NCTU CV HW4: Structure from Motion

tags: `computer vision`

資科工碩　309551178　秦紫頤
電資學士班　610021　鄭伯俞
清華　H092505　翁紹育

Introduction

Structure from motion (SfM) is a photogrammetric range imaging technique for estimating three-dimensional structures from two-dimensional image sequences coupled with local motion signals. It studied in the fields of computer vision and visual perception. In biological vision, SfM refers to the phenomenon by which humans (and other living creatures) can recover 3D structure from the projected 2D (retinal) motion field of a moving object or scene.

Implementation Procedure

Read intrinsic matrix

For the two image given from the TA, we have gotten the intrinsic matrix of "Mesona" online and "Statue" given by TA. For our own data "books" we use the same camera to take 6 pictures of the checkerboards and do camera calibration to get the intrinsic matrix.

Camera calibration

We use six images to do the camera calibration. We consider the difference between vertical images and horizontal images. If the image direction does not align, the result might not be right, so we unified the image direction to vertical. If we detect the image is horizontal (`image.shape[1]-image.shape[0]>=0`), then I will rotate the image 90 degrees to vertical. Passing in the chessboard image shot by the camera without autofocus, `cv2.findChessboardCorners()` will return corners (corner points on image plane). The object points is defined by (0,0), (0,1)... (7,7). Now that we have object points and image points, we can start to calibrate the camera. We use `cv2.calibrateCamera()` to get the intrinsic matrix of the camera.

Find correspondence

Detect interest points

We use opencv to detect our interest points. First, we define feature type to **sift** by `cv2.SIFT_create()`. Then we detect our interest points and the sift descriptors by `detectAndCompute()`

Get the initial matches

First, we calculate the euclidean distance between all the features in the first image and the second image. Then, for each feature descriptor in the first image, we choose the top two matching feature descriptors in the second image (by min distance). Then we save the distance, image1 feature descriptor index, and image2 feature descriptor index in the data structure we create.

Get the good matches

We use a ratio test to decide suitable matches. For all the matches (original each image1 feature descriptors matches two feature descriptors in the second image), we need to decide whether they are suitable matches by distance; that is, the distance of the first match must be smaller than the second match times a threshold. We need to know whether the first match is significant enough (there is only one match on both images). The figure below explains the reason why we need this ratio test.

In the figure, the top two matches for f1 are f2 and f2'. However, they all look too similar, so we will decide to throw away this feature.

Build the correspondence list and draw matching

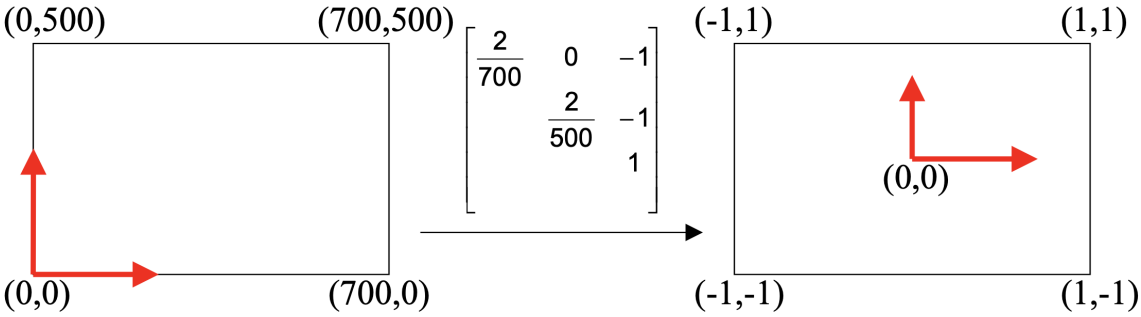
Since we do not need descriptors for the rest of the homework, we decide to save only the matching points. ex:[x1,y1,x2,y2] (this is the correspondence list, (x1,y1) is the coordinate of the feature points in image1 and (x2,y2) is the coordinate of the feature points in image2). Before moving on to the next step, we draw the feature matching results. First, we concatenate the two images so that they lie side by side. Then, we draw a circle on the corresponding feature points of image1 and image2 and draw a line between them to show that they match. For each corresponding feature, we use the same color to differentiate from the other points. We choose the color randomly for each correspondence.

