Computer Vision Programming Assignment 2

Sunghyun Kang[1],
[1]Department of Electrical and Computer Science, GIST, Gwangju, 61005 Republic of Korea

Through this experiment, we will look for the basis of SFM (Structure from Motion) from the beginning to the top. To enable this, we will obtain multiple images and use SIFT (Scale Invariant Feature Transform) method to obtain the features. Then we will see the matches of two images and choose the best pair that shows a maximum number of matches. From those matches, we will apply RANSAC (Random Sample Consensus) to get the essential matrix and its matches. Lastly, we will reconstruct the depth of the image by using the triangulation method and recover the 3D object.

***Index Terms*—SIFT, RANSAC, essential matrix, camera matrix, SVD, triangulation.**

I. Introduction

IMAGE depth estimation with 2D imaging has been a big issue in recent several years. Since this technique is really useful in various fields, there are various ways to reconstruct the 3D model. In this assignment, we will reconstruct the image from scratch to the top by following these steps:

- Extract features from two images using SIFT and normalizes the coordinates by using the given camera's intrinsic matrix K.
- Find reasonable matching features by using vl_ubcmatch[1] and implementing RANSAC. While doing RANSAC, find estimated essential matrix E by using five point algorithm.
- By using SVD (Single Value Decomposition), get 4 estimated camera pose matrix P and do the triangulation to get the one correct P and reconstructed 3D model.

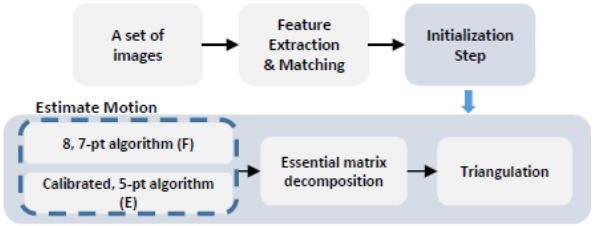

Fig. 1. The overall outline of our implementation. We only used 5-point algorithm for essential matrix.

II. Methodology

A. SIFT and normalizing coordinates

To obtain the feature points of two images, SIFT method was implemented. To normalize the coordinates of thoese features we implemented the equation described below 1. Then for initial matching point estimation, we used vl_ubcmatch method in the vlfeat library[1].

$$p_{norm} = K^{-1} p_{image} \tag{1}$$

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Where:

- p_{image}: original image coordinate (dimension: 3×1).
- p_{norm}: normalized coordinate (dimension: 3×1).
- K: camera intrinsic matrix.
- f_x, f_y: focal length of the camera (x, y axis).
- c_x, c_y: principle point of the camera (x, y axis).

B. RANSAC and essential matrix

The overall outline is described in algorithm 1.

Algorithm 1 RANSAC methodology

Input: normalized featured image coordinates F_a, F_b, initial image matches M_i, five point algorithm method $FPLA(A, B)$, threshold t, iteration time I.
Output: Essential matrix E, reasonable matching points M_r

```
while iteration time < I do
    A ← Random(F_a, 5)    ▷ 5 rand matching point in F_a
    B ← Random(F_b, 5)  ▷ corresponding 5 point from F_b
    E ← FPLA(A, B)
    temp ← 0
    for k ← 1 to M_i do
        A_cord, B_cord ← M_i        ▷ points from each images
        if A^T_cord × E × B_cord < t then
            temp ← temp + 1
            Append(temp_cord, A_cord, B_cord)
        end if
    end for
    if temp > inliers then
        inliers ← temp
        M_r ← temp_cord
    end if
end while
```

In the optimal case, the essential matrix follows this equation:

$$x_a^t \times E \times x_b = 0 \tag{3}$$

Where:

- x_a^t: matched image coordinate from the first camera, this is transposed coordinates (dimension: 3×1, normalized).

- x_b: matched image coordinate from the second camera (dimension: 1×3, normalized)
- E: essential matrix that we obtained by five-point algorithm (dimension: 3×3).

However, because of the distortion of the camera and other limitations, it is hard to make a result of equation 3 as a zero vector. Hence I set the threshold as small as possible and ran the RANSAC like algorithm 1.

C. Essential matrix decomposition

To achieve the estimated camera matrix, we have to follow the steps described in below:

$$SVD(E) = U \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \times V^t \tag{4}$$

$$u_3 = U \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$P_1 = [UWV^t| + u_3] \tag{6}$$

$$P_2 = [UWV^t| - u_3] \tag{7}$$

$$P_3 = [UW^tV^t| + u_3] \tag{8}$$

$$P_4 = [UW^tV^t| - u_3] \tag{9}$$

Where:

- SVD: stands for Single Value Decomposition.
- U, V^t: matrix obtained from single value decomposition of E (dimension: 3×3).
- u_3: last column vector of U (dimension: 1×3), it is set as translation matrix.
- W: orthogonal matrix.
- P_1, P_2, P_3, P_4: estimated camera pose matrix.

Note that the determinant of UWV^t and UW^tV^t are both to be 1. If they are -1, then the position of the camera matrix turned out to be opposite, thus have to set the P_1 to P_4 as negative. Otherwise, then the implementation should be wrong. In addition, only one of the camera pose matricies is valid. This will be evaluated in the next step.

D. Triangulation

In the triangulation part, we will reconstruct the 3D coordinates from 2D pixel images by following these equations. Please note that my implementation for this is quite a different that from the project spec sheet. I referenced it in here [2].

$$R_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} p^{1t} \\ p^{2t} \\ p^{3t} \end{bmatrix} \tag{10}$$

$$R_2 = P = \begin{bmatrix} p'^{1t} \\ p'^{2t} \\ p'^{3t} \end{bmatrix} \tag{11}$$

$$A = \begin{bmatrix} xp'^{3t} - p'^{1t} \\ yp'^{3t} - p'^{2t} \\ x'p^{3t} - p^{1t} \\ y'p^{3t} - p^{2t} \end{bmatrix} \tag{12}$$

$$A\vec{x} = \vec{0} \tag{13}$$

Where:

- R_1: camera pose matrix of first image. It is fixed with identity matrix and zero translation matrix (dimension: 3×4).
- R_2: camera pose matrix of second image. We have to run this for all P_1, P_2, P_3, P_4.
- x, y: x and y coordinates of first image that normalized.
- x', y': x and y coordinates of second image that normalized.
- $\vec{x}$: 3D coordinates that we want to recover.

From obtained linear combination matrix A and homogeneous linear equation 13, we can derive the homogeneous coordinates $w\vec{x}$ in 3D vector space and obtain $\vec{x}$. This will be followed by these equations:

$$SVD(A) = U \times W \times V \tag{14}$$

$$X' = V \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, X_{c1} = X' \times \begin{bmatrix} 1/w \\ 1/w \\ 1/w \\ 1/w \end{bmatrix} \tag{15}$$

Where:

- U, W, V: matrix obtained from single value decomposition of A, V's dimension is 3×4.
- X': homogeneous coordinates in 3D space (i.e. $w\vec{x}$), dimension is 4×1.
- w: last element in X'.
- X_{c1}: 3D coordinate we want to obtain from camera 1. The last element should be omitted, thus dimension 3×1.

Also, to validate these coordinates in other camera matrix, it is necessary to implement this following step:

$$X_{c2} = R \times X_{c1} + t \tag{16}$$

Where:

- X_{c2}: 3D coordinates we obtained through the equation, this is from camera 2 (dimension 3×1).
- R: matrix that we obtained as UWV^t or UW^tV^t. This should be the same matrix that we used to obtain X (dimension 3×3).
- t: translation matrix u_3 or $-u_3$. This also should be the same matrix that we used to obtain X (dimension 3×1).

Hence, we are able to obtain X_{c1}, X_{c2} for each matching coordinates for each P_1, P_2, P_3, P_4. From there we can observe whether the X_{c1}, X_{c2} coordinate z value is positive or negative. We should exclude all coordinates that show a negative z value since it implies that the target coordinate locates behind the camera. By doing those sessions, we can evaluate which P_n can produce the most possible matching coordinates. Thus we can get the valid P_n and reproduce 3D coordinates from it.

