CV HW4 -
Struture from Motion

Group 18

Introduction

Structure from Motion (**SfM**) is a photogrammetric range imaging technique for estimating three-dimensional structures from two-dimensional image sequences that may be coupled with local motion signals. It is 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.

There are several knowledge in SfM, “Camera Calibration”, “Feature Matching”, “Fundamental Matrix”, “Essential Matrix” and “Epipolar Geometry”. Some of them are implemented in HW1 and HW3, and the others are what we focus on in this assignment.

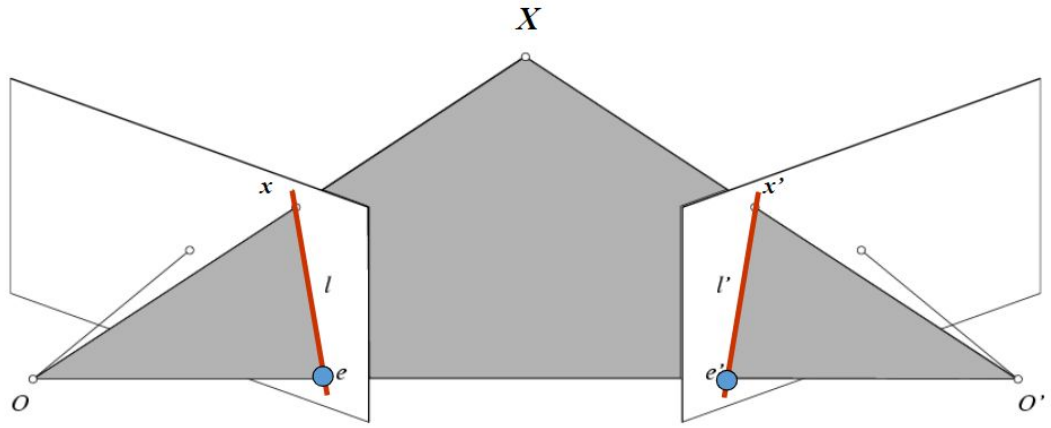
Implementation procedure

Step 1: Image loading

Step2: Using camera calibration with serval chess board images to get the intrinsic matrix (HW1)

Step 3: Feature extraction & matching with SIFT (HW3)

Step 4: Fundamental matrix calculation



There are two image, the left image corresponds to epipolar line l' in right image. The right image corresponds to epiploar line l in left image. Therefore, the fundamental matrix maps from a point in one image to a line in the other image. $l = F^T x'$ and $l' = Fx$

Because the x and x' are all correspond to the X center 3D point, the formula should be $x^T F x' = 0$.

To get fundamental matrix, we use 8-point algorithm, however, before we do that, we need to normalize our points to cancel the orders of magnitude difference. We transform our input image to [-1,1] x [-1,1].

First we define the normalization matrix T.

$$\tilde{x} = Tx \quad \tilde{x}' = T'x'$$

Because of the condition $x^T F x' = 0$, we can get the relation:

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

Now we can use 8-point algorithm and solve the F by svd.

$$\mathbf{Af} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = \mathbf{0}$$

After we got Af, we also need to consider the constrain **det(F) = 0**. To deal with the constrain, we use svd to solve F, note that $F = U * S * V$ and S[2] = 0.

Finally, we donormalize F.

$$F = T'^{\top} \tilde{F} T$$

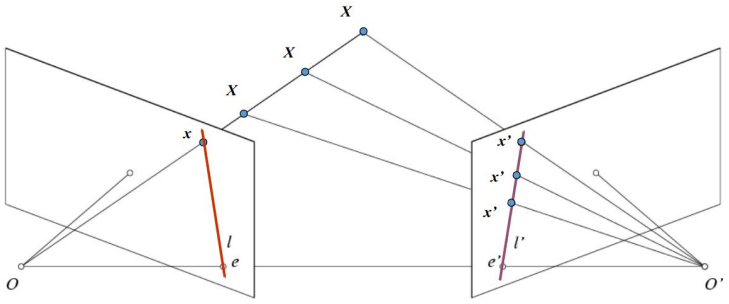
Step 5: RANSAC (RANdom SAmple Consensus)