Estimate fundamental matrix

In this step, we use the RANSAC algorithm to estimate the fundamental matrix. We iterate for 1000 iterations; we randomly choose eight correspondence from the correspondence list above for each iteration. We run through the normalized 8 point algorithm to get the fundamental matrix. Then we use Sampson distance error to get the inliers which evaluate the goodness of the fundamental error. The most inlier corresponding fundamental matrix is our fundamental matrix.

Get normalize x and x'

Before 8 points algorithm, we need to normalize the correspondence because orders of magnitude difference between columns of the data matrix, when using least square error yields poor results. So we normalize the point between [-1,1] and the center point in the middle of the image. The normalization is as the figure below.



In the implementation, we first split the correspondence into two vectors of points, one for points in image1 we called it x and another for points in image2 we called it x'. To normalize them:

$$x = Tx$$
$$x' = T'x'$$

Since the image size of image1 and image2 are the same so T=T'. T is as below:

```
# normalize x
x_mean = np.mean(x, axis=1)
S = np.sqrt(2) / np.std(x[:2])
T = np.array([[S, 0, -S * x_mean[0]], [0, S, -S * x_mean[1]], [0, 0, 1]])
x = T @ x
```

Eight point algorithm

We use eight point algorithm to find the fundamental algorithm. We first build the constraint matrix as below:

$$\begin{bmatrix} u_1u'_1 & u_1v'_1 & u_1 & v_1u'_1 & v_1v'_1 & v_1 & u'_1 & v'_1 & 1 \\ u_2u'_2 & u_2v'_2 & u_2 & v_2u'_2 & v_2v'_2 & v_2 & u'_2 & v'_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu'_n & u_nv'_n & u_n & v_nu'_n & v_nv'_n & v_n & u'_n & v'_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

(u,v) represents points in x and (u',v') represents points in x'. We use SVD and extract the fundamental matrix from the column of V corresponding to the smallest singular value. The resulting matrix may not satisfy the internal constraint of the fundamental matrix; that is, two of its singular values are equal to nonzero, and the other is zero. Depending on the application, smaller or larger deviations from the internal constraint may or may not be a problem. If the estimated matrix must satisfy the internal constraints, this can be accomplished by finding the matrix F' or rank two, which minimizes:

$$||F' - F_{est}||$$

F_est is the fundamental matrix solve by the constraint matrix. The solution to the problem is given by first computing a singular value decomposition of E_est:

$$E_{est} = USV^T$$

where U, V are orthogonal matrices, and S is a diagonal matrix that contains the singular values of E_est. In the ideal case, one of the diagonal elements of S should be zero, or at least small, compared to the other two, which should be equal. In any case, set:

$$S' = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

where s1, s2 are the largest and second-largest singular values in S, respectively. Finally, F' is given by:

$$F' = US'V^T$$

The matrix F' is the resulting estimate of the fundamental matrix provided by the algorithm. Since we have normalized the corresponding points to get the fundamental matrix, we need to denormalize it back by multiplying the normalizing matrix:

$$F = T^T F' T'$$

Sampson distance error

We use the Sampson distance error to evaluate the goodness of the fundamental matrix we found. The Sampson distance is the first-order approximation of geometric distance. Given a fundamental matrix F and a pair of correspondence (x,x') such that x'Fx=e, what is the distance/error of this pair of correspondence? The geometric distance is defined for all correspondence (y,y') such that y'Fy=0, the minimum value of ||x-y||^2+||x'-y'||^2 (in other words, the closest correspondence pair to (x,x') that satisfies the F matrix exactly). Intuitively, Sampson error can be roughly thought of as the squared distance between a point x to the corresponding epipolar line x'F. We get all the distance errors from all the correspondence. Then we set a threshold to count inliers. If the distance error is smaller than the threshold, we count it as inliers. We set the threshold to 0.025. Then the fundamental matrix corresponds to the most inliers is represented as the real fundamental matrix.

