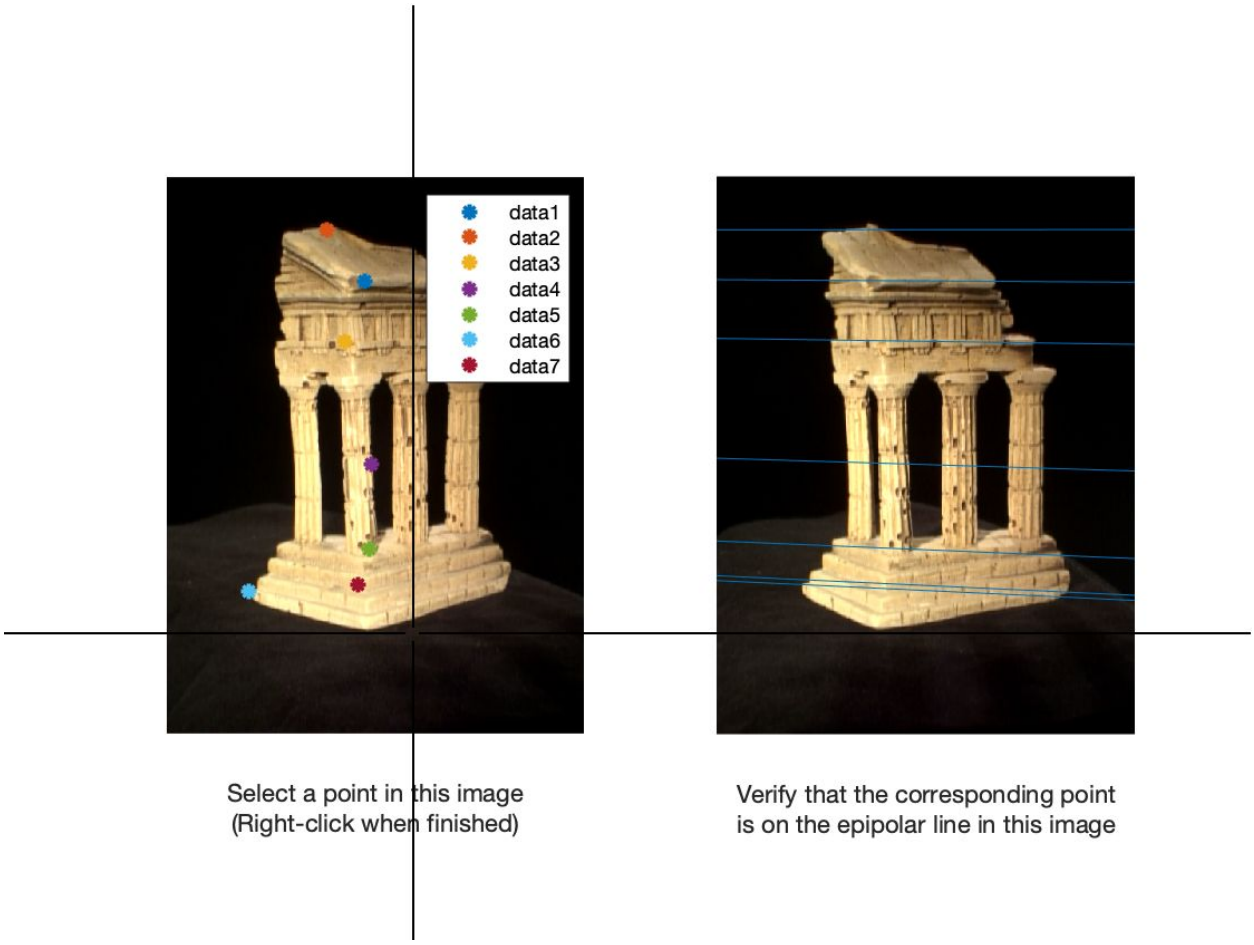
3.1.1 Implement the eight point algorithm

F:

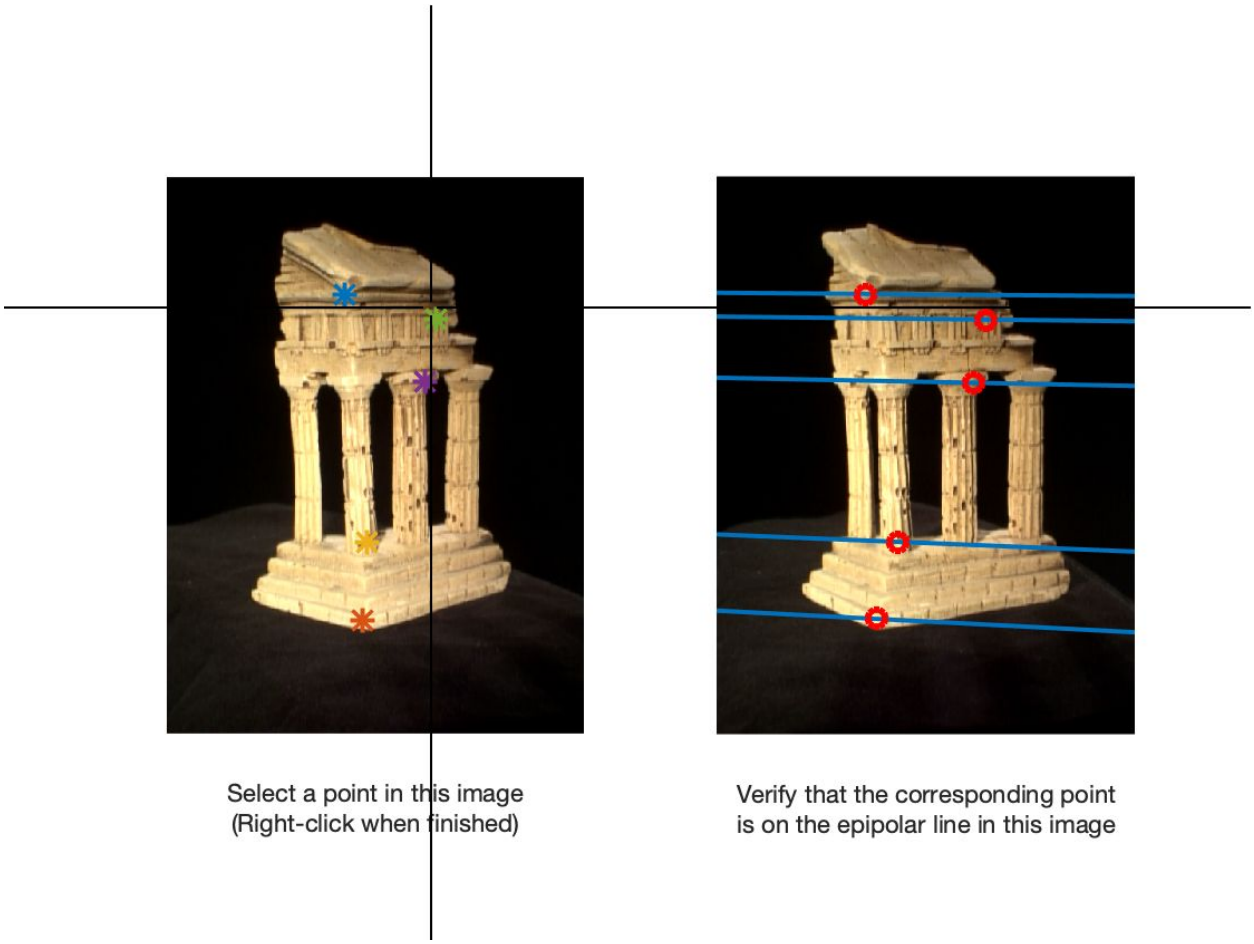
-1.08375815130976e-09	1.75438948702935e-07	-1.17061202991406e-05
8.02952179369438e-08	5.43989454126549e-09	-0.00152201102745318
-1.45035302667240e-05	0.00146039564192221	0.00640087473418170

Epipolar Lines:



3.1.2 Find epipolar correspondences

EpipolarMatchGui:



Similarity metric:

I used Euclidean distance between sub-images.

Failed cases:

1 When repeated patterns exist, it fails. Specially when patterns repeated multiple times along epipolar line. It is reasonable, because repeated pattern always have similar distance score, and it mislead the algorithm.

2 When image have areas with pure color, it fails. Because there is no pattern exist in these color area. If its all back, any point can be similar point.