To extract the good match points to calculate the fundamental matrix, we use ransac algorithm, which randomly chooses 8 data points for 8-point algorithm. We use Sampson error to make the approximation of geometric distance in two image. Finally, we can get our maximized inliers and best fitting F to draw epipolar lines and find essential matrix.
Sampson error:

$$e_i^2(\mathbf{F}) = \frac{(\underline{\mathbf{y}}_i^\top \mathbf{F}\underline{\mathbf{x}}_i)^2}{\|\mathbf{S}\mathbf{F}\underline{\mathbf{x}}_i\|^2 + \|\mathbf{S}\mathbf{F}^\top\underline{\mathbf{y}}_i\|^2} \qquad \mathbf{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Step 6: Draw epipolar lines

After we have the fundamental matrix and the inlier data points, we can get the epipolar lines, $l' = Fx$ and $l = F^T x'$, where x and x' are from inlier data.



According to OpenCV tutorials, we draw the epipolar lines and features on our image.

```
def drawlines(img1,img2,lines,pts1,pts2):
    ''' img1 - image on which we draw the epilines for the points in img2
        lines - corresponding epilines '''
    r,c = img1.shape
    img1 = cv.cvtColor(img1,cv.COLOR_GRAY2BGR)
    img2 = cv.cvtColor(img2,cv.COLOR_GRAY2BGR)
    for r,pt1,pt2 in zip(lines,pts1,pts2):
        color = tuple(np.random.randint(0,255,3).tolist())
        x0,y0 = map(int, [0, -r[2]/r[1] ])
        x1,y1 = map(int, [c, -(r[2]+r[0]*c)/r[1] ])
        img1 = cv.line(img1, (x0,y0), (x1,y1), color,1)
        img1 = cv.circle(img1,tuple(pt1),5,color,-1)
        img2 = cv.circle(img2,tuple(pt2),5,color,-1)
    return img1,img2
```

Step 7: Decompose essential matrix into [R|t]

In this step, we use the fundamental matrix which we got in previous steps to get the essential matrix E.

$$E = K_1^\top F K_2$$

We know essential matrix is parameterized by **R** and **t**, so we use svd to get 4 possible solutions of essential matrix.

$R1 = UW^TV^T$, $R2 = UWV^T$, $t1 = u3$, $t2 = -u3$

We assume first camera matrix as **P**: [I|0]

Second camera matrix **P** : [R1|t1] [R1|t2] [R2|t1] [R2|t2]

And then we can calculate the projection matrices (K[R | t]) of both cameras, so we can project the 2D data points to 3D coordinate.

Step 8: Triangulation

After getting all the 2D coordinates in both images, projection matrices of both cameras, we can reconstruct the 3D object. The method to reproject 2D points to 3D world is called Triangulation.

The triangulation linear solution is as following:

Triangulation

$$x = w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} \mathbf{p}_1^\top \\ \mathbf{p}_2^\top \\ \mathbf{p}_3^\top \end{bmatrix} \quad A = \begin{bmatrix} u\mathbf{p}_3^\top - \mathbf{p}_1^\top \\ v\mathbf{p}_3^\top - \mathbf{p}_2^\top \\ u'\mathbf{p}_3'^\top - \mathbf{p}_1'^\top \\ v'\mathbf{p}_3'^\top - \mathbf{p}_2'^\top \end{bmatrix}$$

$$x' = w\begin{bmatrix} u \\ v' \\ 1 \end{bmatrix}$$

Given P, P', x, x'

1. *precondition points and projection matrices*
2. *create matrix* A
3. $[U, S, V] = svd(A)$
4. $X = V(:, end)$

Step 9: Find the correct camera matrix

After getting four possible projection matrices, there is only one matrix that all 3D points are in front of 2 cameras. So we have to compute the forward vector of both cameras to find the best solution of 3D points reconstruction.

The rotation and transformation matrix we have got in the projection matrix is based on camera coordination. Since all the 3D points we have got is in world coordinate, so we have to change the camera center to the world coordinate and its forward vector.