Draw epipolar line

We only use the inliers above to draw the epipolar line, not all the correspondence. We use get the epipolar lines by `cv2.computeCorrespondEpilines()`. We will draw image2's epipolar line on the image1. To show the inlier correspondence, we use `cv2.circle()` to draw on both images. Then we use `cv2.line()` to draw the lines found by `cv2.computeCorrespondEpilines()` on image1. Then we concatenate the output image1 and image2.

Find Essential Matrix

From the definition of Essential and Fundamental matrices, we have

$$\overrightarrow{x_1}^T F \overrightarrow{x_2} = \overrightarrow{x_1}^T K_1^{-T} E K_2^{-1} \overrightarrow{x_2}$$

so

$$E = K_1^T F K_2$$

we use the intrinsic matrix K1 and K2 to get E.

Find and Choose Camera Matrix

Next we need to decompose the essential matrix into extrinsic informaiotn of the camera.

For simplicity, we assume that the optical ceter of camera1 is the orgin of world coordinate, and the orientation of camera1 is described by identity matrix, so

$$P1 = \begin{bmatrix} I & | & \vec{0} \end{bmatrix}$$

Let $E = U L V^T$ be the SVD of E
There are four possible choice for P2

$$
\begin{aligned}
P_2 &= [UWV^T \quad | \quad +u_3] \\
P_2 &= [UWV^T \quad | \quad -u_3] \\
P_2 &= [UW^TV^T \quad | \quad -u_3] \\
P_2 &= [UW^TV^T \quad | \quad +u_3]
\end{aligned}
$$

In order to choose the right P2, for each possibility, we use first use triangulation procedure (discribed in next section) to find out the 3D position of key points, and see if they are located in the front of both cameras.

We use inner product to check if a point X is in front of a camera at position C and orientation R, the discriminating equation is as follows :

$$(X - C) \cdot R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} > 0$$

The P2 that makes most points in front of both camera is chosen.

Triangulation

Now we have camera matrix P1 and P2 (motion part), for each correspoding point in the two images with coordiante (u1,v1)and (u2,v2), we define a matrix A as
let

$$
\begin{aligned}
Q_1 &= K_1 P_1 \\
Q_2 &= K_2 P_2
\end{aligned}
$$

in

$$A = \begin{bmatrix} u_1 Q_{11}^T - Q_{13}^T \\ v_1 Q_{12}^T - Q_{13}^T \\ u_2 Q_{21}^T - Q_{23}^T \\ v_2 Q_{22}^T - Q_{23}^T \end{bmatrix}$$

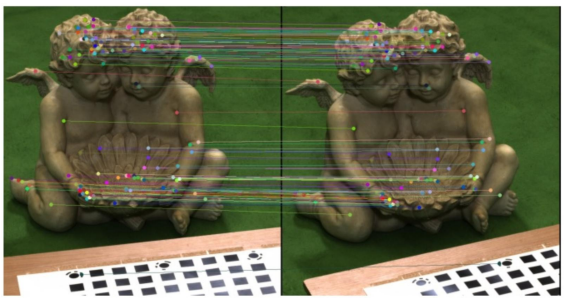
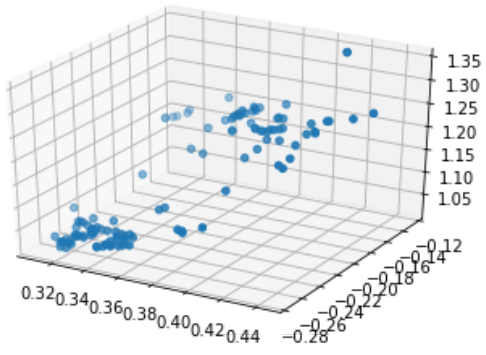
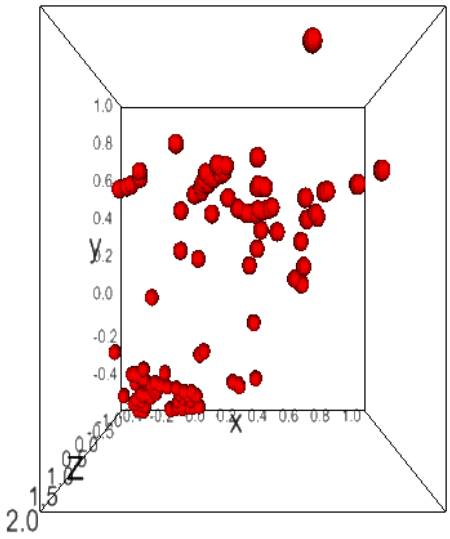
Then the 3D homogeneouse coordinate of that point is the basis of the null space of A, which we find with SVD.

Repeat this process for each correspondence , we get the location of all points in 3D (structure part).

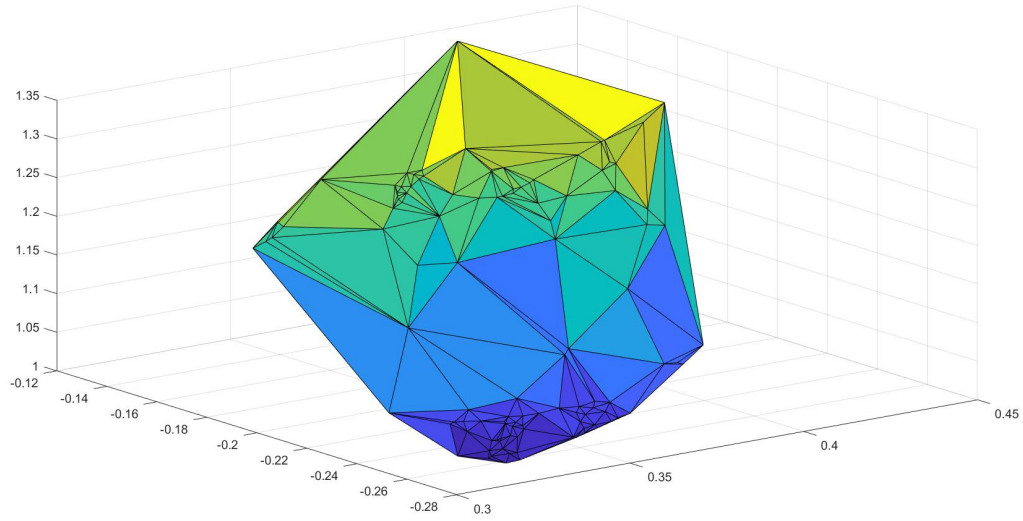
Experiment result

TA's data

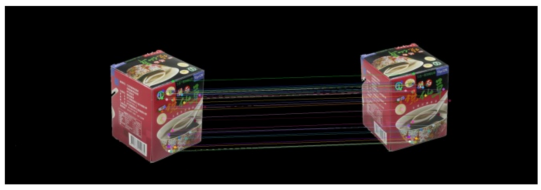
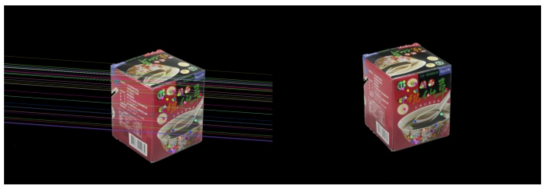
Statue

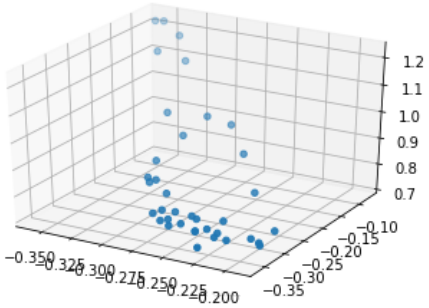
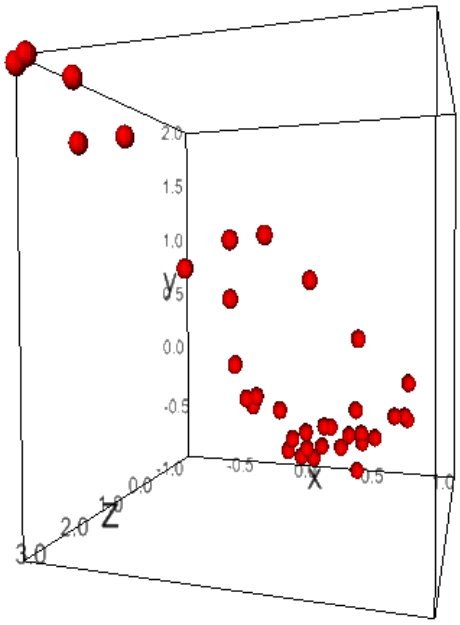
Key points matching	Epipolar line
	
3D view (I)	**3D view (II)**
	

Matlab result

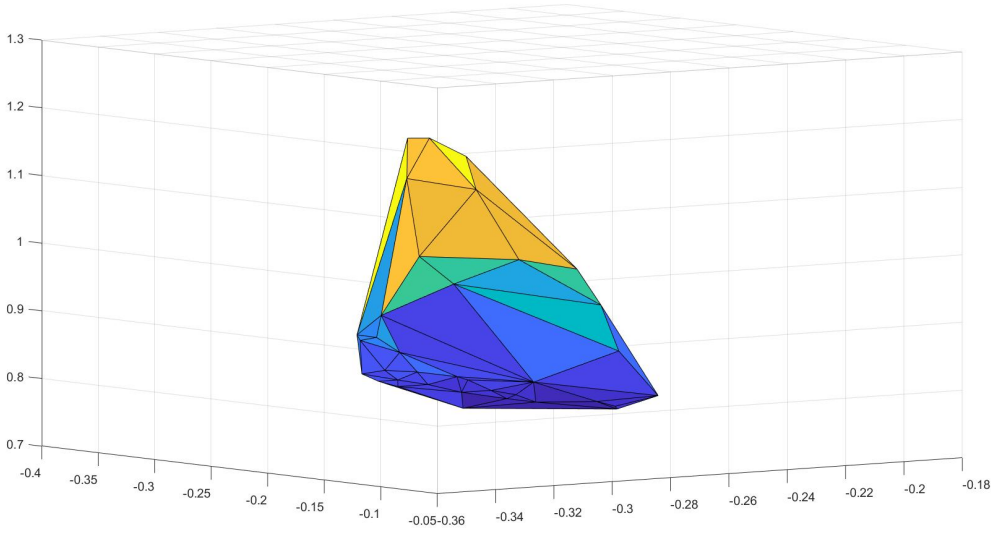


Mesona

Key points matching	Epipolar line
	

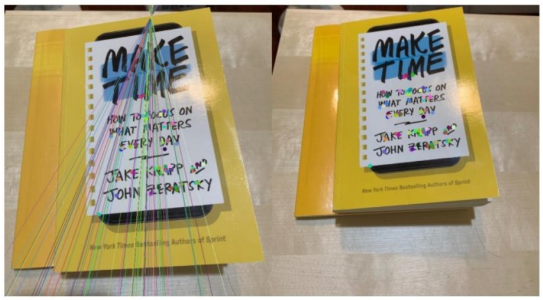
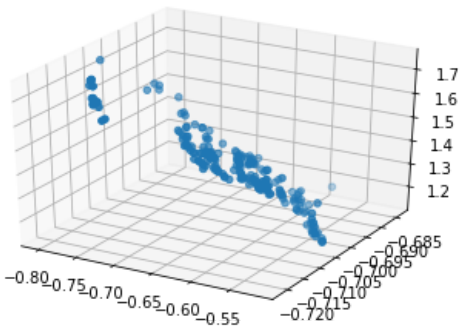
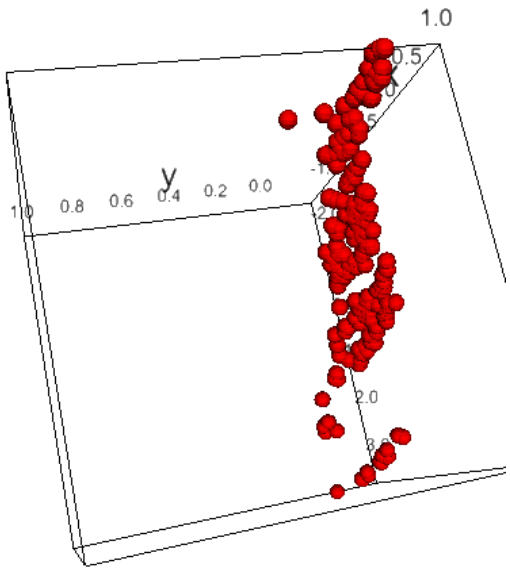
3D view (I)	3D view (II)
	

Matlab result

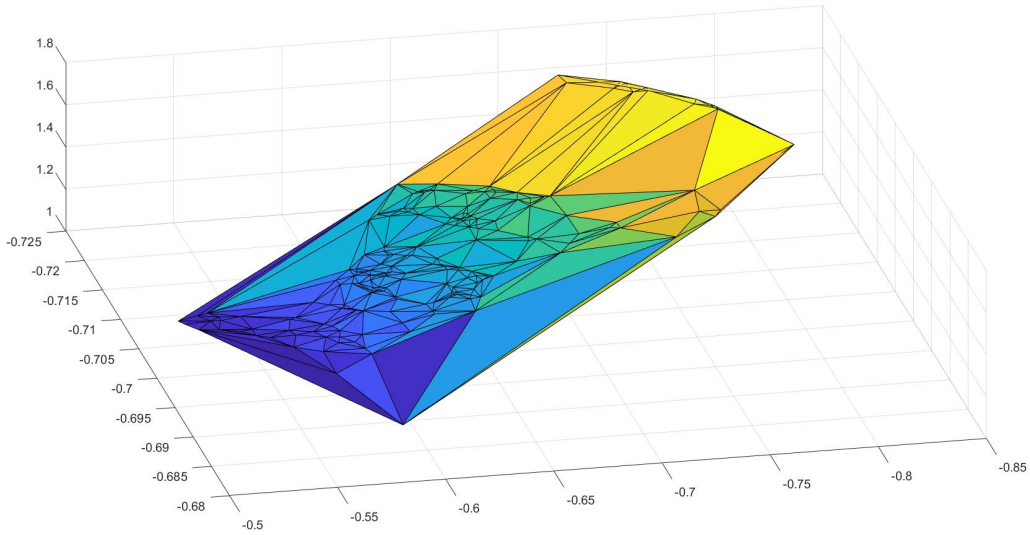


Our data

Books

Key points matching	Epipolar line
	
3D view (I)	**3D view (II)**
	

Matlab result



Discussion

It's critical to select the ratio of finding good match points and thresholds. In order to get enough points for reconstruction, we can not pick too small ratio. Yet, the ratio can not be too large for the reason that there may be too many noise. Among these data, our thresholds are around (0.3,0.6) and we get these values by trial and error.
For the 3D reconstruction results, some of them are not good. Because the structure of Books is quite simple, we only need to take two photos to get acceptable result. However, Statue and Mesona have more complicate structures, it's hardly reconstructed perfectly with only two images. The results above can only capture roughly shape.

Conclusion

In this homework, we fulfill SfM by combining what we've learned from the previous assignments. We go through camera calibration to get camera matrix, apply SIFT to find good feature points, and use RANSAC to get best fundamental matrix. Moreover, we induce epipolar line through 8 points algorithm and translate coorespondence from 2D to 3D by triangulation. It's quite challenge to go through so much steps, each minor error will lead to a huge difference to the results. However, we learn a lot from it.