Computer Vision Programming Assignment #2

Kyeong Seop Park[1]

[1]School of Electrical Engineering and Computer Science, GIST, Gwangju, Korea

kyeongspark@gist.ac.kr

Abstract

In programming assignment 2, I have implemented structure from motion (SFM) which performs a reconstruction of three-dimensional structure from two different view images. The implementation of the SFM was in the order of camera calibration, feature extraction and matching, essential matrix estimation, essential matrix decomposition, and triangulation. The resulted reconstruction is stored on 'starbucks.ply' using the input of the given image and 'self.ply' is the reconstruction output while the self-taken images is given as the input.

Introduction

Stucture from motion (SFM) is a process of building three-dimensional structures from multiple two-dimensional images inferring the camera poses [1]. In this programming assignment, from the given images, I have estimated the camera poses and done a 3D reconstruction using two images stored in a ply format. The explanation of the files inside the zip folder is described by the table below (Tab. 1).

Title	Description
calibration	contains calibration tools, self-taken pictures
pictures	contains input images
init.m	main MATLAB script for SFM
initSelf.m	script for 3D construction using self-taken images
normCoord.m	code for normalizing the pixel image coordinate system
RANSAC.m	code of the RANSAC algorithm
tri.m	code for estimating the camera motion
starbucks.ply	result of the 3D construction using the given images
self.ply	result of the 3D construction using self-taken images

Table 1. Description of the files inside the submitted zip folder

Methods

The pipeline of the SFM can be divided as first, camera calibration, which is extracting the intrinsic camera matrix, feature detection and matching, estimation of the essential matrix, extraction of the camera extrinsic matrix, and then generation of 3D points using triangulation. For reference, the scripts were executed on the MATLAB 2015b version.

1. Camera Calibration

Camera calibration is a technique of determining the camera's intrinsic parameters to figure out the geometric characteristics inside the 2D image creation process from the 3D real world coordinates. Intrinsic parameters include the focal length and the principal point which is used to derive a normalized coordinate system from the pixel image coordinate system. To elaborate, focal length means the distance between the camera lenses center and the image sensor. Also, the principal point is the intersection point of the optical axis and the image plane. With the focal length f_x, f_y and the principal point c_x, c_y respectively referred to the x-axis and the y-axis, we can obtain the camera intrinsic matrix K by the equation below,

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

I have used the camera calibration toolbox for MATLAB to get a camera intrinsic matrix of my own fixed-focus camera. I printed a checkerboard with the colored box having a length of 28mm and have taken five images from a different camera pose. The images used for calibration is mentioned below (Fig. 1).

2. Feature Extraction and Matching

A feature stores important information about the content of the image used for solving a computational task. Feature extraction happens by picking an image point which has dramatic changes regarding its adjacent points. For example, interestingly shaped patches, corners, edges, and peaks inside the image would be selected as features. Then, feature matching is performed which correlates features between two images of the same object. Inside the MATLAB code, feature extraction is performed by *vl_feat* and feature matching by *vl_ubcmatch* which are built-in functions inside the open-source library VLFeat.

3. Essential Matrix Estimation

Briefly, the essential matrix is used for finding the relative camera poses between two images. Figure 2

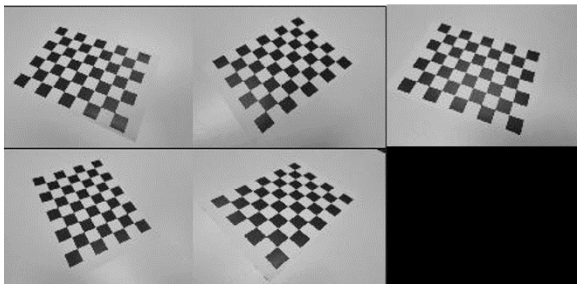

Figure 1. Checkerboard images used for camera calibration

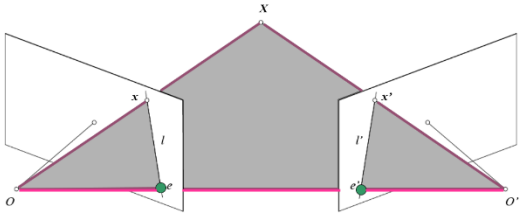

Figure 2. Outline of the epipolar geometry

gives an outline of the epipolar geometry which treats the geometric relation between the left image and the right image taken from the same scenery. Let the left camera center and the right camera center be denoted as O, O' respectively. The line connecting the two camera centers is called the baseline and intersections of the baseline and the image is named epipole denoted as e, e'. Also, if the world 3D coordinate X is projected in the left image and the right image as x, x', then the line l connecting x and e is called the epipolar line. With the essential matrix E and knowing the point x, we can derive the epipolar line l' that passes the point x'. Using the normalized image coordinate system $\hat{\mathrm{x}} = p_{norm}(u, v)$ instead of the pixel image coordinate system $\mathrm{x} = p_{img}(x, y)$ which can be easily derived by $p_{norm} = K^{-1}p_{img}$, the essential matrix E establishes a characteristic,

$$\hat{\mathrm{x}}^T E \hat{\mathrm{x}}' = 0 \qquad (2)$$

As we already know the corresponding points between the two images by feature matching, we can estimate the optimal essential matrix by the five-point algorithm with random sample consensus (RANSAC). RANSAC is an iterative method which finds out the best candidate that has the greatest number of inliers. Also, the five-point algorithm randomly selects five feature correspondences to generate a candidate of E and I have implemented the algorithm to run through eqn. (2) with an absolute threshold of 0.0005 for every feature correspondence to see how many inliers the candidate E possesses. This method iterates 10,000 times to find the best essential matrix.

4. Essential Matrix Decomposition

After getting the essential matrix, we have to figure out the optimal camera pose. By singular value decomposition (SVD) of E, E can be described as $E = Udiag(0,0,1)V$. Assuming that the first camera matrix is $P = [I \mid 0]$, the four possible choices for the second camera matrix is expressed as,

$$P' = [UWV^T \mid +u_3] \; or \; [UWV^T \mid -u_3] \; or$$
$$[UW^TV^T \mid +u_3] \; or \; [UW^TV^T \mid -u_3] \qquad (3)$$

where $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $u_3 = U(0,0,1)^T$.

Among the four candidates in eqn. (3), only one of the candidates show that the object is in front of the first and second camera. With reconstructing the 3D world coordinates from the corresponding points of the image using four possible P' matrices, only one matrix which is the optimal camera pose shows that the depth for the two cameras are in a positive value [2].

5. Triangulation

Triangulation is a method of getting 3D points from two camera poses and the image correspondences. The relation between the point of the pixel image coordinate x_{ci} and the 3D point X_i is $x_{ci} = PX_i$. Then, it implies $[x_{ci}]_\times PX_i = 0$. Overall, this is concluded by solving $AX = 0$, with,

$$A = \begin{bmatrix} xP^{3T} - P^{1T} \\ yP^{3T} - P^{2T} \\ x'P'^{3T} - P'^{1T} \\ y'P'^{3T} - P'^{2T} \end{bmatrix} \qquad (4)$$

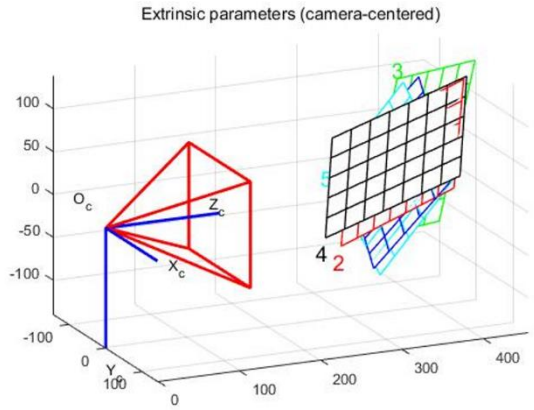

Figure 3. Visualization of the relative positions of the grids with the camera center fixed to the origin