III. Experiment

For this programming assignment, I submitted the files described in table I. Also, I set my local environment like this:

- MATLAB version: 2022a
- VLFeat version: 0.9.21
- Camera Calibration Toolbox for MATLAB version: December 4^{th}, 2003

To begin with, I chose images sfm03 and sfm04 to obtain best results. Although it does not shows the maximum number of matching points in vl_ubcmatch, those two images were most optimal to see the results since the pair (sfm09 and sfm10) that showed having most matches are containing noisy background compared to sfm03 and sfm04.

Also, I did 10,000 iteration time and threshold as 0.0007 for RANSAC and obtained around 1000$\tilde{1}$500 reasonable matching pair.

TABLE I
IMPORTANT FILE LIST IN CODES DIRECTORY

File list	File explanation
step1.m	Image feature extraction and matching
step2.m	Running RANSAC to get E and good matching
step3.m	To get 4 candidates of camera pose matrix
step4.m	Triangulation and 3D point estimation
matrix_normalize.m	Coordinate normalization
RANSAC.m	Where the RANSAC1 happens
triangulation.m	Where the triangulation part happens
color_reshape.m	recover the color for 3D images
write_ply.m	External source code to get .ply
match_plot.m	External source code to get match plot

IV. Results

A. SIFT and initial matching results

In imaging matching, I implemented vl_ubcmatch from VLfeat library[1]. For all image pairs, the result was shown as figure 2.

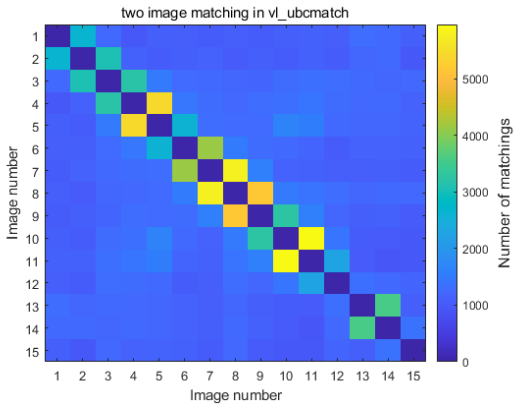

Fig. 2. matching result from all combination of given images. Note that image 1 is sfm00, image 2 is sfm01, ... goes on.

Here, we can see that (sfm03, sfm04) pair, (sfm06, sfm07) pair, (sfm07, sfm08) pair, and (sfm09, sfm10) pair show the most matching compared to other pairs. However, As we can check the images in figure 3, some of the matches are due to high background noise since they are not able to match the target object. Thus I used (sfm03, sfm04) pair for this project due to those reasons.

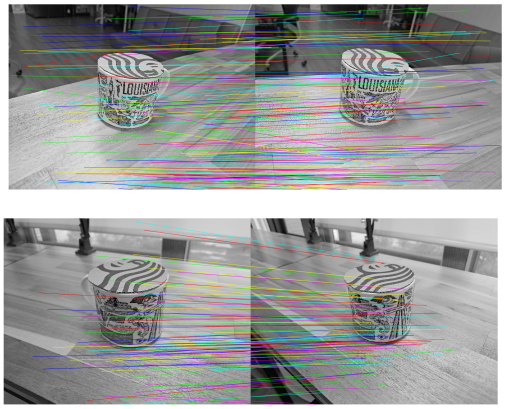

Fig. 3. Up: image 9 and 10 matching result after the RANSAC (randomly selected 100 points within 3224 matchings). Down: image 3 and 4 matching result after the RANSAC (randomly selected 100 points within 2951 matchings).

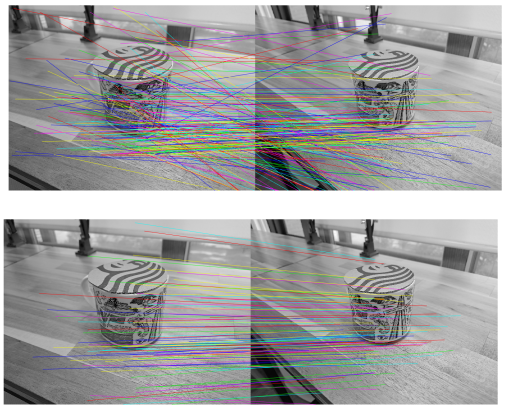

Fig. 4. Up: image 3 and 4 matching result before the RANSAC (only with vl_ubcmatch, randomly selected 100 points). Down: image 3 and 4 matching result after the RANSAC (randomly selected 100 points from RANSAC results).

From those matches, I checked the scores which represents the estimated distance between those matches. In addition, from given camera matrix K for sfm images I normalized all coordinates using equation 1.

B. RANSAC results

After the matching image selection and vl_ubcmatch method implementation, RANSAC was implemented to get essential matrix E and more reasonable matching points. In this lab, the iteration time was set as 10,000 times and the threshold as 7×10^{-5}. For the essential matrix five-point algorithm was used. The result after these experiments can be seen in figure 4.

Here, we can observe that RANSAC can not only infer essential matrix E but also gives us reasonable matching points

compared to original matching points. Especially, we can see the crossing points that are seen upper side of figure 4 are gone on the downside of it.

TABLE II
HYPERPARAMETER VALUES DURING RANSAC

Hyperparameter	Value
Iteration time	10,000
Step size (α)	0.0007

C. Triangulation results

From obtained essential matrix from RANSEC, I decomposed that matrix using singular value decomposition (equation 4). From there I obtained four camera matrices (equation 6, 7, 8, and 9) and checked whether the determinant of UWV^t and UW^tV^t are 1 or not. If it showed as -1, then I re-runed RANSEC until I obtained the determinant as 1 (This also can be solved by letting $R = UWV^t$ and UW^tV^t as $-R$).

After obtaining 4 camera pose matrices, I estimated the reconstructed 3D coordinates using equation 15, 16. From those coordinates, I choose one camera matrix that showed many positive z coordinates. Then I restored the RGB value of each coordinate and included it in the 3D coordinates. The final result is shown in figure 5. As we can see, the coordinates were reconstructed but only a few parts of the original object appeared. Also, the floor was observed as not flat.

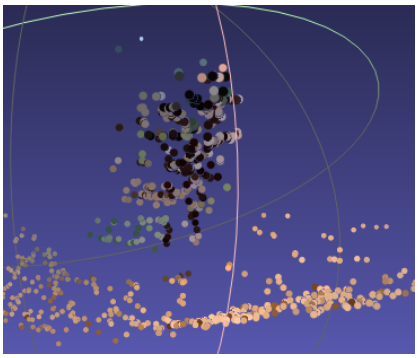

Fig. 5. The result of this overall challenge, as we can see there is clear brown desk surface with black & green color cup.

V. 3D IMAGE RECONSTRUCTION FROM MY IMAGE

To solve this problem with my image, I used my friend's phone camera to set the fixed focus. Then I printed out the given calibration checkerboard and used it to calibrate the camera and achieve the intrinsic matrix K of it. From obtained K, I utilized it to reconstruct the 3D coordinates from my image.

A. Camera calibration

Initially, we need to print out the checkerboard (9 box in x axis, 7 box in y axis, 28mm square each) and place it in the specific location where we will place the object that we want to reconstruct. Then I photoed 3 checkerboard images using a fixed focus camera and marked the grid for each image 9 times to minimize the calibration error. The calibration result is shown in figure 6 and table III.

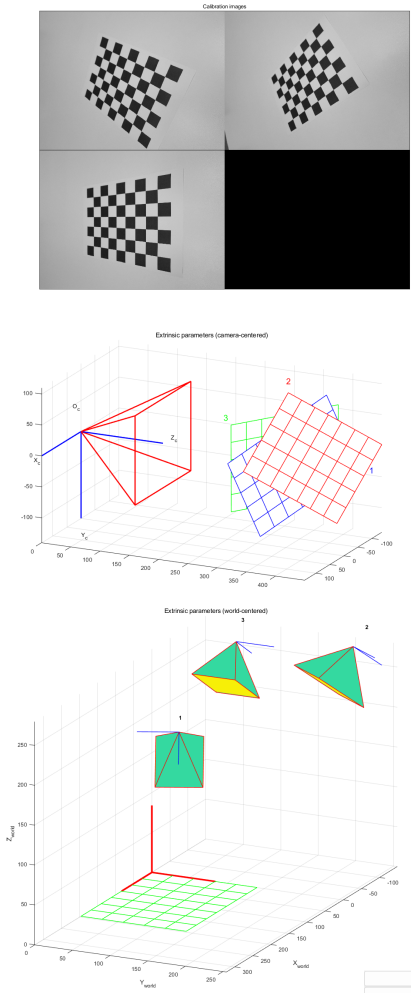

Fig. 6. From up to down: list of checkerboards that I used for calibration, extrinsic parameters result, and the view that seen in golobal area.

TABLE III
CAMERA CALIBRATION RESULTS

camera parameters	x axis	y axis
focal length	2846.56	2935.25
focal length error	368.64	267.24
principle point	2015.5	1511.5
principle point error	0.0	0.0

B. Image obtain

Then I obtained the image of the object in the same focus and the same position where the checkerboard was located. The object was selected based on the special characteristics (e.g. having many edges, a special color that differentiates from the background, etc.,). Hence I selected the object in figure 7 and did the RANSAC algorithm to see the matching.

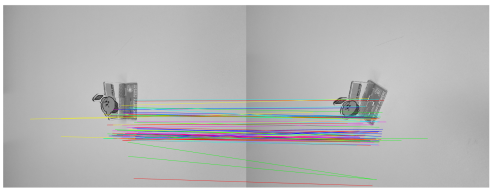

Fig. 7. The matching of the image that I taken for my own. The background is white desk, and I set my figure in the middle. Hence the feature was detected and matched (around 110 matching).

C. results

However, the reconstructed image from that figure was not well observable as we saw in figure 5. This may be caused due to the lack of skills as I took the image on the upper side, thus make it hard to estimate the image depth compared to the image that I obtained in figure 4.

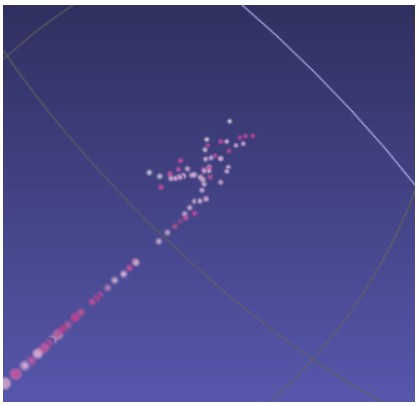

Fig. 8. Result of 3D coordinate reconstruction from upper matching. The result is not clear compared to previous results.

VI. Discussions

In step 1, I seen the matched feature in vl_ubcmatch has shown some unreasonable matches. This may happened because the ubcmatch mainly detects the matches in pixel values [1]. Also, the matches mostly detects not only the target object but also the backgrounds of it. Hence I chosen image 03 and 04 pair although it did not showed the maximum number of matches. If we run RANSAC (algorithm 1) with 5-point algorithm the result has been compelling as figure 4. However, it mostly detects matches the floor or the nearby terrain compared to the target object. This may due to consistent feature of the floor (e.g. floor pattern, color) was more favorable to algorithm besides of the feature of the target. Also, in RANSAC I observed some interesting characteristic that if I set the threshold a little bit higher (e.g. 5×10^{-3}) then the overall result from triangulation showed the straight line or some unrecognizable object. Hence it is crucial that to select the threshold in moderate way. Lastly, the sfm result varies although I set the same preset for RANSEC and triangulation. This happens due to the random process during 5 point algorithm[3]. Hence there is a necessity to implement multi-view (i.e. growing step) to obtain stable results.

VII. Conclusion

Through this lab, we have seen various methods such as SIFT, matching scheme (vl_ubcmatch), RANSAC using 5-point algorithm, and triangulation methods. It turned out that these overall workflow is actually valid way to determine the 3D construction of specific images. However, this method is hard to implement in general way because the result is not stable as we run this scheme repeatedly. In addition, the result is very affected to the threshold that are used during the RANSAC, thus it is necessary to implement the more stable method such as multi-view imaging techinques.

References

[1] VLFeat authors. 2007. *vl_ucbmatch explanation*. VLFeat.org. https://www.vlfeat.org/matlab/vl_ubcmatch.html

[2] Richard Szeliski. 2022. *Computer Vision: Algorithms and Applications.*. Springer. (2022), https://szeliski.org/Book/

[3] Hartley, R. I. and Zisserman, A. 2004. *Multiple View Geometry in Computer Vision*. Cambridge University Press. (2004), Second Edition. ISBN: 0521540518