ENPM 673 - Project 5 Visual Odometry

Hae Lee Kim

Zhengliang Liu

Table of Contents

I. Introduction

Visual Odometry is the process of incrementally estimating your position and orientation with respect to an initial reference frame by tracking visual features [1]. In this project, image frames of a driving sequence taken by a camera in a car are given. The goal of the project is to implement the steps to estimate the 3D motion of the camera by providing an output plot of the trajectory of the camera.

II. Data Preparation

For each image in the sequence, the image is converted from Bayer format to BGR, the intrinsic camera parameters (fx, fy, cx, cy, G camera image, LUT) are extracted using the provided script, and lastly, it is undistorted also using the provided script.

Fig. 1: Original frame

Fig. 2: Frame after data prep

III. Feature Matching and Fundamental Matrix

For two successive image frames, a keypoint algorithm is used to match features. In this project, SIFT was used as shown in Fig. 3 and 4.

Fig. 3: Original image frames

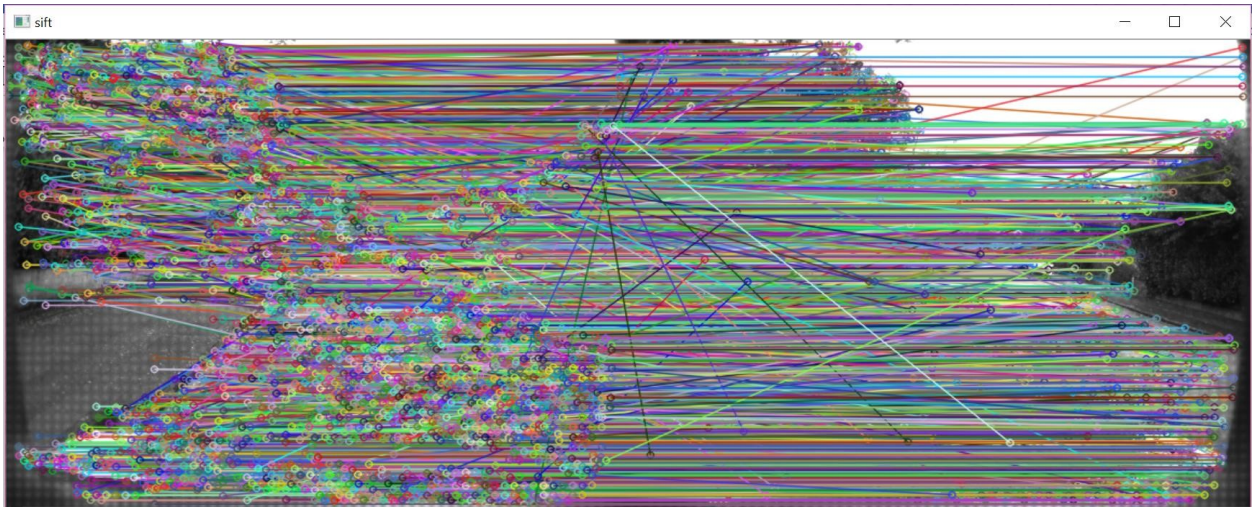

Fig. 4: After applying feature matching with SIFT

Feature matches need to be identified to be able to calculate the fundamental matrix. The Fundamental, F, matrix is a 3x3 (rank 2) matrix that relates the corresponding set of points from SIFT in two images from different views. Eight points need to be selected to create the F matrix. In order to best estimate the eight matches to use, the RANSAC outlier rejection method is used (as seen in the next section).

$$\begin{bmatrix} x_1x_1' & y_1x_1 & x_1' & x_1y_1' & y_1y_1' & y_1' & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx_m' & y_mx_m & x_m' & x_my_m' & y_my_m' & y_m' & x_m & y_m & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

With 8 correspondences between two images, the fundamental matrix can be obtained by solving the system of equations by solving the linear least squares (as seen above) using Singular Value Decomposition (SVD). The last column of the V is reshaped to be a 3 x 3, then the matrix is forced to be rank two by taking the SVD of the F matrix, substituting the last element of S with a 0, then re-multiplied to make the F matrix again. These set of steps to find the fundamental matrix is from the CMSC733 assignment pdf.

However, when we started debugging the code against the OpenCV built-in functions, we noticed that the fundamental calculation was the only part that was outputting different results. In order to minimize the error, the normalizing transformations were calculated for each of the correspondences to have the correspondences' mass center at the coordinate origin. Then, the A matrix was calculated and the rest of the steps outlined in the pdf were followed.

IV. RANSAC

As seen in the previous figures for the output of SIFT, the keypoint algorithm makes many correspondences even those that are outliers from noise in the data. In order to choose the best 8-point correspondences to use in estimating the fundamental matrix, the outlier rejection method RANSAC is implemented.

The steps for implementing the RANSAC method for creating the best fundamental matrix is as follows:

1. Initialize parameters: M = # of iterations, n = 0 (# of inliers), N = number of feature correspondences, inlier threshold = % condition for desired # of inliers divided by total number of correspondences (N), e = threshold for being an inlier (# close to 0)
2. For the number of iterations, M, do the following:
    a. Randomly choose 8 correspondences
    b. Restructure data to accommodate coordinate input format for the developed function for estimating the fundamental matrix
    c. Iterate through all correspondences and calculate the match error:

$$\frac{(x'^T F x)^2}{(Fx)_1^2 + (Fx)_2^2 + (F^T x')_1^2 + (F^T x')_2^2}$$

Where x' is the destination point, x is the source point, and the denominator is the sum of the square of the i-th entry of vector Fx. If the error is less than a small threshold e, the correspondence is appended into the "inliers" set.

3. If the number of inliers in the set is greater than n, then n is set to be the length of the number of inliers and loop will continue with the next set of random correspondences, otherwise, the inlier set with the most inliers will be saved along with the F matrix for at set of correspondences.

V. Essential Matrix

After obtaining the best fundamental matrix from RANSAC, the essential matrix can be computed by using the following equation: $\mathrm{E} = \mathrm{K}^T FK$, where K is the camera calibration matrix. Due to the noise of K, the essential matrix needs to be corrected by reconstructing it with (1,1,0) singular values, $\mathrm{E} = \mathrm{U}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}V^T$

VI. Camera Pose

The camera pose includes 6 degrees of freedom which are rotation and translation respect to the world frame. From the essential matrix E, 4 camera pose configurations $(C_1, R_1), (C_2, R_2)$, (C_3, R_3) , (C_4, R_4) where $\mathrm{C} \in \mathrm{R}^3$ and $R \in SO(3)$ can be obtained. The equations of four camera pose configurations are the following:

1. $\mathrm{E} = \mathrm{UDV}^{\mathrm{T}}$ and $W = \begin{bmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}$

2. $C_1 = U[:,3]$ 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 $R_4 = UW^TV^T$

3. $\mathrm{P1} = \mathrm{KR}[I_{3X3} - \mathrm{C}]$ (previous frame)

   $\mathrm{P2} = \mathrm{KR}[R_2 - \mathrm{C}]$ (current frame)

4. Check if the $\det(\mathrm{R}) = -1, if\ so\ make\ C = -C$ and $R = -R$

VII. Triangulation

Linear triangulation method can be used to find the 3D point X which is the observed points. X_i can be used to back project the two image points and determine their intersection. From reference lecture slide, the X_i can be found by using following steps in a least squares sense:

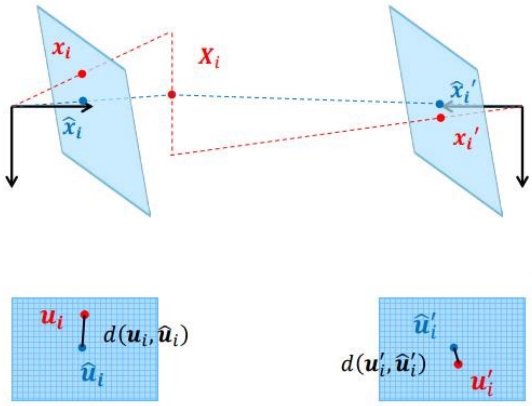

Fig. 5: Natural estimate for X_i

Each camera model gives two equations on the three entries of X_i

$$\tilde{u}_i = P\tilde{X}_i \text{ and } \tilde{u}_i' = P'\tilde{X}_i$$

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \times \begin{bmatrix} P^{1T} \\ P^{2T} \\ P^{3T} \end{bmatrix} \tilde{X}_i = 0 \text{ and } \begin{bmatrix} u_i' \\ v_i' \\ 1 \end{bmatrix} \times \begin{bmatrix} P'^{1T} \\ P'^{2T} \\ P'^{3T} \end{bmatrix} \tilde{X}_i = 0$$

Combining these two equations provide an over determined homogeneous system of linear equations which can be solved with SVD to obtain X_i .

$$\begin{bmatrix} v_i P^{3T} - P^{2T} \\ u_i P^{3T} - P^{1T} \\ v_i' P'^{3T} - P'^{2T} \\ u_i' P'^{3T} - P'^{1T} \end{bmatrix} \tilde{X}_i = 0 \text{ where } \begin{bmatrix} v_i P^{3T} - P^{2T} \\ u_i P^{3T} - P^{1T} \\ v_i' P'^{3T} - P'^{2T} \\ u_i' P'^{3T} - P'^{1T} \end{bmatrix} = A$$

Then the 3D coordinate can be obtained by apply SVD to A and take the last column of V matrix with normalization.

VIII. Cheirality Condition

Using the Cheirality Condition the disambiguate can be removed and the correct unique camera pose can be obtained. Two conditions were used for checking:

1. Check the sign of the depth Z in camera coordinates system w.r.t. center.

2. A 3D point X is in front of the camera iff: $r_3(X - C) > 0$ where r_3 is the third row of the rotation matrix. Each translation and rotation matrix will be counted when the conditions are true and the one with maximum number of true marks will be used as the best camera configuration.

```
ch= R2[2]@(x_check[0:3]-C2)

if ch>0 and x_check[2]>0:
    check_number+=1
```

Fig. 6: Implementing cheirality condition

IX. Plotting x,z

In order to be able to do a comparison plot for section X, the x and z coordinates were stored in a separate csv file to reduce processing time when plotting the built-in with together.

```
# Create CSV  and save data
with open('x_z_data.csv','a', newline='') as f:
    writer=csv.writer(f)
    writer.writerow([x1, -z1, x, -z])
```

Fig. 7: Creating CSV file to save data

X. Comparison

Built-in functions from OpenCV were implemented on the same sequence of image frames to compare the results with the trajectory plot from our own code. The feature matches were found the same way using SIFT. For each match, the position coordinates were extracted then the essential matrix was calculated using cv2.findEssentialMat(). Then the new pose was obtained by using cv2.recoverPose(). From the new pose, the x,z coordinates can be extracted and saved into a csv file (for faster plotting).

In Fig. 8, the blue plot shows the build-in function and the orange line shows the output of our own function. The created code and built-in functions code have a similar trend except our own code's trajectory is shifted slightly up in the z-direction and also has a x-direction offset. Also, the turning angle is slight off, however, the similar trend however confirms the validity of our algorithm.

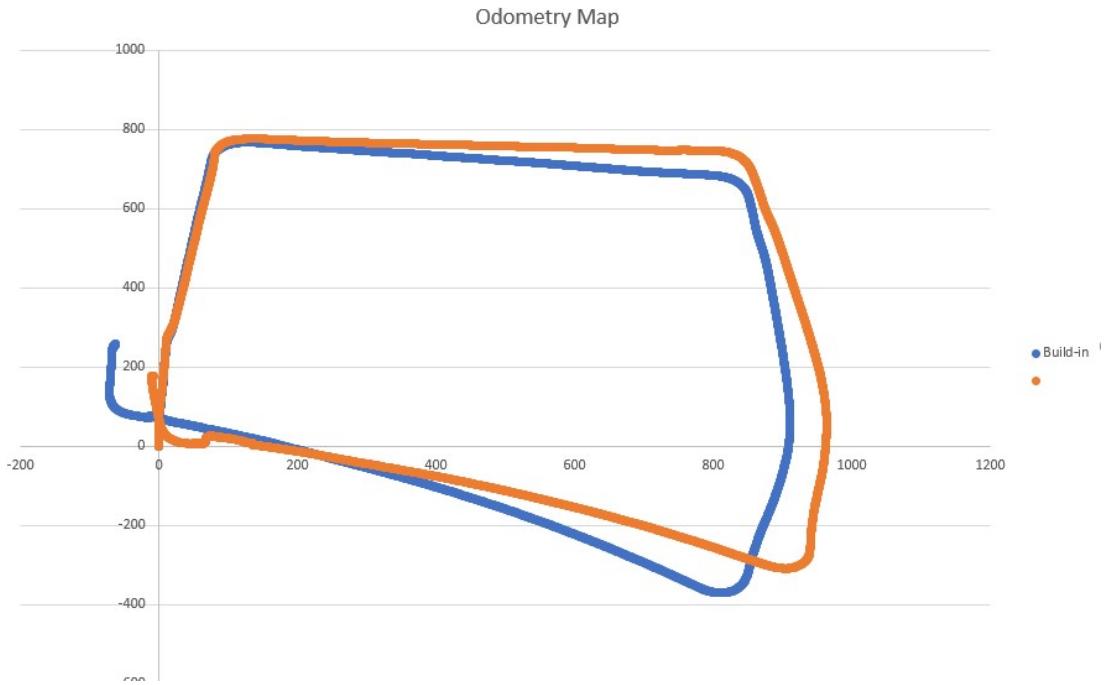

Fig. 8: Result comparison with built-in functions

Refer to this link for final output video:
https://drive.google.com/open?id=1qlUM5sxpJLfFdDOPp1hundKc-SkMnVLi

NOTE: The built in function in the video is colored red and our own code is blue.

References

[1] https://www.coursera.org/lecture/robotics-perception/visual-odometry-ReEv0

[2] https://www.uio.no/studier/emner/matnat/its/UNIK4690/v16/forelesninger/lecture_7_2-triangulation.pdf

[3] https://github.com/opencv/opencv/blob/master/modules/calib3d/src/fundam.cpp