- Camera Center

$$\begin{bmatrix} X_{\text{cam}} \\ Y_{\text{cam}} \\ Z_{\text{cam}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Leftrightarrow \mathbf{C} = \begin{bmatrix} X_{\text{world}} \\ Y_{\text{world}} \\ Z_{\text{world}} \end{bmatrix} = R^{\top} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - R^{\top} t = -R^{\top} t$$

- Camera Extrinsic [R|t]

$$\begin{bmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{bmatrix} = R \begin{bmatrix} X_{\text{world}} \\ Y_{\text{world}} \\ Z_{\text{world}} \end{bmatrix} + t \Leftrightarrow \begin{bmatrix} X_{\text{world}} \\ Y_{\text{world}} \\ Z_{\text{world}} \end{bmatrix} = R^{-1} \left(\begin{bmatrix} X_{\text{cam}} \\ Y_{\text{cam}} \\ Z_{\text{cam}} \end{bmatrix} - t \right) = R^{\top} \begin{bmatrix} X_{\text{cam}} \\ Y_{\text{cam}} \\ Z_{\text{cam}} \end{bmatrix} - R^{\top} t$$

- View Direction

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Leftrightarrow \left(R^{\top} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - R^{\top} t \right) - (\mathbf{C}) = \left(R(3,:)^{\top} - R^{\top} t \right) - \left(-R^{\top} t \right) = R(3,:)^{\top}$$

$* \text{ Just need to test } (X - \mathbf{C}) \cdot R(3,:)^{\top} > 0?$

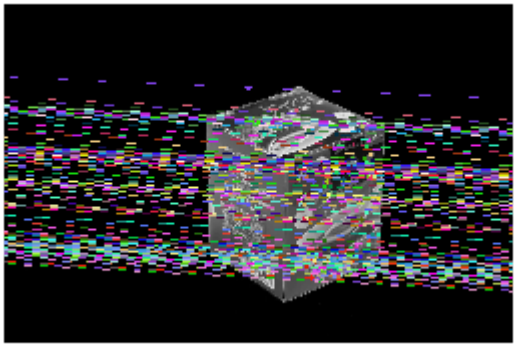
Step 10: Construct 3D model by Matlab

With 3D points in the world coordinate, we used matlab (TA provided code) to reconstruct the 3D model.

Experiment Result

1. Mesona

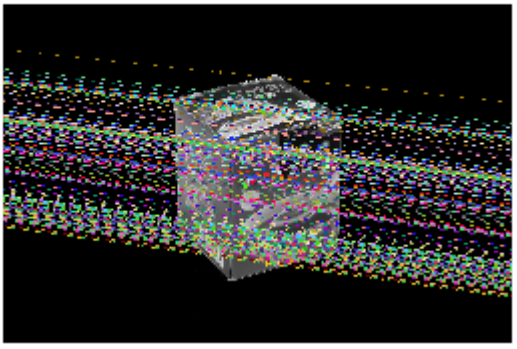
First Image (pts2 line)