3.1.3 Write a function to compute the essential matrix

E:

-0.00250523285609937	0.407014564491960	0.0475533245688117
0.186283168025193	0.0126660916593609	-2.28334643026901
0.00758878532338782	2.31139866934321	0.00258506045181384

3.1.4 Implement triangulation

Re-projection Error (test_script_2.m):

```
test_script_2
 0.8503
```

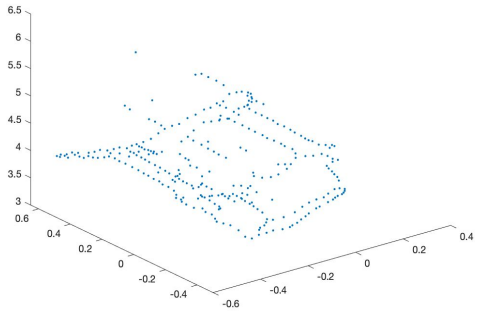
Determined which extrinsic matrices is correct:

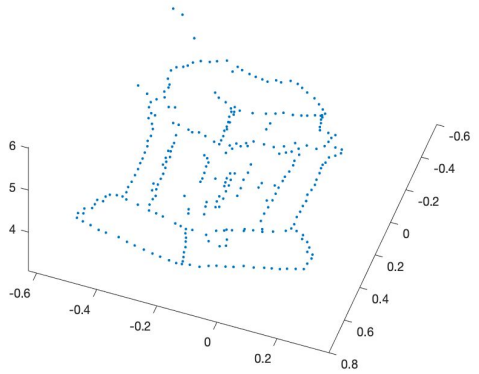
> Briefly speaking: positive depth test.
> In testTempleCoords.m, after I get 4 matrces from camera2.m, I compute Projection matrix (P2) with each of them, and triangulate 3D points base on those P2. Then use 3D points multiple by projection matrix, and see if third value of outcome > 0. If > 0, vote for that matrix.

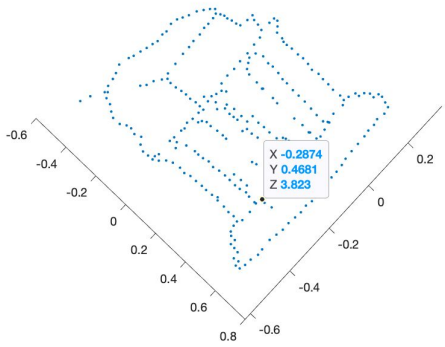
```
>> testTempleCoords
     0   288   274    14
```

> Here is the votes of 4 matrices, so I pick second matrix as my Extrinsics

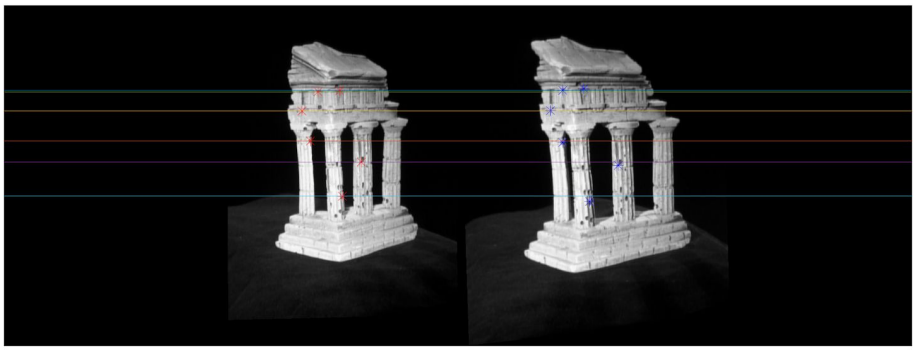
3.1.5 Write a test script that uses templeCoords





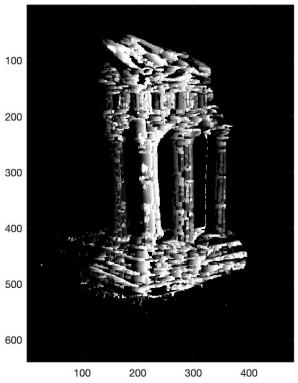


3.2.1 Image rectification



3.2.2 Dense window matching to find per pixel density

Disparity Map:



3.2.3 Depth map

Depth Map: