CMPT 412

Project 5

3D Reconstruction

Instructor : Yasutaka Furukawa

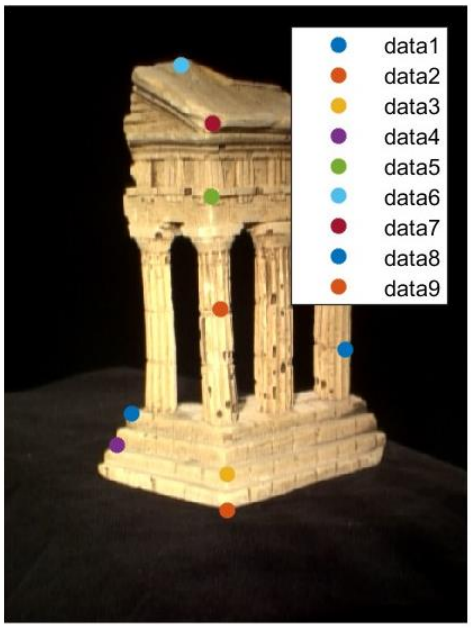
Name: Kaikun Fang

Student ID: 301416542

3.1.1 Implement the eight point algorithm:

My recovered F:

```
>> test_eightpoint

F =

    0.0000   -0.0000   -0.0000
   -0.0000   -0.0000    0.0005
    0.0000   -0.0005   -0.0018
```



Select a point in this image
(Right-click when finished)

Verify that the corresponding point
is on the epipolar line in this image

3.1.2 Find epipolar correspondences:

- The similarity metric I use is the Singular value of the grayscale values calculated by "`norm(abs(im1_window-im2_window))`".

  Step 1: Both images are processed with "rgb2gray".

  Step 2: In the first image, use the point of pts1 as the center, 21*21 as the window size, and take out the grayscale value as Matrix1.

  Step 3: Find all possible corresponding points on the epipolar lines.

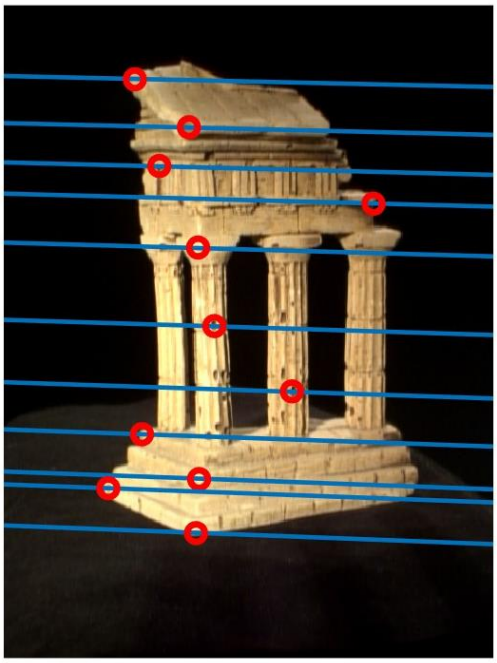
  Step 4: In the second picture take the corresponding point as the center, 21*21 as the size

of the window, and take out the gray value as the corresponding matrix. The number of correspondence matrices should be the same as the number of possible correspondence points.

Step 5: Calculate the singular value(`norm(abs(im1_window-im2_window))`) of Matrix1 with each corresponding matrix in turn, and the corresponding point of the corresponding matrix with the smallest singular value is the correct corresponding point.

- My algorithm is likely to have wrong matching points on the columns of the temples. This should be because when epipolar lines pass through multiple columns there will be many localized very similar points, plus the black background between the columns causes interference, resulting in incorrect matching of corresponding points.

Select a point in this image
(Right-click when finished)



Verify that the corresponding point
is on the epipolar line in this image

3.1.3 Write a function to compute the essential matrix:

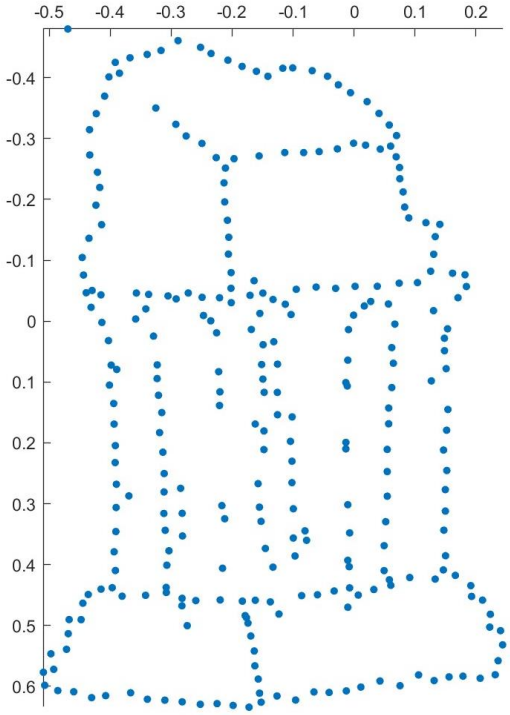
```
>> test_essentialMatrix

E =

    0.0044   -0.0400   -0.0207
   -0.1543   -0.0009    0.7253
   -0.0002   -0.7343   -0.0013
```

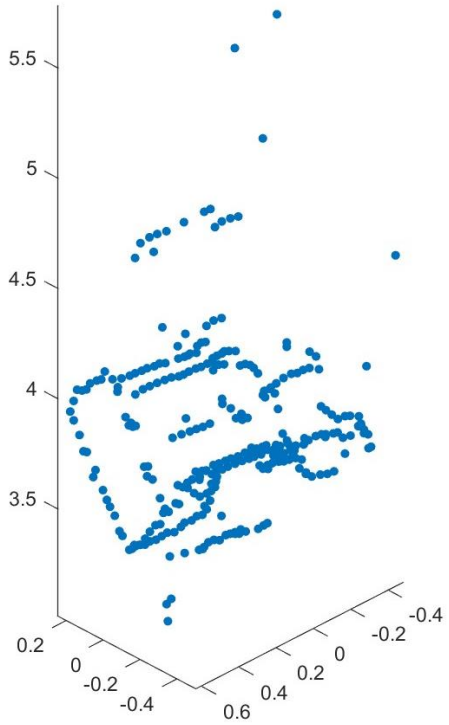
3.1.4 Implement triangulation:

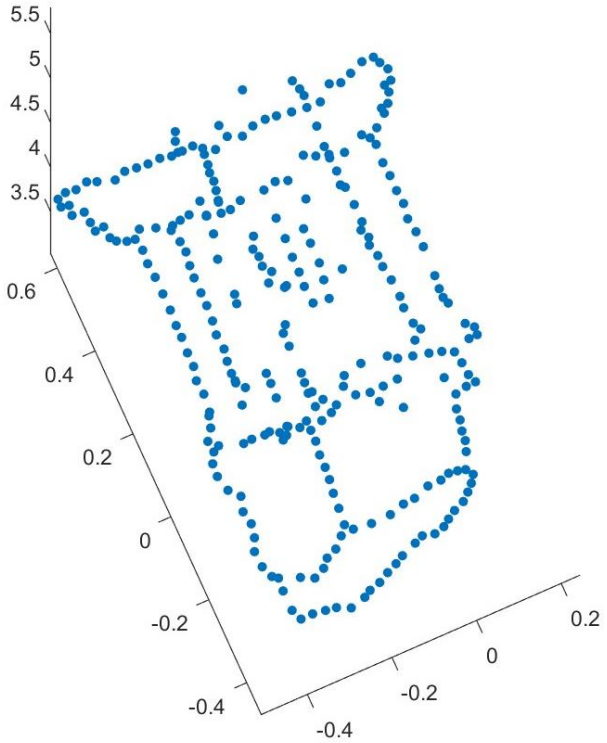
The way I find the correct extrinsic matrices is to calculate the coordinates obtained for each extrinsic matrices in turn, then check the coordinates of all the points in each group, and if the z-axis coordinate of any point is negative, the extrinsic matrices are proven to be wrong. Only when the z-axis coordinates of all points in the group are positive, the extrinsic matrices corresponding to this group are correct.

My re-projection error using pts1, pts2 from someCorresp.mat is 0.5547 for pts1, 0.5585 for pts2.(Use *test_triangulate.m* to get the results.)