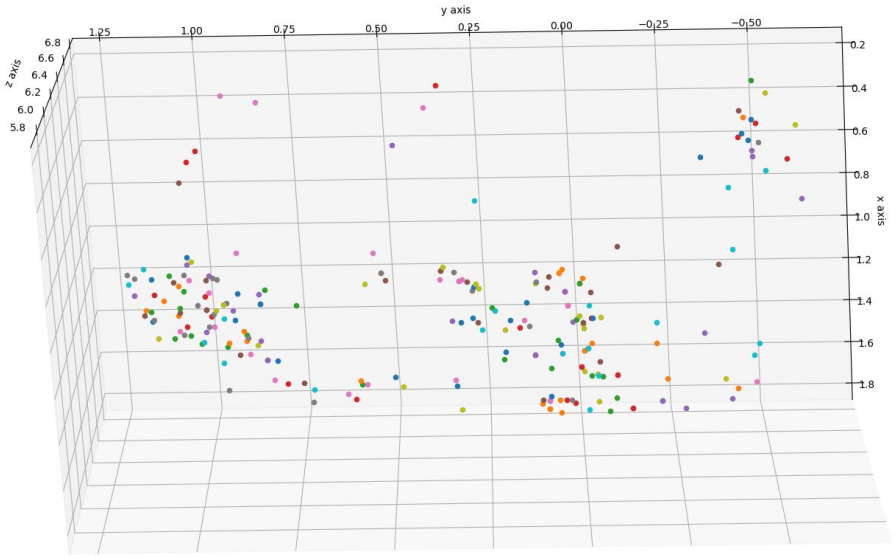


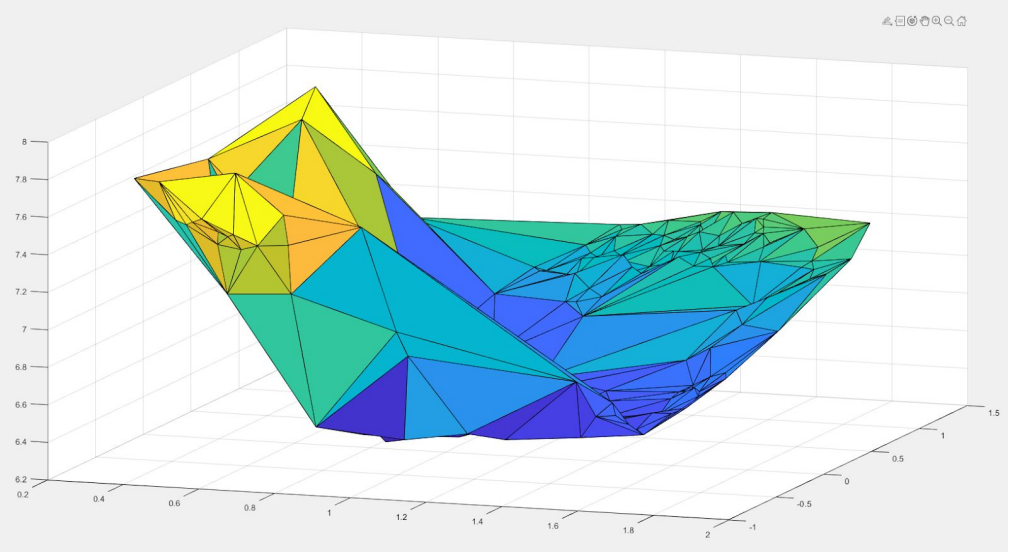
Second Image (pts2)

First Image (pts1)

Second Image (pts1 line)

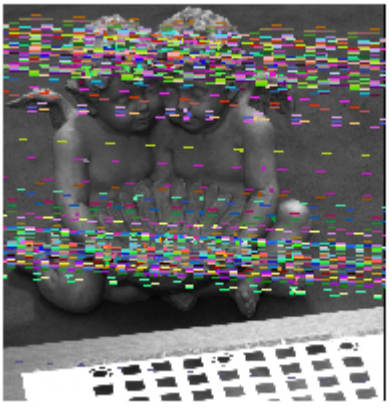




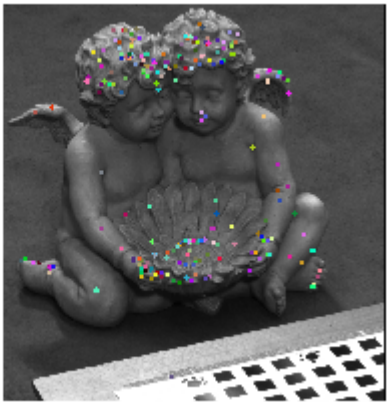


2. Statue

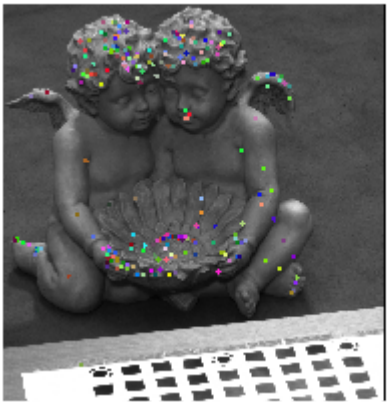
First Image (pts2 line)



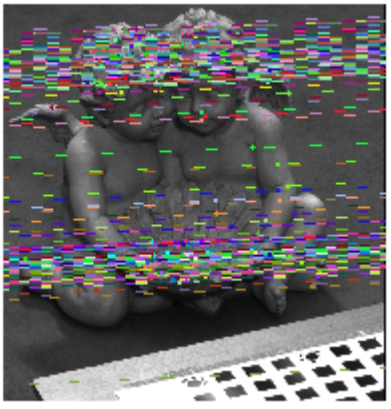
Second Image (pts2)

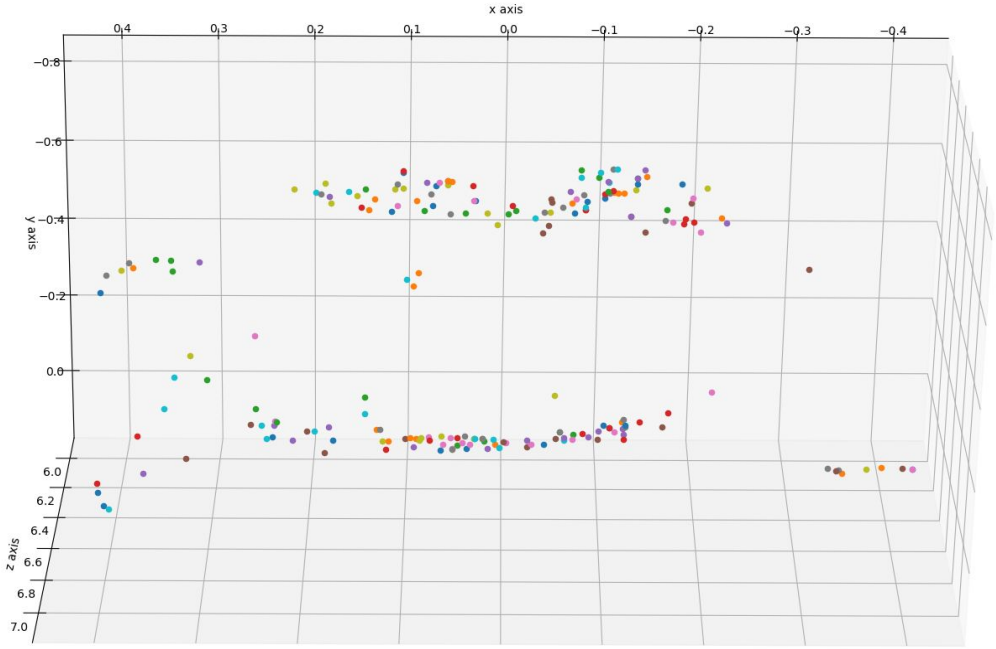


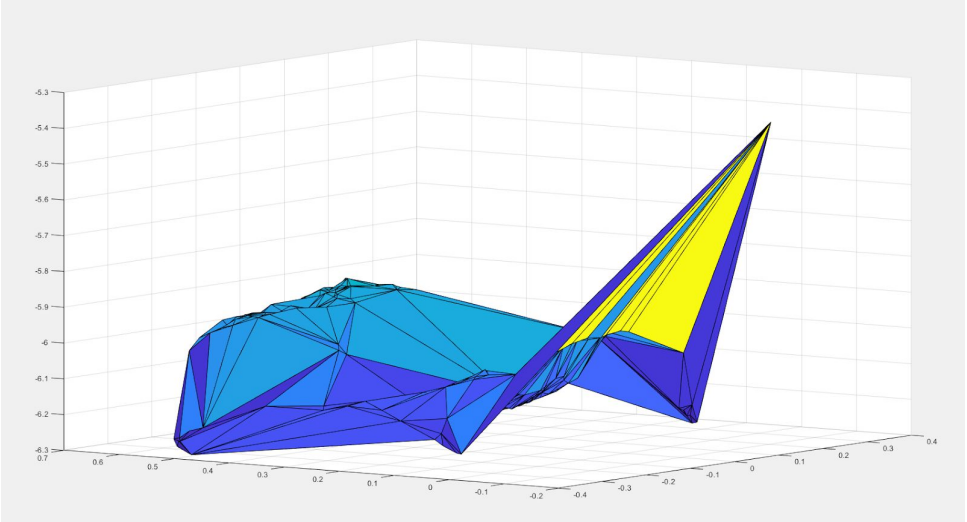
First Image (pts1)



Second Image (pts1 line)

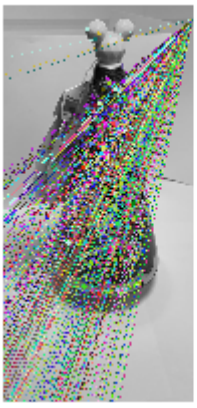



x axis
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
y axis
−0.8
−0.6
−0.4
−0.2
0.0
z axis
6.0
6.2
6.4
6.6
6.8
7.0

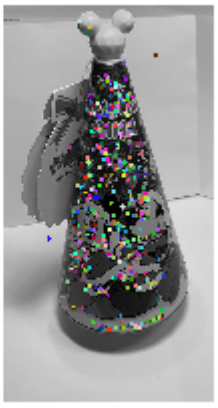


3. Disney (ours)

First Image (pts2 line)

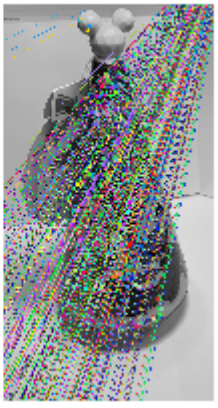


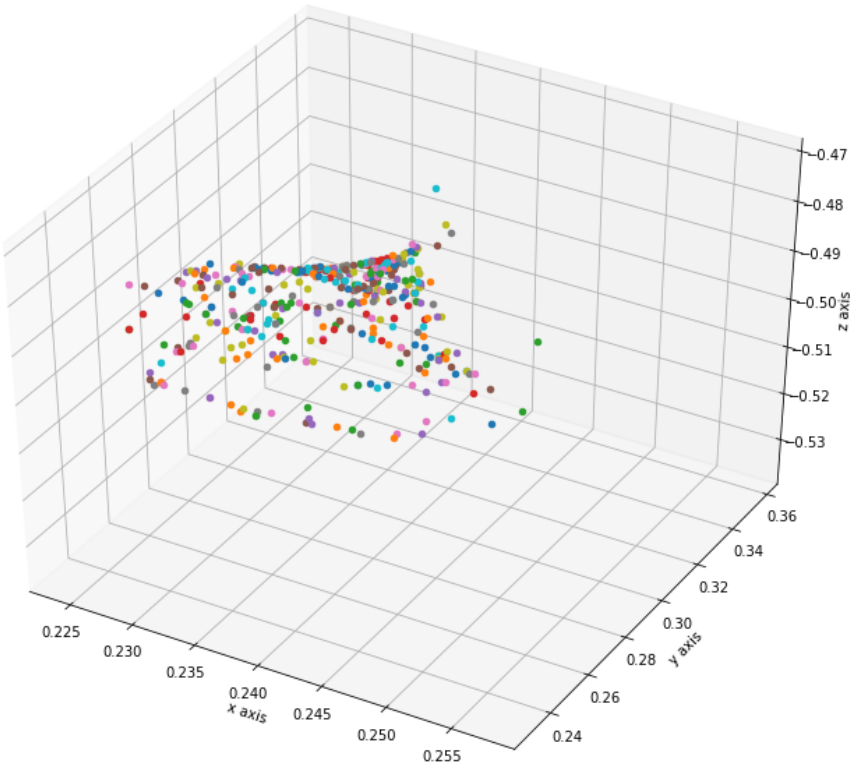
Second Image (pts2)

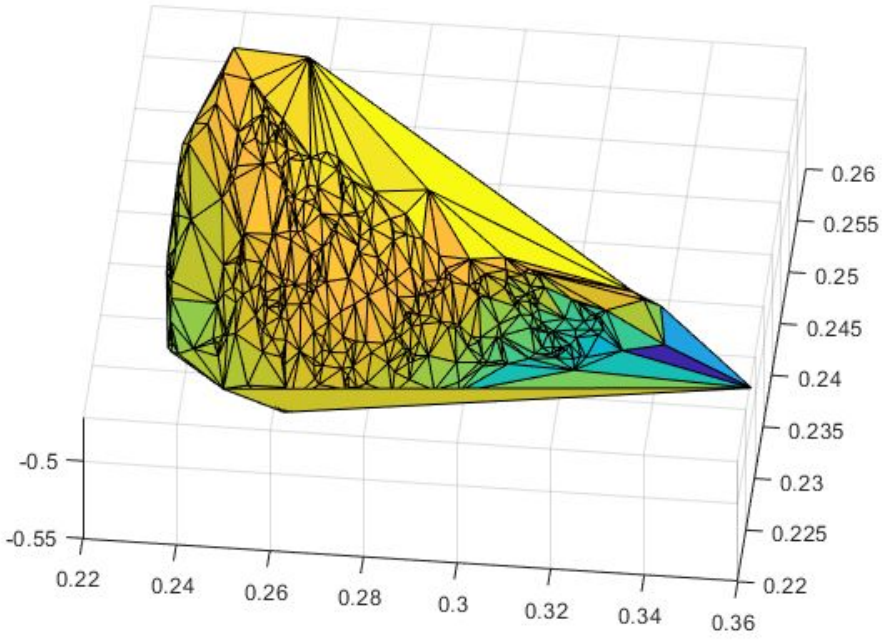


First Image (pts1)

Second Image (pts1 line)

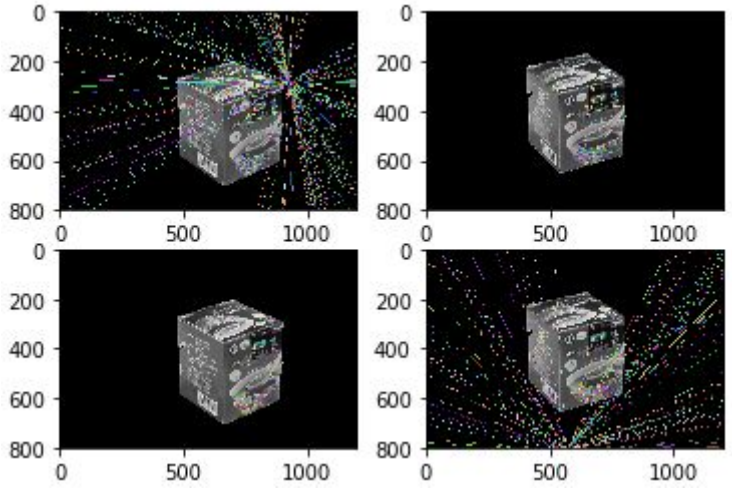



-0.47
-0.48
-0.49
-0.50
-0.51
-0.52
-0.53
z axis
0.36
0.34
0.32
0.30
0.28
0.26
0.24
y axis
0.225
0.230
0.235
0.240
0.245
0.250
0.255
x axis


0.26
0.255
0.25
0.245
0.24
0.235
0.23
0.225
0.22
-0.5
-0.55
0.22
0.24
0.26
0.28
0.3
0.32
0.34
0.36

Discussion

1. In the RANSAC step, Sampson error is used to help us to find F, however, if we use the error |x'Fx| < threshold, sometimes we will get the result below:



We cannot find very good inlier and F.

2. For the camera calibration of our own data, we use our code in HW1 to get the intrinsic matrix of our camera. When trying our own images to do SFM, the hardest part is to choose an object with enough feature points. If the feature point is too low, the RANSAC part will also be hard to do, so we actually tried a lot of different objects. When testing our own data, the threshold and points number need to be adjust, too.

Conclusion

In this homework, we implement Structure from Motion(SfM). We use SIFT to find the feature points, and then ration distance is used for extracting good match points. We implement 8-point algorithm and RANSAC to get fundamental matrix. After we found the fundamental matrix, we further estimate the essential matrix and then use SVD to decompose essential matrix to 4 possible extrinsic matrixes of the second camera. Finally, we perform Triangulation to transform 2D point to 3D coordinate. Then we choose the right extrinsic matrix with most 3D points in front the camera and we can build the 3D model form input images.

Work assignment plan between team members

We finish this homework together.

Additional references

1. Epipolar Geometry