $C \in R^3$ is the camera center and $R \in SO(3)$ is the rotation matrix. Thus the camera centre is of the form (x, y, z) and R is an orthonormal 3 by 3 rotation matrix.

When $\mathbf{E} = UDV^T$ and $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, we get four configurations of the camera which are:-

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 $R_4 = UW^TV^T$

Note that U(:, 3) means that we want the third column of U matrix which is obtained using U(:,2) in the code.
Important note: If determinant of R is -1, we have to correct the camera pose by replacing C with -C and R with -R.

Step 5: Check for Cheirality Condition using Triangulation

The reconstructed points must be in front of the cameras. To check this we perform the cheirality condition (to remove the ambiguity), where we triangulate the 3D points (given two camera poses) using linear least squares to check the sign of the depth Z in the camera coordinate system w.r.t. camera center. A 3D point X is in front of the camera if and only if it satisfies the below condition, where r3 is the third row of the rotation matrix (z-axis of the camera).

$$r_3(X - C) > 0 \tag{5}$$

We select the best camera configuration (C, R, X) that produce the maximum number of points that satisfy this condition.

Step 6: Plot the position of the camera center

In each iteration we get R (rotation) and t (translation) and we use them to get the homography matrix H corresponding to the transformation from the current frame to the next frame. Using the homography transformation matrix (H) we calculate the origin of the each individual frame and plot the X and Z coordinates of the origin as the Y is not in the plane of motion.

Extra Credit: Comparison

Using the cv2.findEssentialMat and cv2.recoverPose from openCV we generate a better video where the path or the trajectory is much smoother than the behaviour found in the previous case. It is evident that the threshold values estimated by the openCV might be the reason for the better performance.

Output Videos

The output videos can be found at: **Google Drive**
Link: https://drive.google.com/open?id=1CoZFeAbb7h$_oCaCc9aLz34y0FRPliLJA$

References

[1] Problem explanation - CMSC733 Project 3
`https://cmsc733.github.io/2019/proj/p3/pnp`

[2] ORB - SIFT algorithm
`https://docs.opencv.org/3.4/db/d95/classcv`$_{11}ORB.html$