To derive the 3D point X, we perform SVD to A as $U_1 diag(0,0,1) V_1$. The last column of V_1 gives the 3D homogenous coordinate, $[x_1 w; y_1 w; z_1 w; w]$, and with dividing w for its column's elements, we can obtain the 3D coordinate $[x_1; y_1; z_1]$. I have stored the coordinates if z_1 is above zero and using the second camera's optimal pose $P' = [R \mid t]$, the 3D coordinate derived from the second camera can be expressed as,

$$[x_2; y_2; z_2] = R[x_1; y_1; z_1] + t \qquad (5)$$

Results & Discussion

1. Camera Calibration

Using fixed-focus camera mode on the smartphone, Samsung Galaxy S20 Ultra, I have taken five pictures of the checkerboard from multiple perspectives. With the multiple images, I performed calibration with the help of the camera calibration toolbox for MATLAB (Fig. 3) and derived my own camera's intrinsic matrix,

$$K = \begin{bmatrix} 3384.0 & 0 & 2017.8 \\ 0 & 3419.3 & 1449.9 \\ 0 & 0 & 1 \end{bmatrix} \qquad (6)$$

2. Feature Extraction and Matching

Using the open-source library VLFeat, feature extraction was performed by *vl_feat* and feature matching by *vl_ubcmatch*. For the given Starbucks image, with the absolute threshold of 0.0005 and 10,000 iterations inside the RANSAC algorithm to find the essential matrix, the inliers of the match correspondences are represented as Figure 4. Out of 5,959 correspondences, the inlier count was 2,832. For the self-taken image of mountain dew, the parameters of the RANSAC algorithm were set with the absolute threshold of 0.05 and 5,000 iterations (Fig. 5). Out of 768 correspondences, the inlier count was 339.

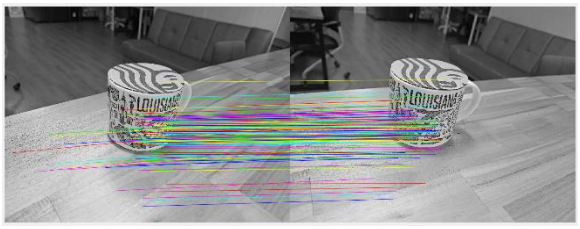

Figure 4. Visualization of the match correspondences. Although the inlier count was 2,832, the figure contains only 100 match correspondences.

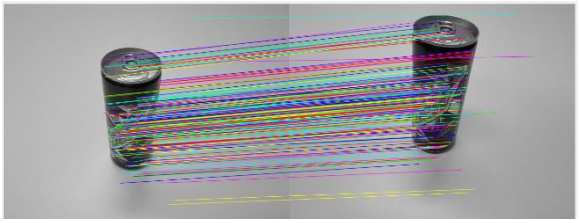

Figure 5. Visualization of the match correspondences of self-taken image

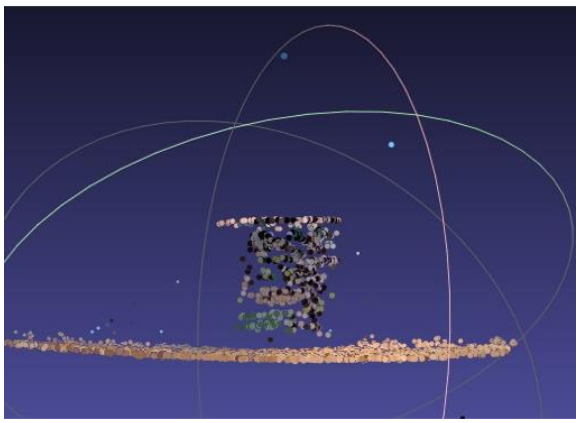

Figure 6. 3D reconstruction result from 'sfm03.png' and 'sfm04.png'

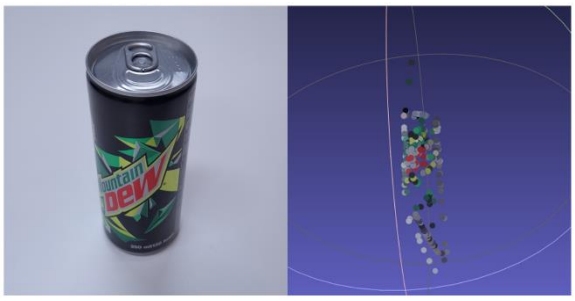

Figure 7. *Left:* Image of mountain dew from 'dew02.png'; *Right:* 3D reconstruction result from 'dew02.png' and 'dew03.png'

3. Result of the 3D Reconstruction

The result of the three-dimensional structure of the given Starbucks image is shown as Figure 6. By Fig. 6, we can identify the table and the Starbucks cup. Also, the result of the three-dimensional structure of the self-taken mountain dew image is shown as Figure 7.

Additional Methods

1. Growing Step

I have implemented the growing step as the additional method. The initialization step used the images of 'sfm03.png' and 'sfm04.png'. By the initialization step, we already know the some of the correspondences between the normalized image coordinate system of 'sfm04.png' and the world coordinate system. By performing match correspondences between 'sfm04.png' and

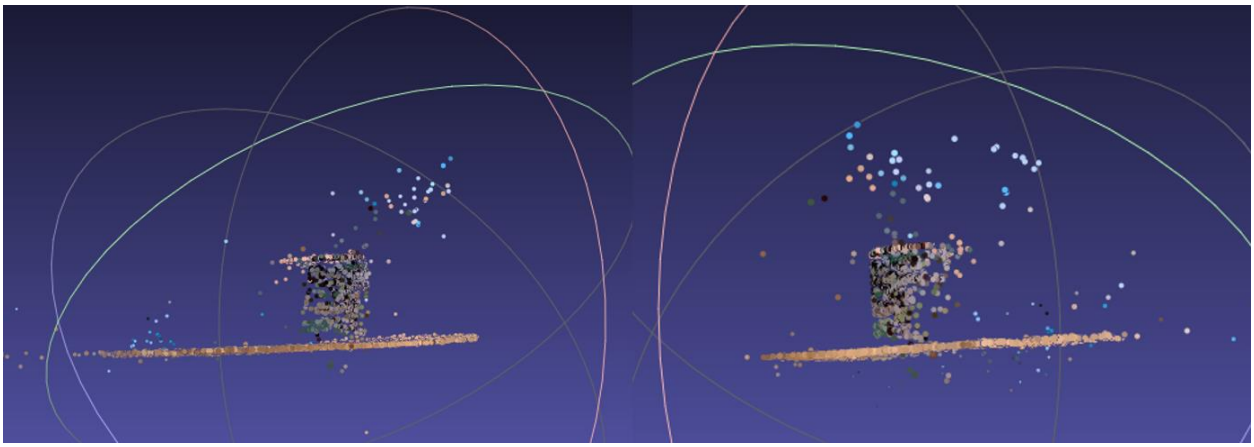

Figure 8. *Left:* 3D reconstruction result from 'sfm03.png', 'sfm04.png', and 'sfm05.png' when viewed on the left side; *Right:* 3D reconstruction result from 'sfm03.png', 'sfm04.png', and 'sfm05.png' when viewed on the right side;

'sfm05.png', I have found the intersections between the known 3D points for the normalized pixels of 'sfm04.png' and the match correspondences between 'sfm04.png' and 'sfm05.png'. The randomly selected three corresponding normalized pixels from the new 'sfm05.png' and the 3D world coordinate point are the input for the three-point algorithm [3].

By the three-point algorithm, we can get the candidates of the camera pose of the new image. Using the geometric error,

$$d(\hat{\mathrm{x}}, K^{-1}PX) \tag{7}$$

where $\hat{\mathrm{x}}$ is the corresponding normalized coordinate of 'sfm05.png', K is the camera intrinsic matrix, P is the camera pose candidate, X is the corresponding 3D world coordinate, and $d(x_1, x_2)$ stands for the function of the squared distance between x_1 and x_2. The threshold of the geometric error was set to 0.5 with 7,500 iterations. After getting the best camera pose candidate, I got the inliers between 'sfm04.png' and 'sfm05.png' and then performed triangulation to reconstruct 3D points. The newly constructed 3D world coordinate count was 1,499 where the original count of the construction was 3,472 and the number of the newly reconstructed points was 4,971. The result is shown above (Fig. 8). For reference, the image is saved on 'addMethod.ply'.

References

1. Snavely, N., S.M. Seitz, and R. Szeliski, *Photo tourism: exploring photo collections in 3D*, in *ACM siggraph 2006 papers*. 2006. p. 835-846.
2. Hartley, R. and A. Zisserman, *Multiple view geometry in computer vision*. 2003: Cambridge university press.
3. Haralick, B.M., et al., *Review and analysis of solutions of the three point perspective pose estimation problem.* International journal of computer vision, 1994. **13**(3): p. 331-356.