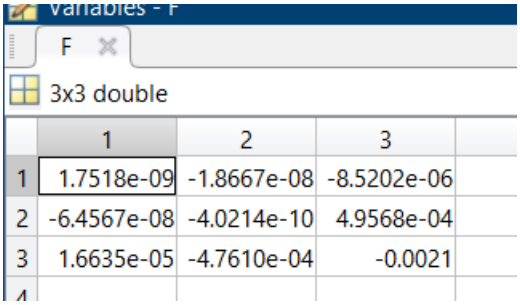
CMPT412 Assignment5

3.1 Spare Reconstruction

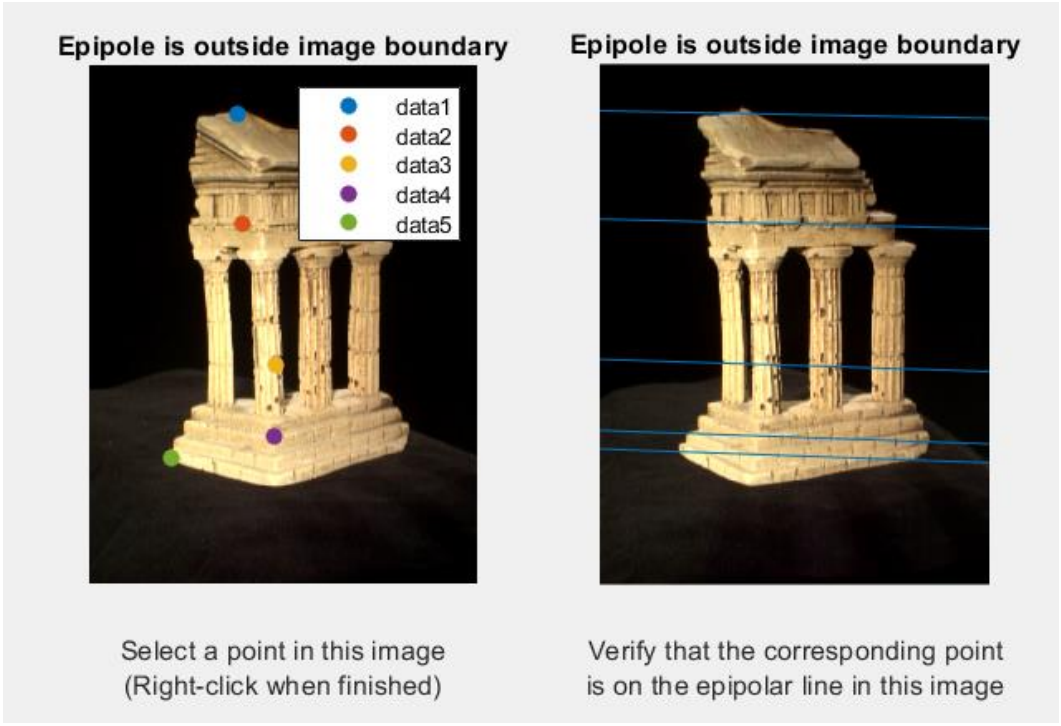
3.1.1.

In your write-up, please include your recovered F and the visualization of some epipolar lines like the figure below.



	1	2	3
1	1.7518e-09	-1.8667e-08	-8.5202e-06
2	-6.4567e-08	-4.0214e-10	4.9568e-04
3	1.6635e-05	-4.7610e-04	-0.0021

F= [0.0000 -0.0000 -0.0000
-0.0000 -0.0000 0.0005
0.0000 -0.0005 -0.0021]

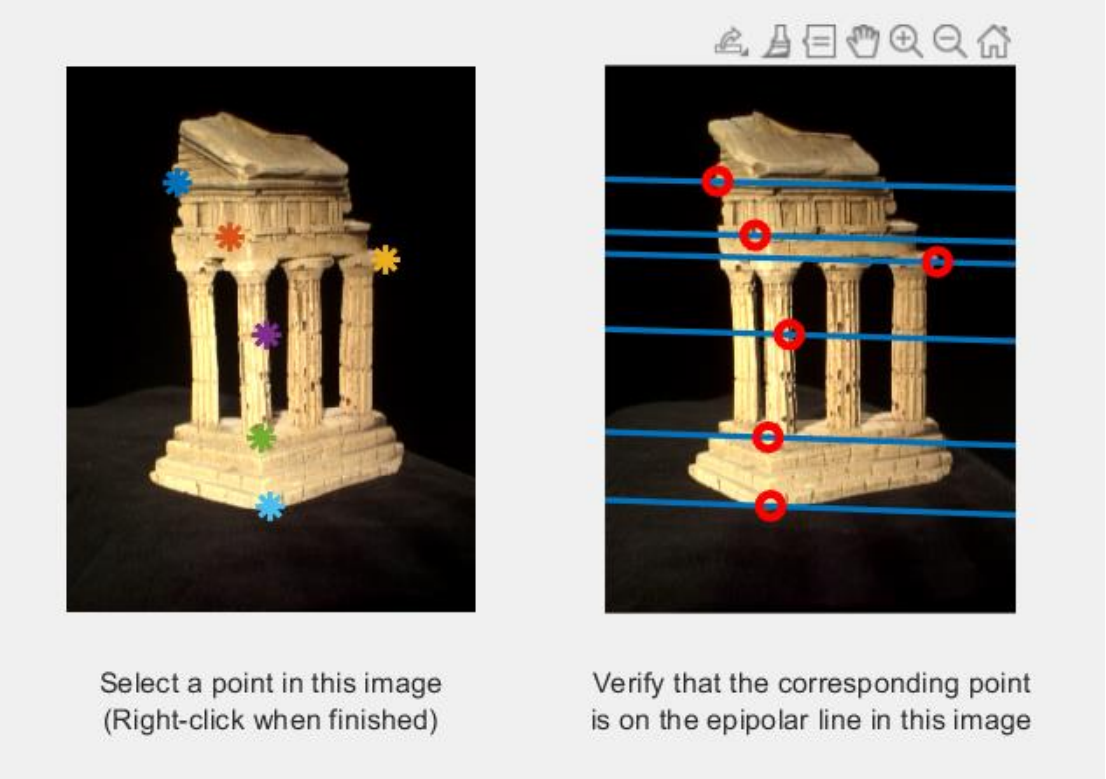


3.1.2

In your write-up, include a screenshot of epipolarMatchGui running with your implementation of epipolarCorrespondence. **Mention the similarity metric you decided to use**. **Also comment on any cases where your matching algorithm consistently fails, and why you might think this is.**

I selected a window of size 21 about point x and compared the targeted window to the window formed from candidate points in the second image. I computed the Euclidean distance for the targeted window and each candidate window. Finally, I would select the candidate point with the smallest Euclidean distance.

However, my metric will fail at images that contain repetitions. I need to compare the pixel similarity between the targeted window and candidate windows and then select the candidate with the highest similarity. If there are repetitions in the image, target windows can match to its repetitions in the second image instead of its real match.



3.1.3

In your write-up, write your estimated E matrix for the temple image pair we provided.

F E

3x3 double

	1	2	3
1	0.0040	-0.0433	-0.0192
2	-0.1498	-9.3633e-04	0.7264
3	0.0019	-0.7352	-8.4658e-04

E= [0.0040 -0.0433 -0.0192
-0.1498 -0.0009 0.7264
0.0019 -0.7352 -0.0008]

3.1.4

In your write-up, **describe how you determined which extrinsic matrices are correct.** Report your re-projection error using pts1, pts2 from someCorresp.mat. If implemented correctly, the re-projection error should be less than 1 pixel.

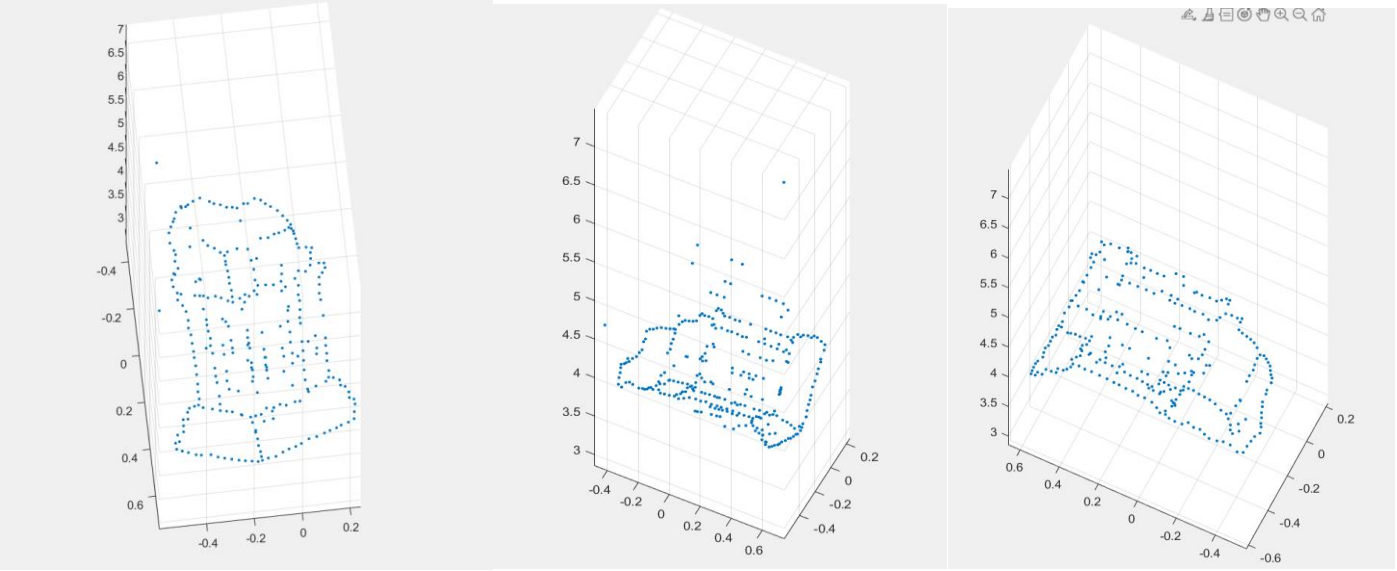
I determined the best extrinsic matrices based on the total number of estimated 3D points that have positive values on the z-axis. I counted the number of estimated 3D points with positive depth computed by the camera matrix made with each candidate extrinsic matrix. The best extrinsic matrix generates the most points with positive depth compared to other candidates.

re-produce error for pt1 from templeCoords.mat and computed pt2:
mean Euclidean error in img1 is: 0.53.
mean Euclidean error in img2 is: 0.53.

The number of positive depth of best candidate is 576.
The best candidate is 2

3.1.5

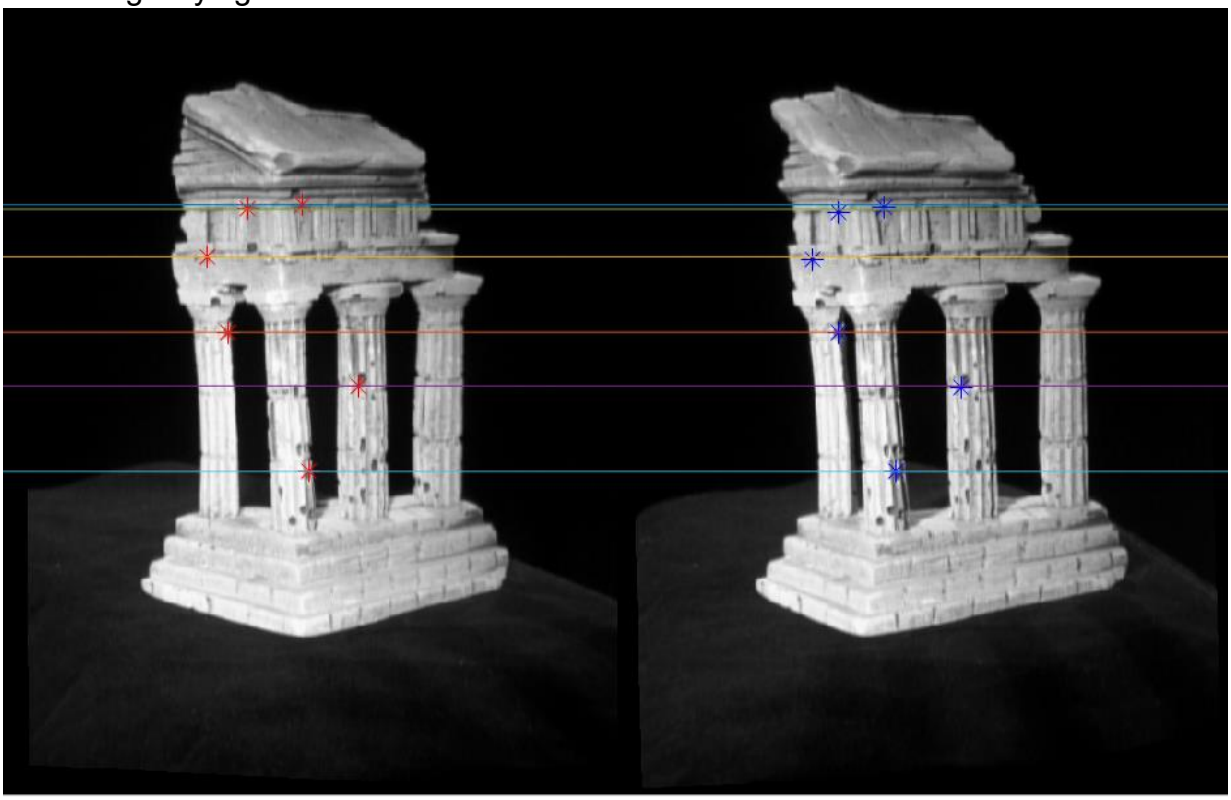
In your write-up, include 3 images of your final of the templeCoords points, from different angles.



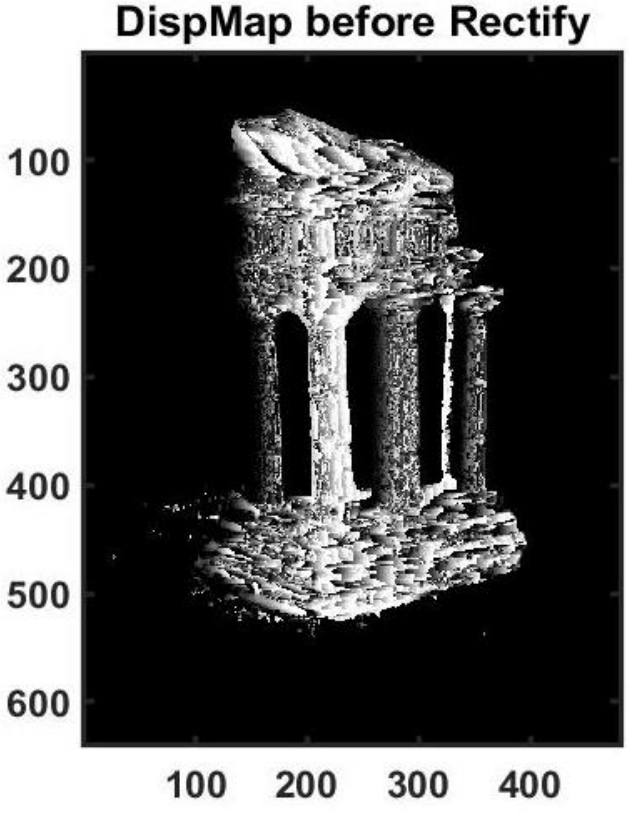
3.2 Dense reconstruction

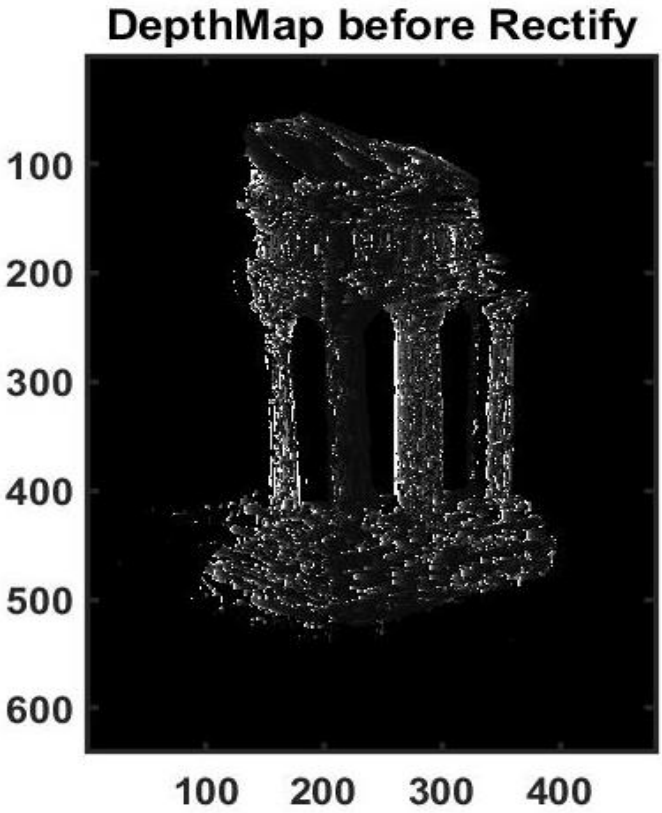
3.2.1

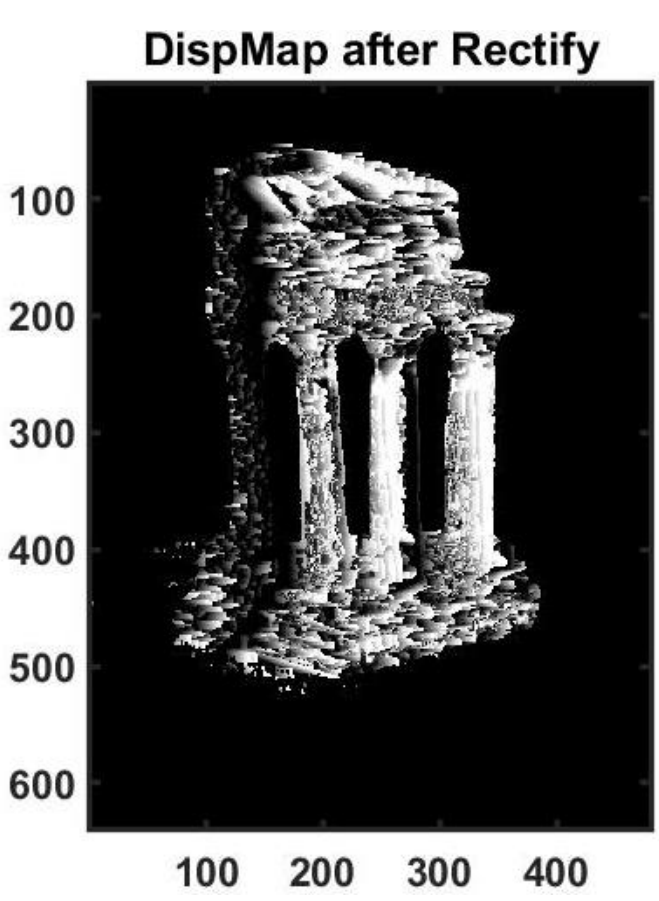
In your write-up, include a screenshot of the result of running testRectify.m on temple images. The results should show some epipolar lines that are perfectly horizontal, with corresponding points in both images lying on the same line.

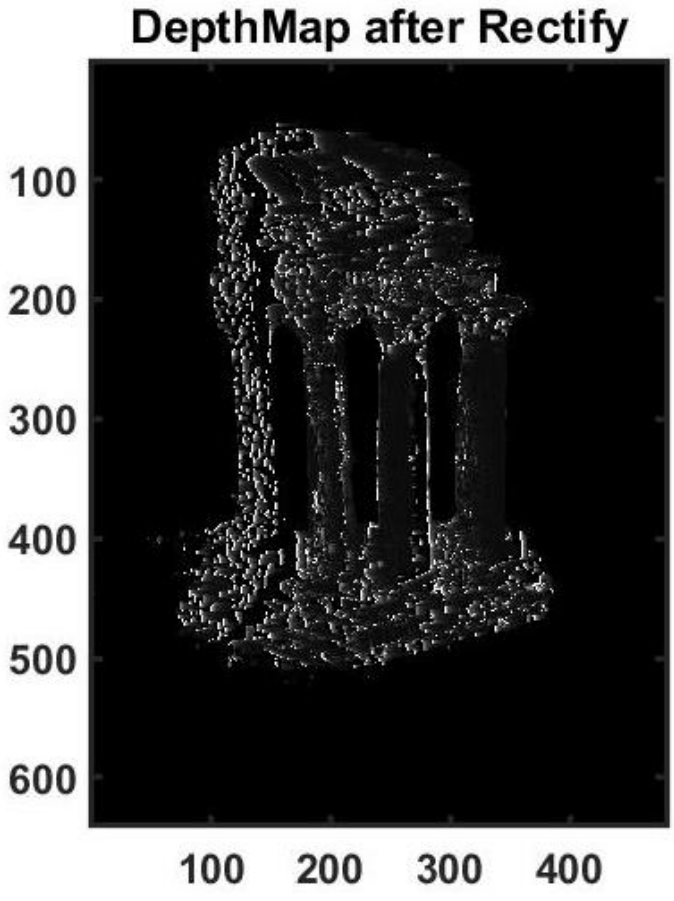


3.2.2
3.2.3
In your write-up, include images of your disparity map and your depth map.









3.3 Pose estimation

3.3.1

In your write-up, include the output of the script 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 2.6236
Pose Error with noisy 2D points is 0.4895
```

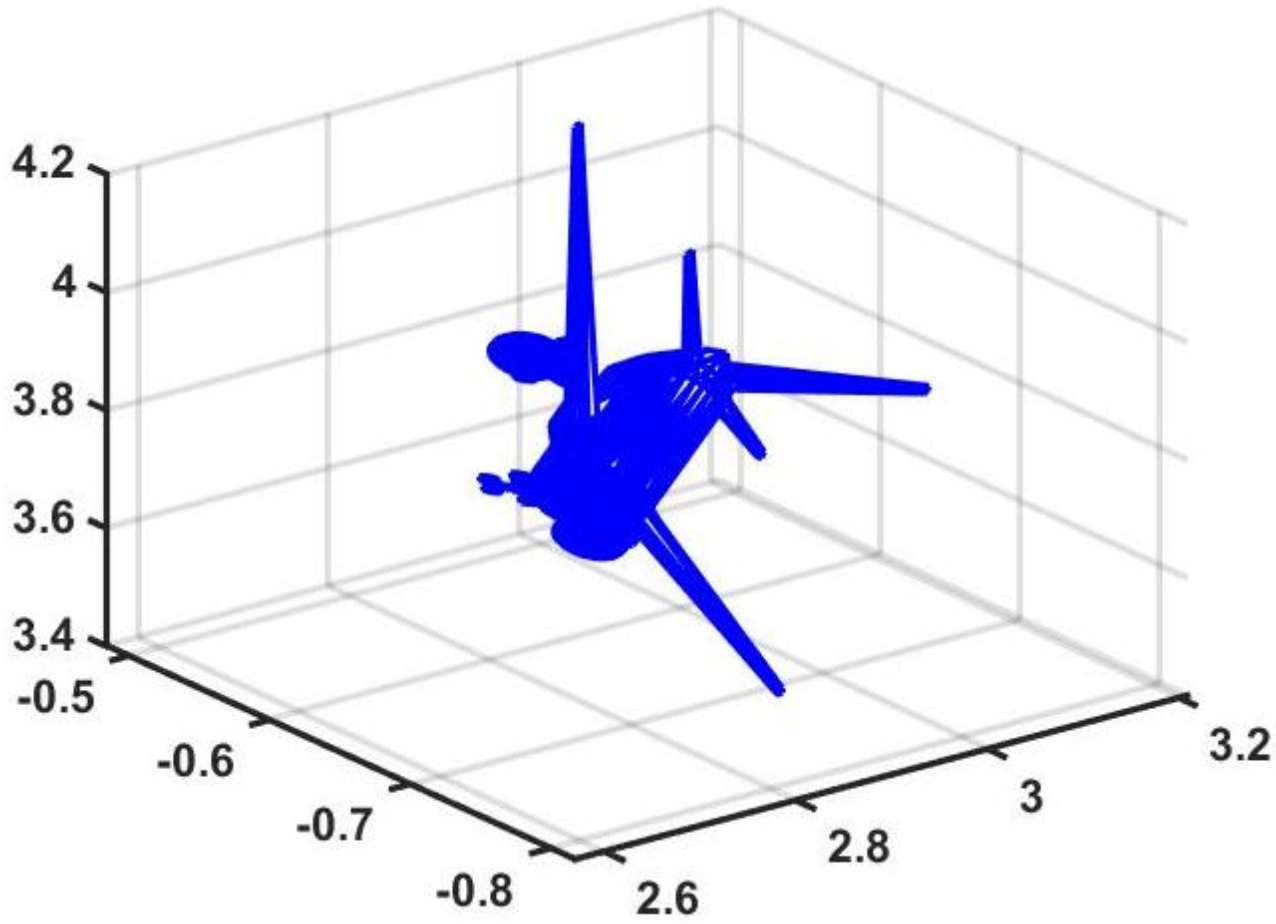
3.3.2

In your write-up, include the output of the script 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 0.6316
Rotation Error with noisy 2D points is 0.2123
Translation Error with noisy 2D points is 0.3203
```

3.3.3

In your write-up, include the three images similar to the above figure. You have to use different colors from the figure. For example, green circle for given 2D points, black points for projected 3D points, blue CAD model, and red projected CAD model overlapping on the image. You will get NO credit if you use the same color.


4.2
4
3.8
3.6
3.4
-0.5
-0.6
-0.7
-0.8
2.6
2.8
3
3.2