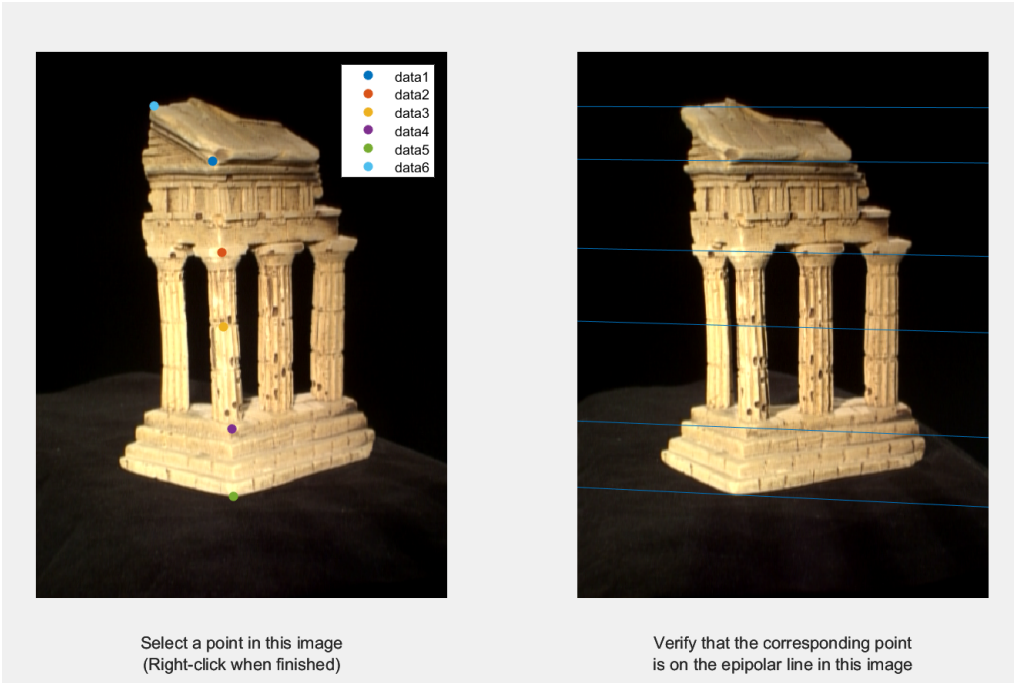
CMPT412 Project 5
3D Reconstruction

Student: Sibei Zhou(301416917)
Computer ID: sibeiz@sfu.ca

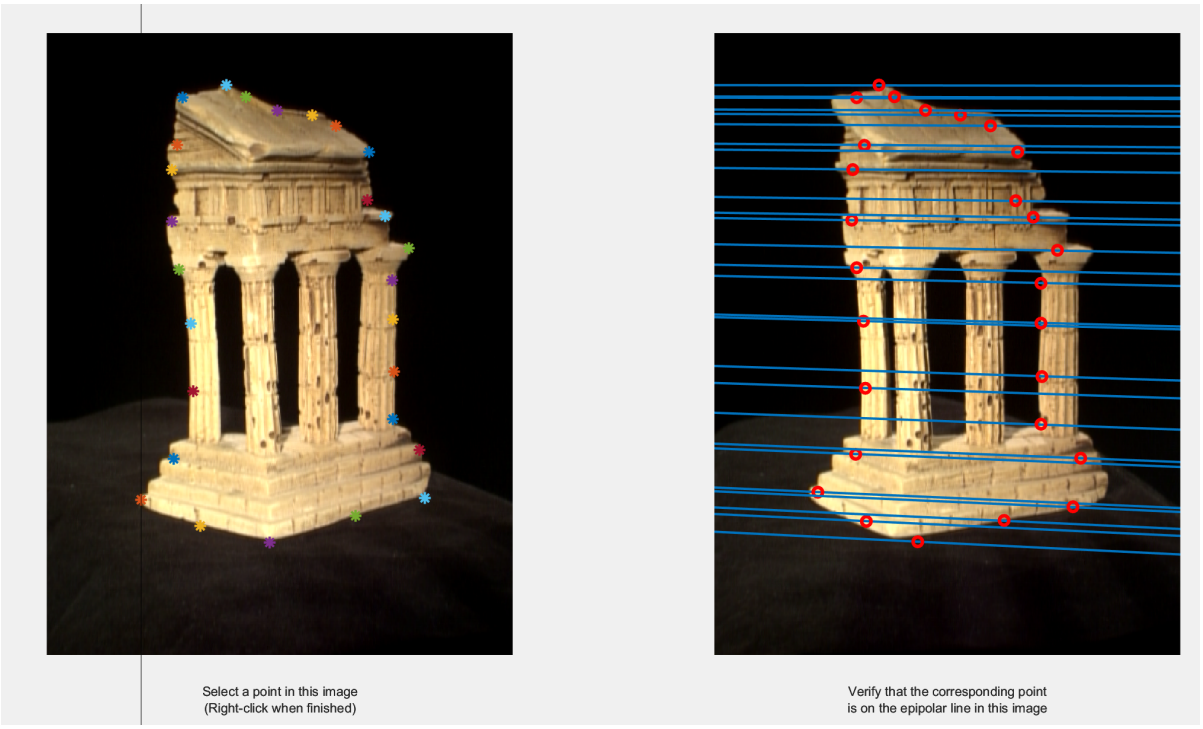
- **3.1 Sparse reconstruction**
  - 3.1.1 Implement the eight point algorithm (2 pts)
    Using test1_1.m to test the eight point algorithm and show the recovered fundamental matrix and visualize the epipolar lines:

    ```
    >> test1_1

    F =

       0.0000   -0.0000    0.0000
      -0.0000   -0.0000    0.0004
       0.0000   -0.0004   -0.0017
    ```



  - 3.1.2 Find epipolar correspondences (2 pts)
    Using test1_2.m to test the epipolarCorrespondence.m which use epipolarMatchGui.m to visually test the function:

    The screenshot of epipolarMatchGui with the point-pairs:



The similarity metric here is calculated using Euclidean distance based on the difference of RGB values of different small windows of size w (I use w = 5 here) around the point x.

And the obvious mismatches happen at the edges of the columns of the right side, one possible reason I guess might be the other edge of the column was within the threshold and had similar neighboring pixels (quite a lot of black pixels from the background) and lead to a similar similarity metric.

- 3.1.3 Write a function to compute the essential matrix (2 pts)
  Using test1_3.m to get the estimated E matrix for the temple image pair:

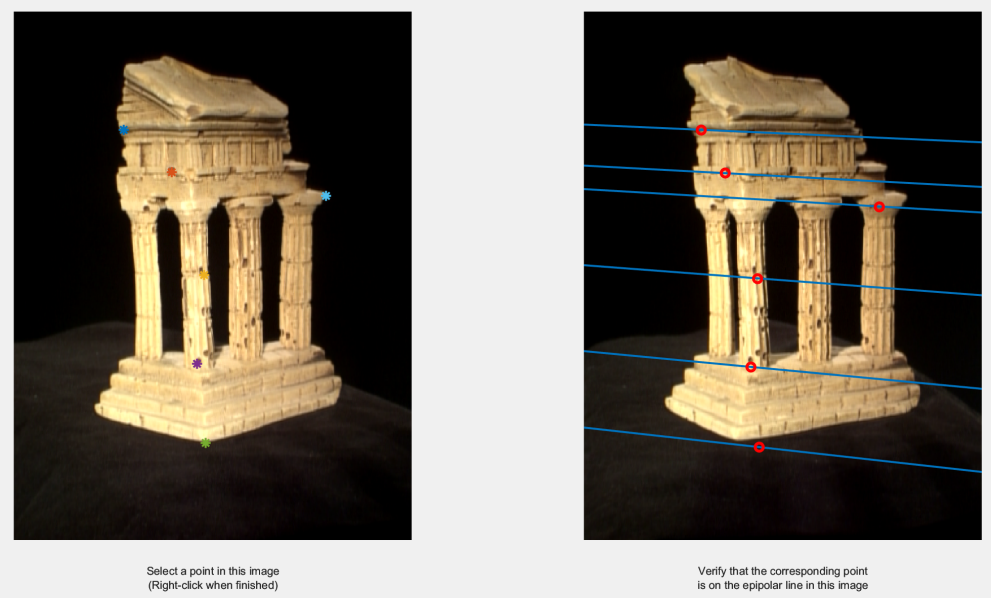
  >> test1_3

  E =

  1.0e-03 *

  -0.0000 0.0001 0.0142
  0.0001 0.0000 -0.6902
  0.0023 0.6866 -0.1131

And the screenshot of epipolarMatchGui(I1, I2, E):

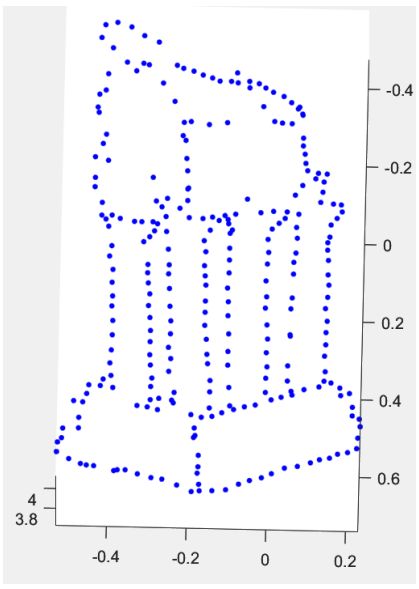


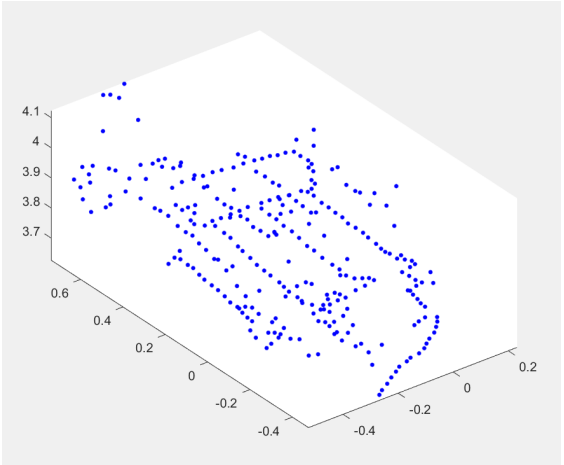
- 3.1.4 Implement triangulation (2 pts)
  Describe how you determined which extrinsic matrices are correct:
  The correct extrinsic matrix should have all values in the third dimension greater than 0, and has the least re-projection error.

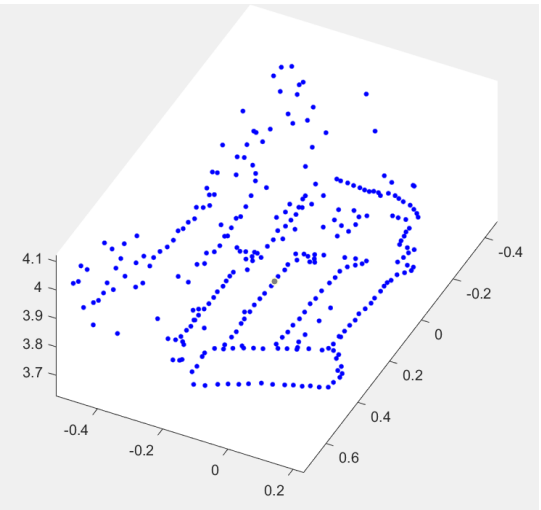
  Using test1_4.m to check the performance by looking at the re-projection error using pts1, pts2 from someCorresp.mat:

  >> test1_4
  Re-projection error of pts1 = 0.1392
  Re-projection error of pts1 = 0.1370

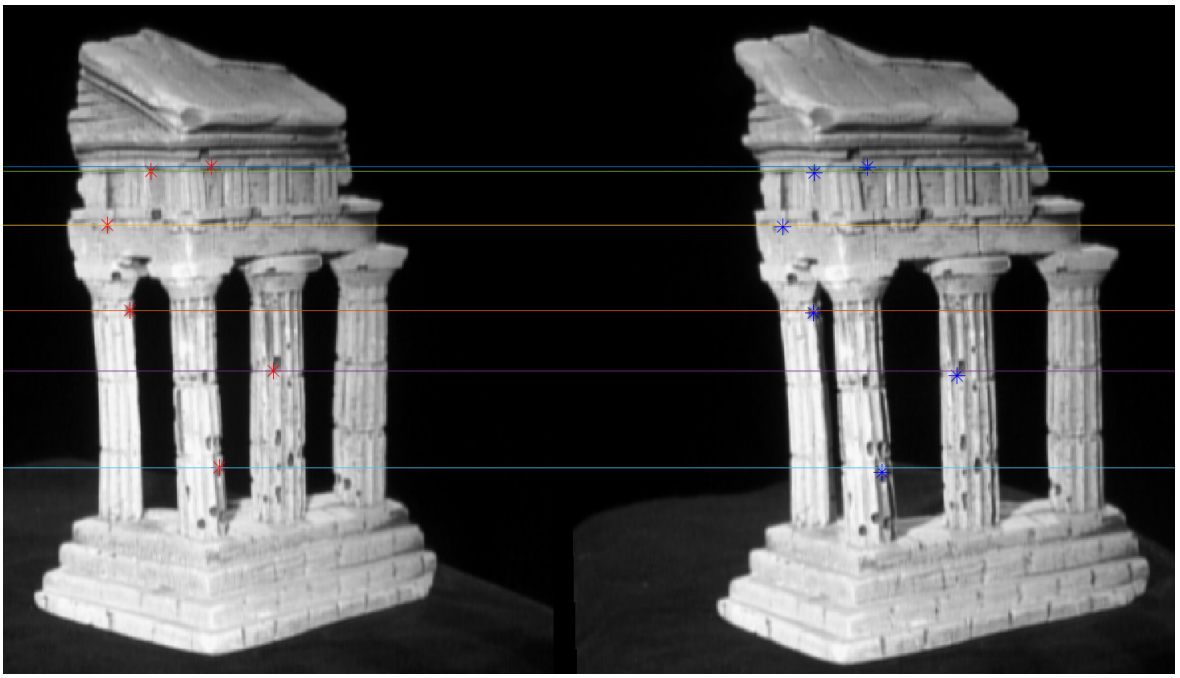
- 3.1.5 Write a test script that uses templeCoords (2 pts)
  3 images of your final reconstruction of the templeCoords points, from different angles:





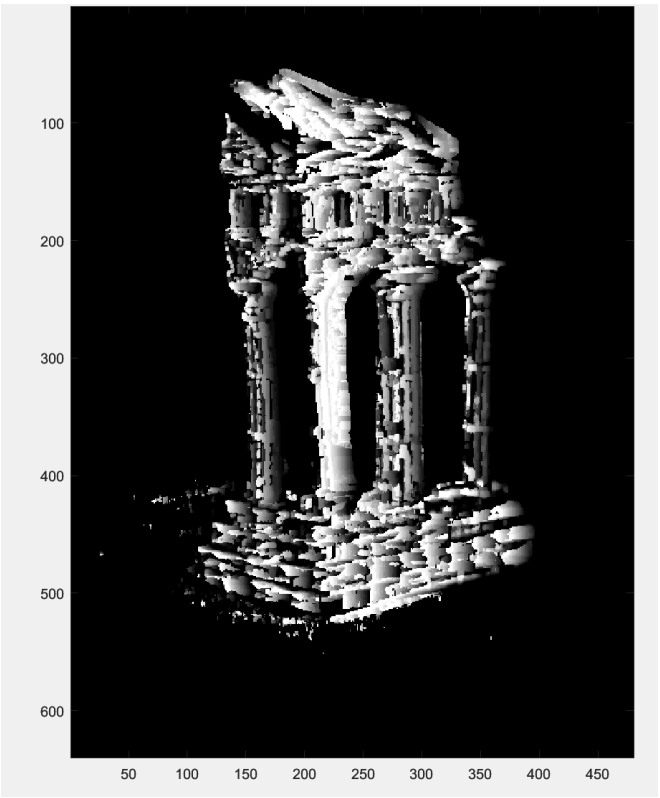


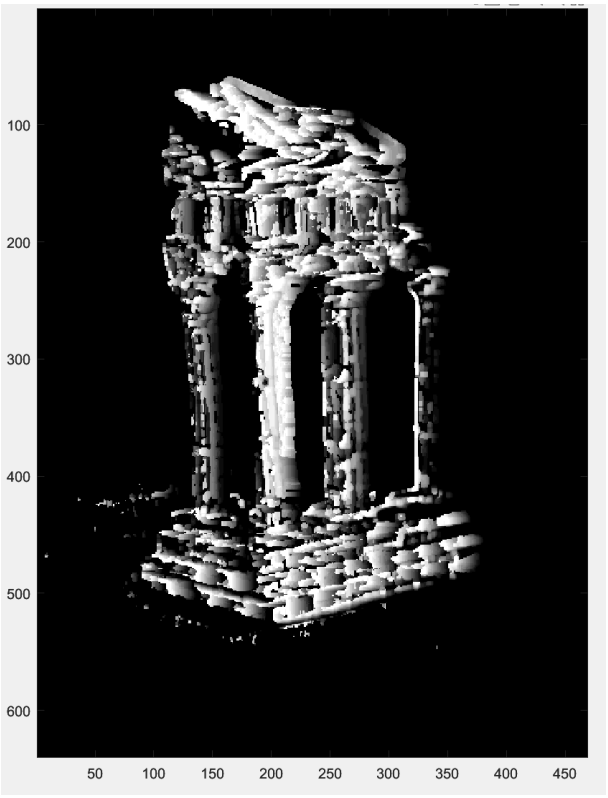
- **3.2 Dense reconstruction**
  - 3.2.1 Image rectification (2 pts)
    Run testRectify.m to test rectify_pair.m:



  - 3.2.2 Dense window matching to find per pixel density (2 pts)

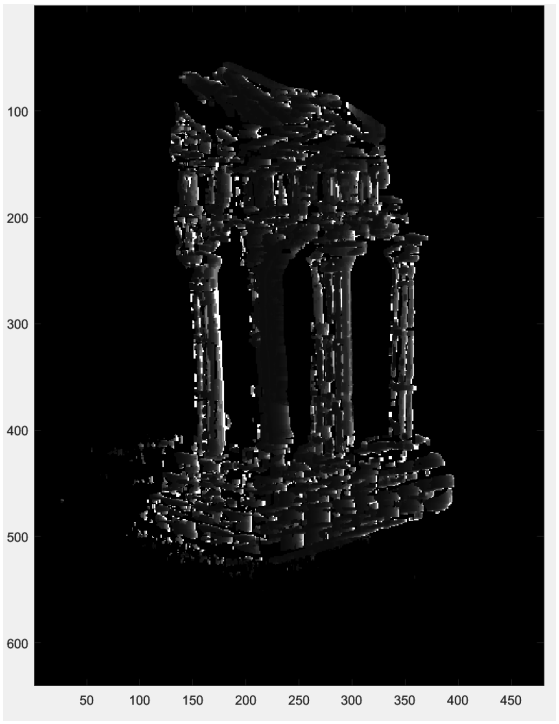
Without rectification (left) and with rectification(right):

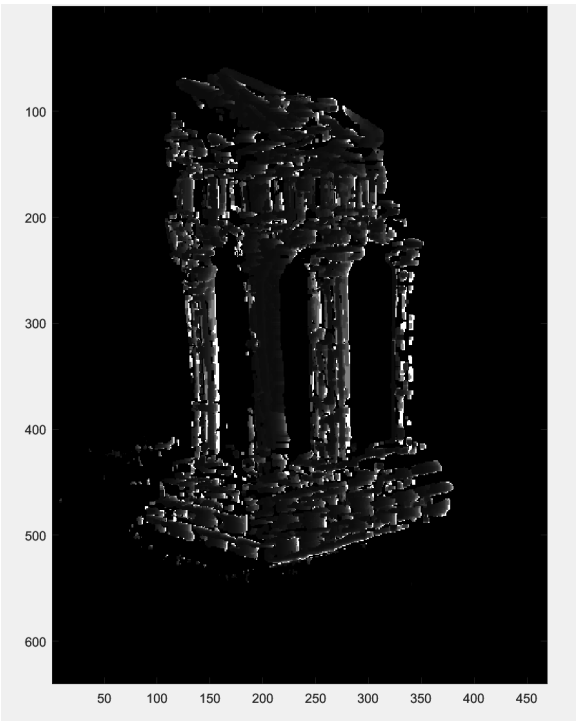




- 3.2.3 Depth map (2 pts)

  Without rectification (left) and with rectification(right):





- **3.3 Pose estimation**
  - 3.3.1 Estimate camera matrix P (2 pts)

    Reprojected errors from reprojection of clean and noisy data:

    >> 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 2.5279
    Pose Error with noisy 2D points is 0.3165

  - 3.3.2 Estimate intrinsic/extrinsic parameters (1 pts)

    From the part of the QR decomposition I refer the first post of the reference (https://math.stackexchange.com/questions/1640695/rq-decomposition), which is:

    Consider the following algorithm:

    i.) Compute $\tilde{A} := PA$ (i.e. reverse rows of A)

    ii.) Compute decomposition of $\tilde{A}^T = \tilde{Q}\tilde{R}$

    iii.) Set $Q := P\tilde{Q}^T$ (i.e. reverse rows of $\tilde{Q}^T$, note that Q is orthogonal)

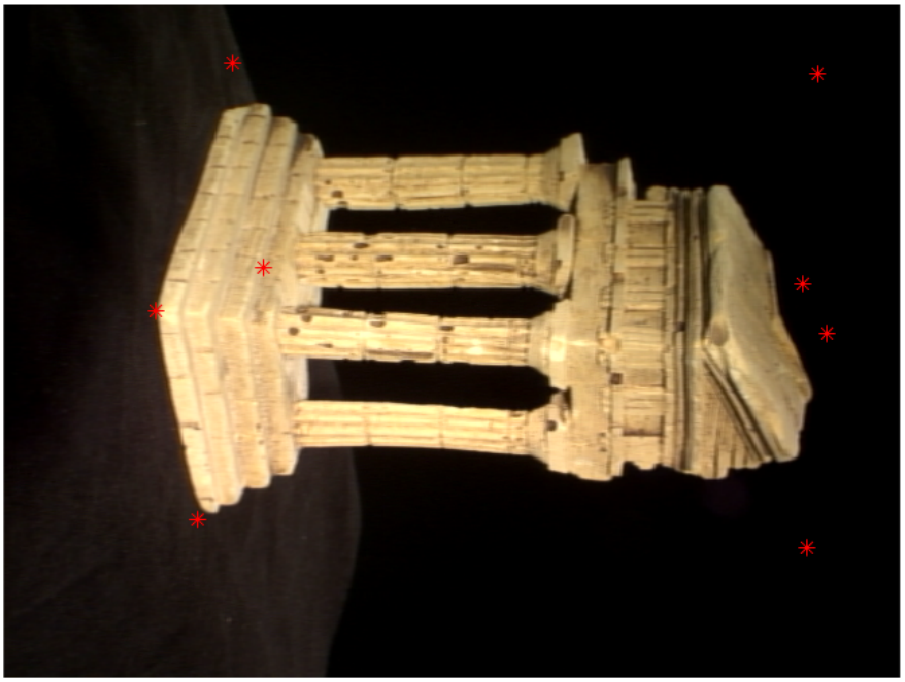
    iv.) Set $R := P\tilde{R}^T P$

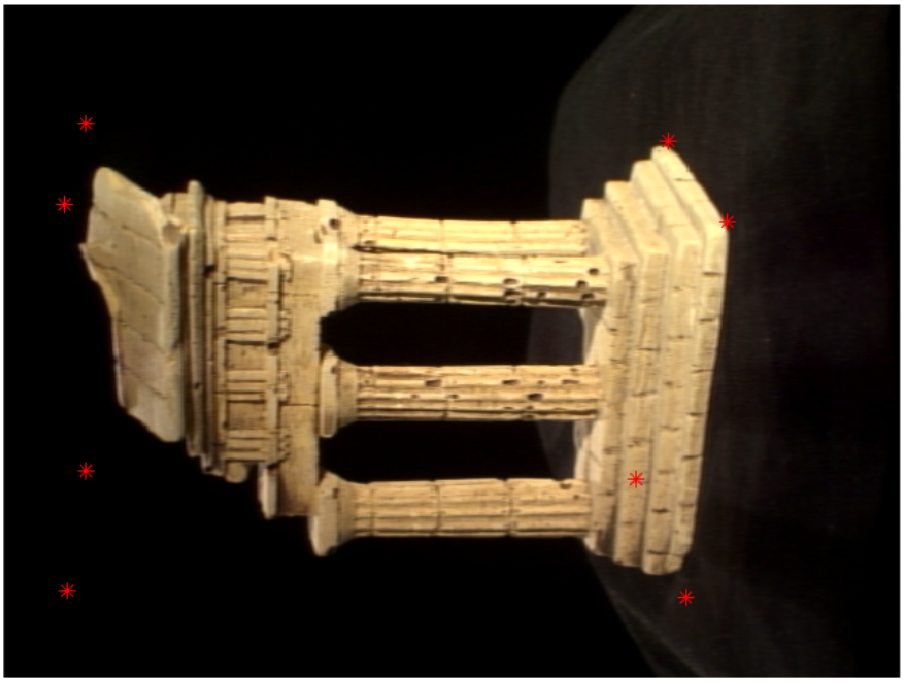
Errors from reprojection of clean and noisy data:

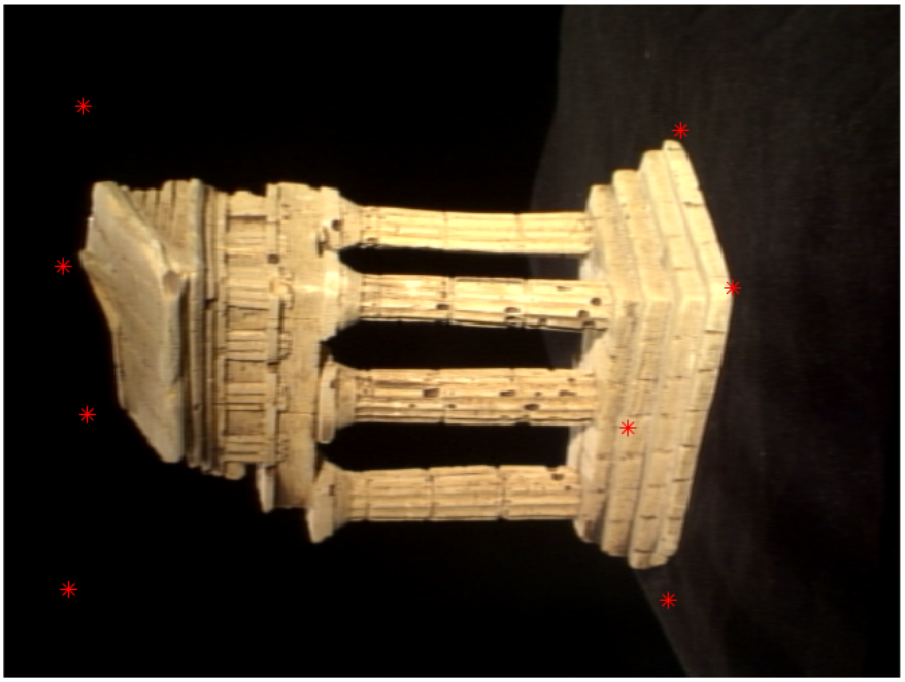
```
>> testKRt
Intrinsic Error with clean 2D points is 0.0000
Rotation Error with clean 2D points is 0.0000
Translation Error with clean 2D points is 0.0000
-----------------------------
Intrinsic Error with clean 2D points is 0.9881
Rotation Error with clean 2D points is 0.0495
Translation Error with clean 2D points is 0.4335
```

- **3.4 Multi-view stereo**
  - 3.4.1 (1 pts)
    I wrote a script named test4_1.m to project the corners to one image and visualize the corners. Change the image to project the corners to different images.

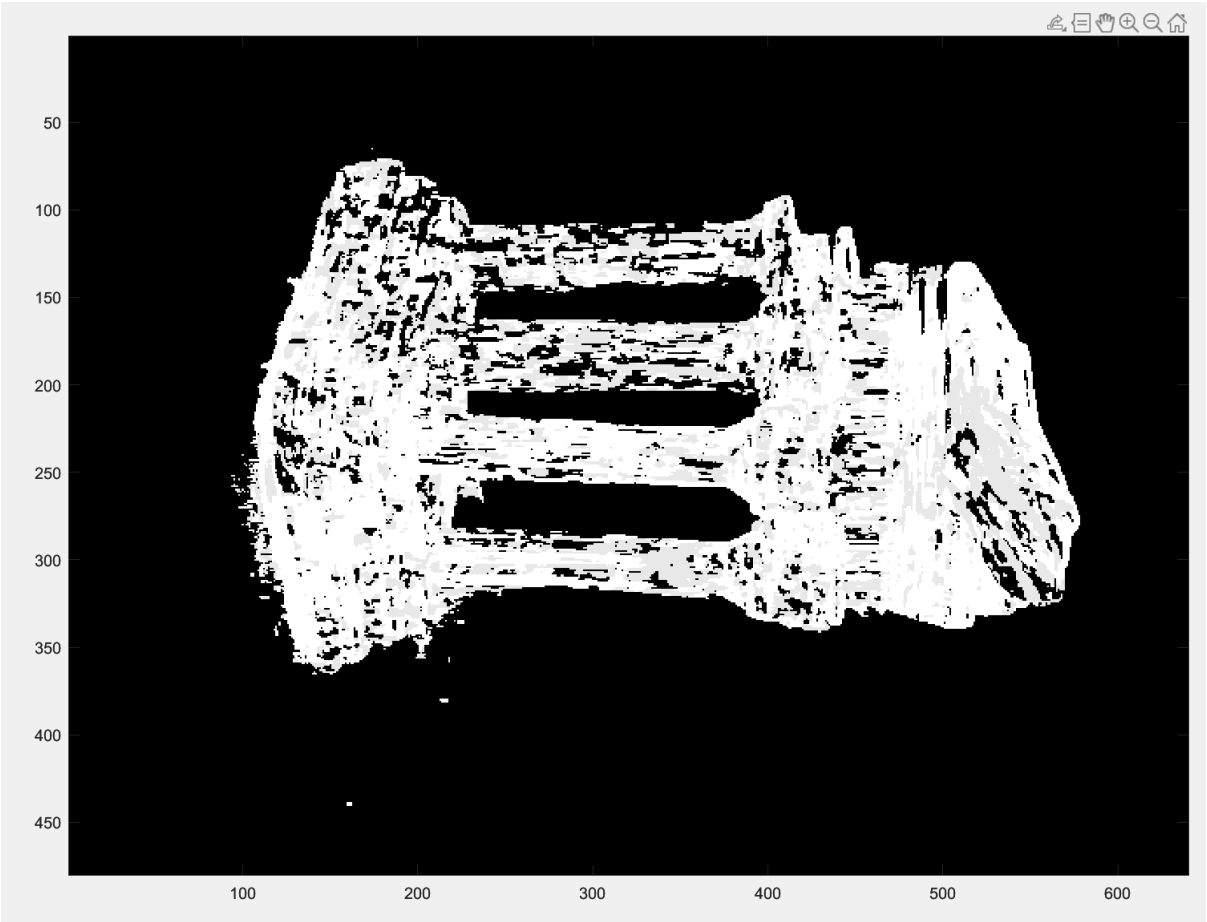
    Results:



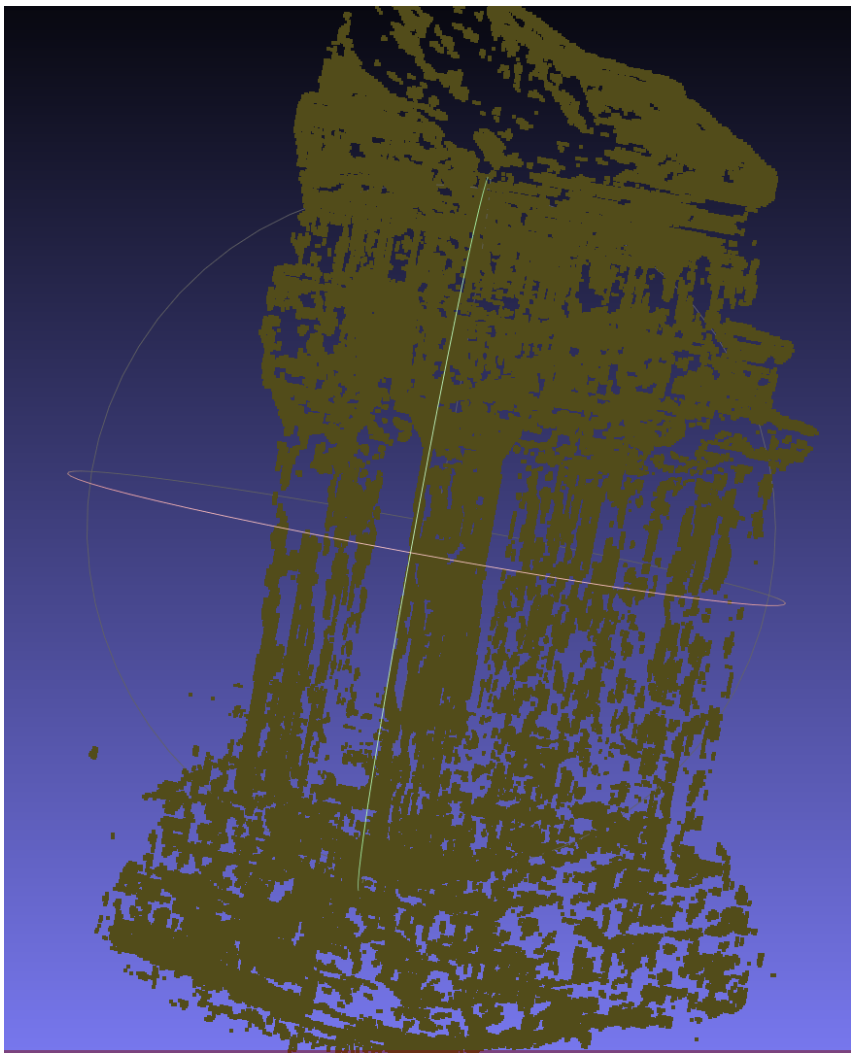


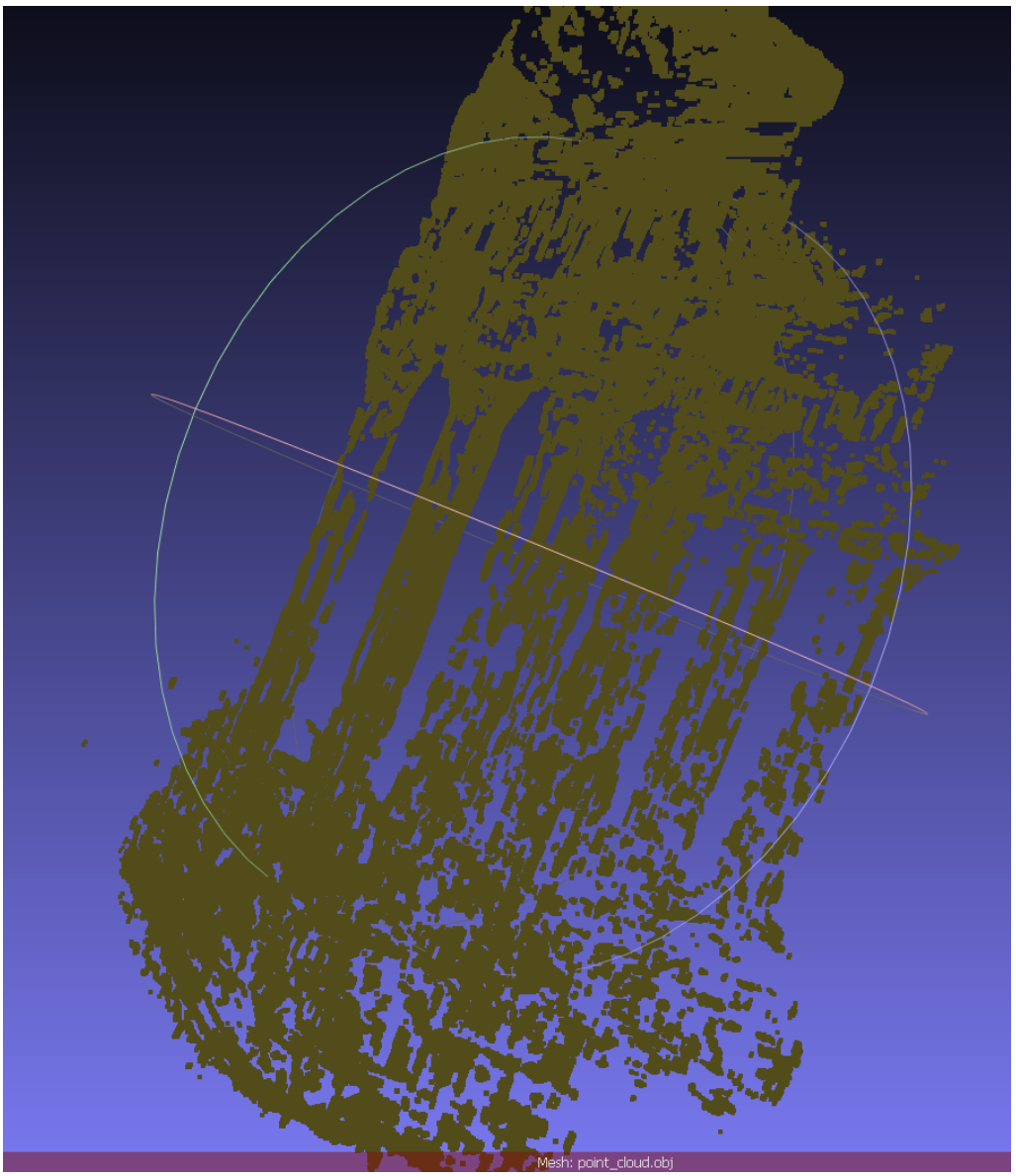


- 3.4.2 (1 pts)
  The script for this part is test4_2.m, run it to show the depth map of Image I0:



- 3.4.3 (1 pts)
  The code for save a depth map as a point cloud in the OBJ format is test4_3.m, and the result from MeshLab is show below:




Mesh: point_cloud.obj