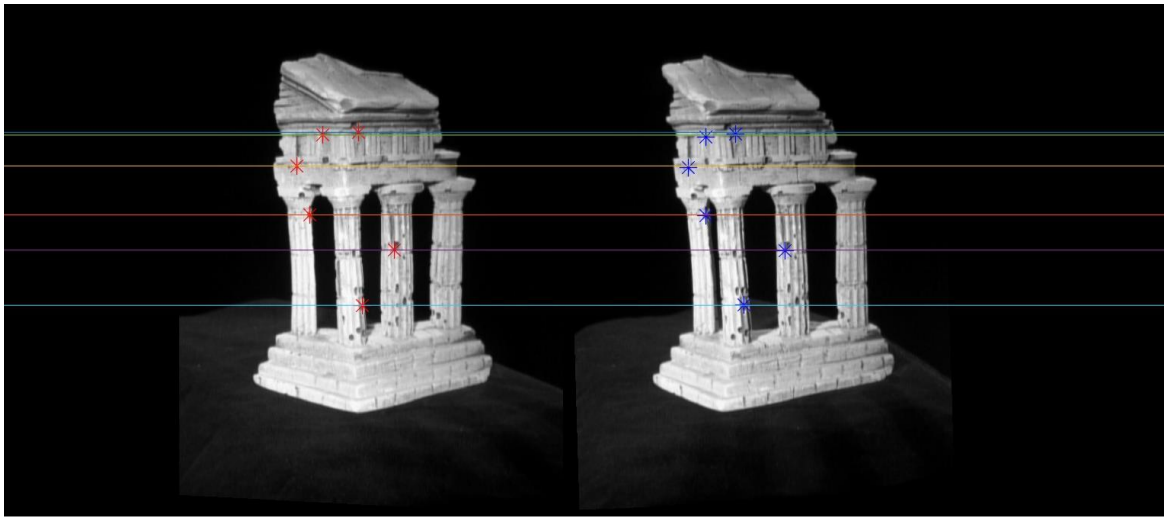
3.1.5 Write a test script that uses templeCoords:


-0.5 -0.4 -0.3 -0.2 -0.1 0 0.1 0.2
-0.4 -0.3 -0.2 -0.1 0 0.1 0.2 0.3 0.4 0.5 0.6

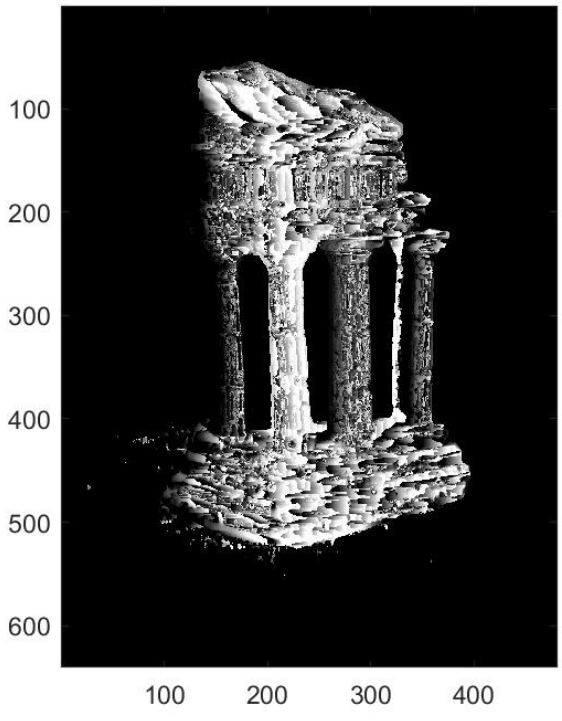

5.5 5 4.5 4 3.5
0.2 0 -0.2 -0.4
-0.4 -0.2 0 0.2 0.4 0.6


5.5 5 4.5 4 3.5
0.6 0.4 0.2 0 -0.2 -0.4
0.2 0 -0.2 -0.4

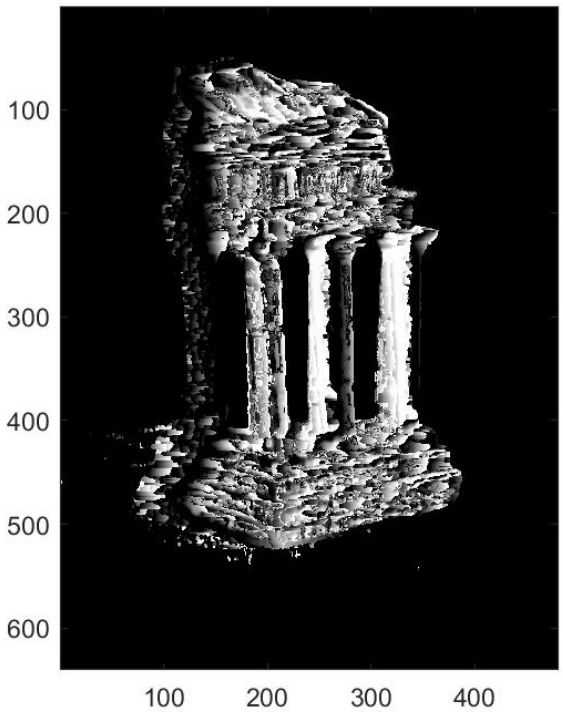
3.2.1 Image rectification:



3.2.2 Dense window matching to find per pixel density:

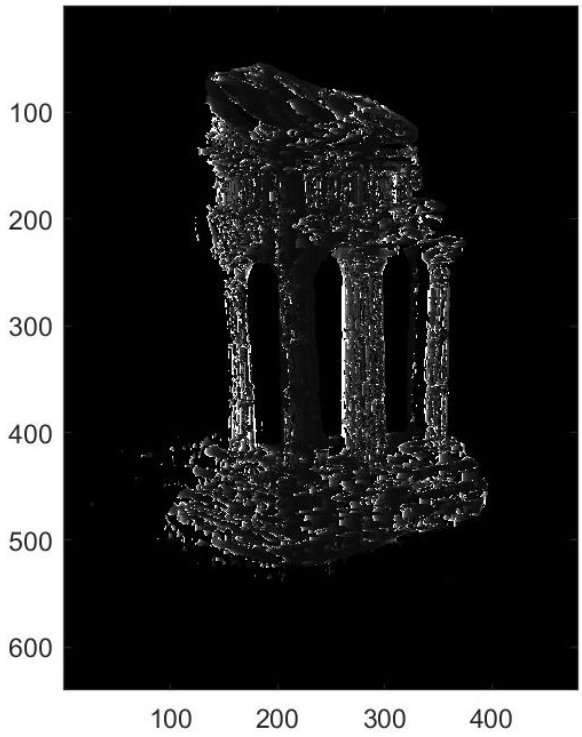


Disparity map (without rectification)

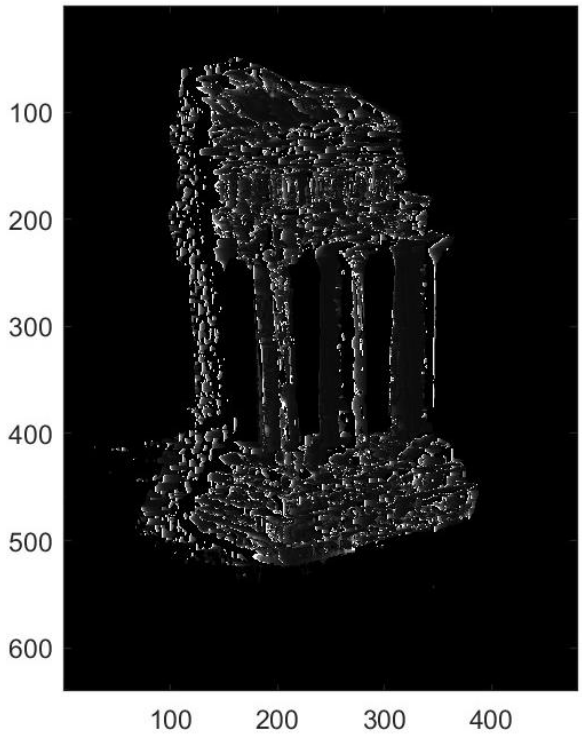


Disparity map (with rectification)

3.2.3 Depth map:



Depth map (without rectification)



Depth map (with rectification)

3.3.1 Estimate camera matrix P:

```
>> testPose
Reprojected Error with clean 2D points is 0.0000
Pose Error with clean 2D points is 0.0000
------------------------------
Reprojected Error with noisy 2D points is 2.4183
Pose Error with noisy 2D points is 0.0849
```

3.3.2 Estimate intrinsic/extrinsic parameters:

```
>> testKRt
Intrinsic Error with clean 2D points is 0.0000
Rotation Error with clean 2D points is 0.0000
Translation Error with clean 2D points is 0.0000
------------------------------
Intrinsic Error with noisy 2D points is 0.3741
Rotation Error with noisy 2D points is 0.0318
Translation Error with noisy 2D points is 0.0295
```