CMPT 412-COMPUTER VISION

PROJECT-5 REPORT (NAVJOT KAUR: 301404765)

3.1. SPARSE RECONSTRUCTION

3.1.1 Implement the eight-point algorithm

Recovered F after running eightpoint.m:

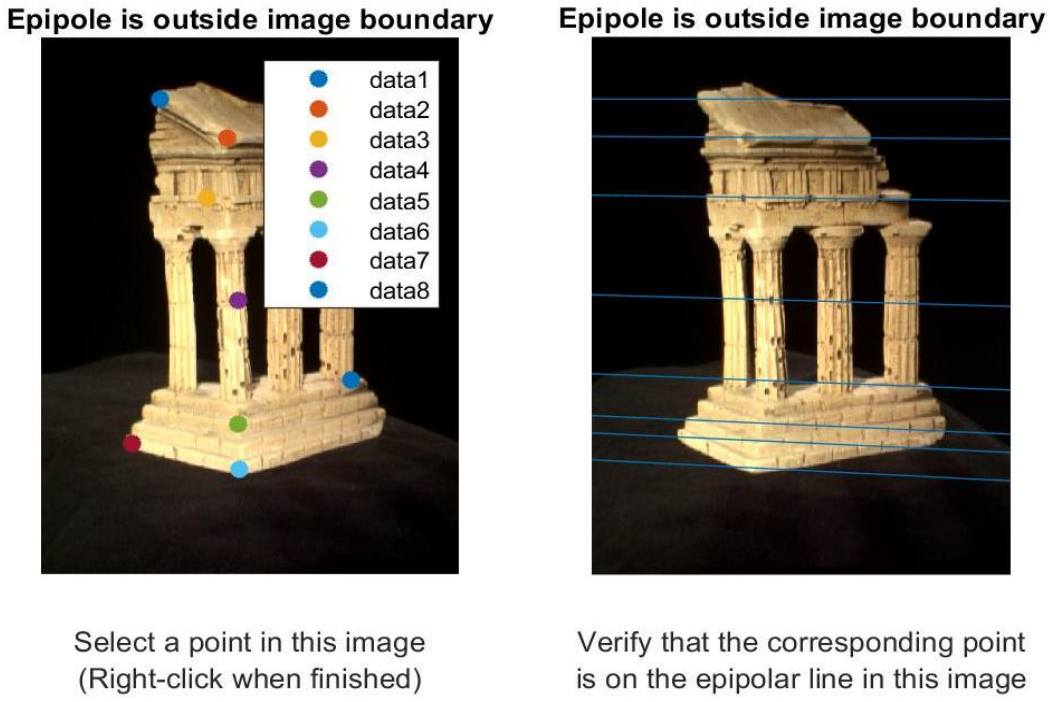
```
Command Window

F =

   -0.0000    0.0000   -0.0000
    0.0000    0.0000   -0.0015
   -0.0000    0.0015    0.0064

fx >>
```

Visualization of some epipolar lines using displayEpipolarF.m:



3.1.2 Find epipolar correspondences

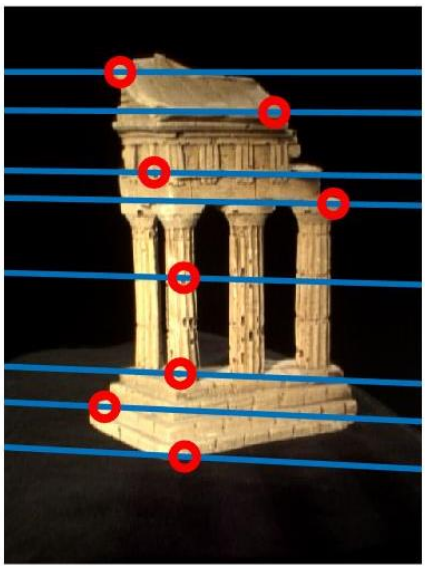
In epipolarcorrespondence.m, the y values of image 2 are calculated with a window size of 20. When the window was too small, I got uneven results so after trial and error window = 20 is chosen.

Euclidean distance of different points of the window is used to calculate similarity metric.

The algorithm succeeded on most of the points such as corners, dots etc, the only cases when the algorithm fails is when the window being observed have too many similar points but that is an unmatched window on the line. This might be because of the similarity of the points inside the observed window and the actual points. For example, in this image the top part has similar neighboring points and the algorithm might fail if on the same epipolar line there are two or more similar points.

Resulting image using window size 20:

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

Essential matrix:

Command Window

```
E =

   -0.0025    0.4070    0.0476
    0.1863    0.0127   -2.2833
    0.0076    2.3114    0.0026

fx >>
```

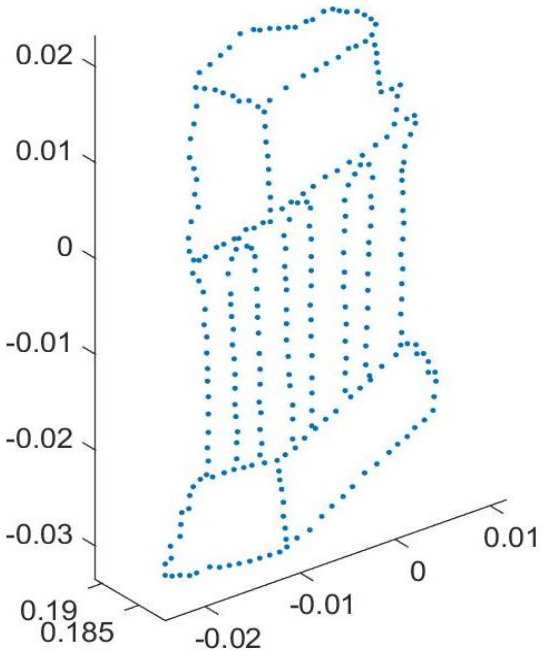
3.1.4 Implement triangulation

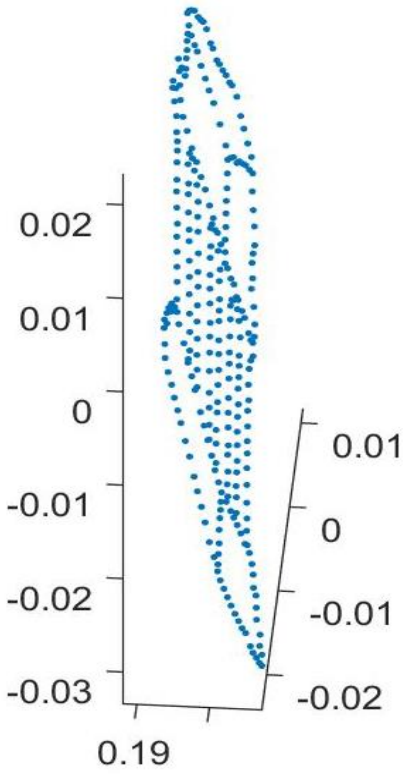
In order to determine the correctness of extrinsic matrices, firstly the camera projection matrices are obtained by multiplying intrinsic matrices. We are given 4 extrinsic matrices and I calculated the 3D points such that the points which are in front of both the cameras are considered. By this I found out that 4th extrinsic matrix was correct because it has the highest number of correspondence points infront of the cameras and this matrix is chosen for calculations.

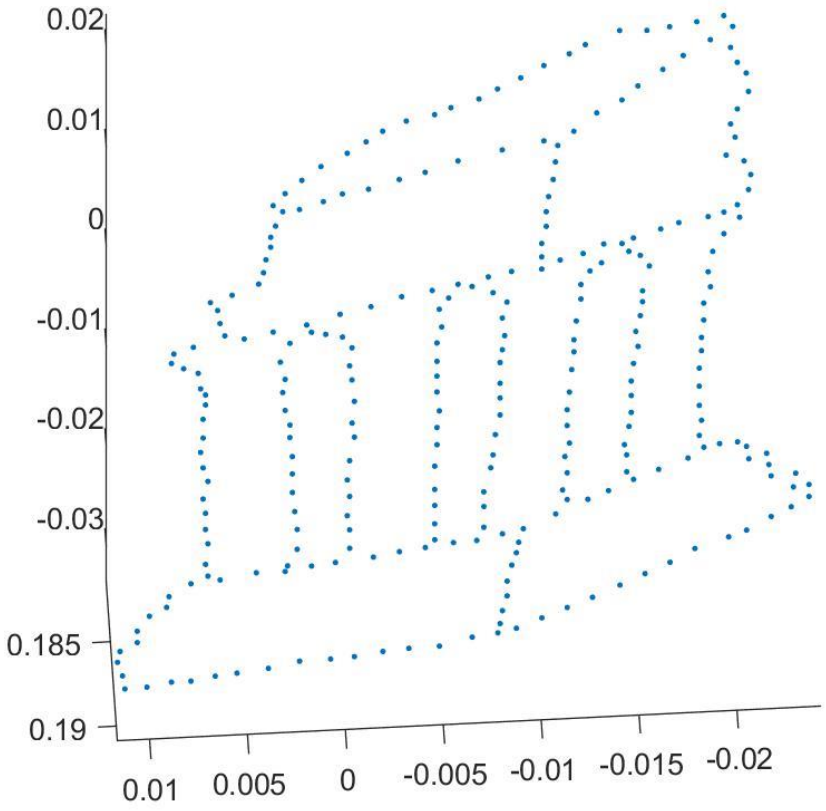
The re-projection error using pts1 and pts2 is calculated via Euclidean where I got 0.6028 for pts1 and 0.6058 for pts2. The re-projection error is correct because it is less than 1 pixel.

3.1.5 TempleCoords

Below are the three images of final reconstruction of temple points from different angles:



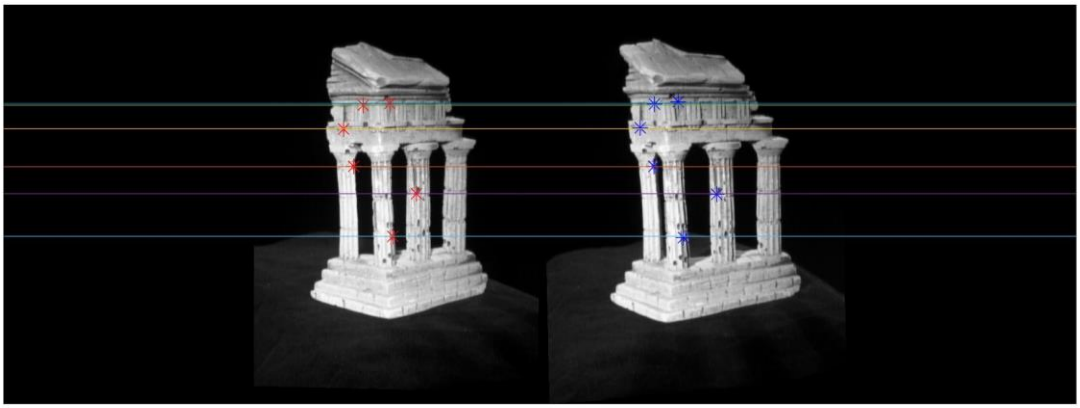




3.2 DENSE RECONSTRUCTION

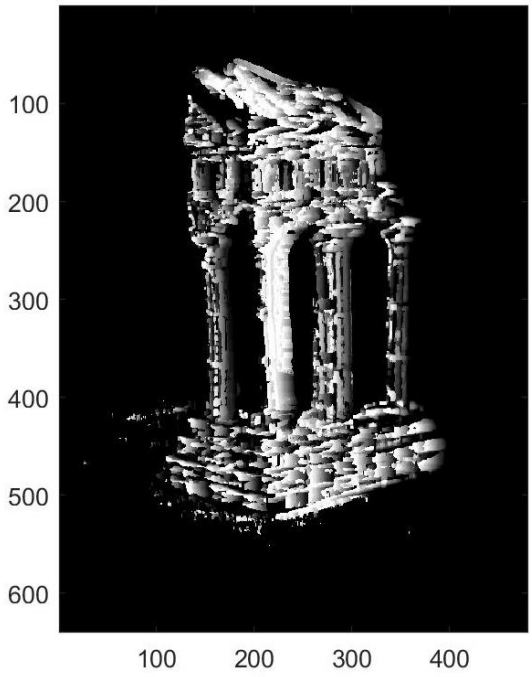
3.2.1 Image Reactification

Results:



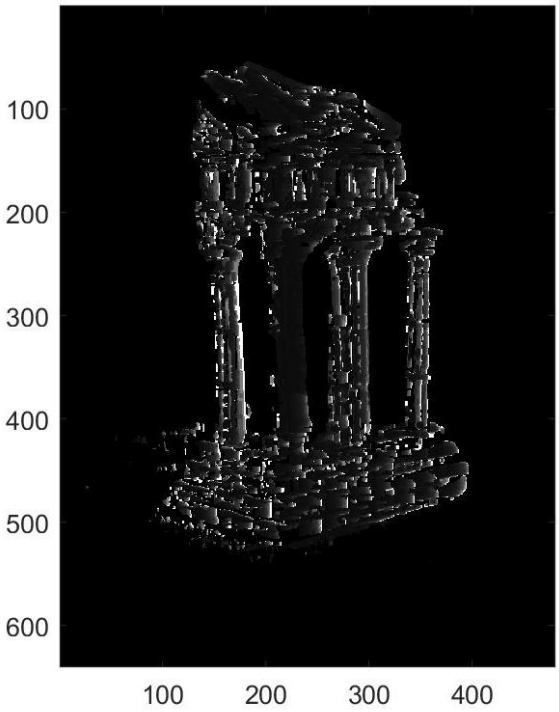
3.2.2 Dense window matching to find per pixel density

Resulting image is of Disparity Map after running testDepth.m:



3.2.3 Depth Map

Resulting image is of Depth Map after running testDepth.m:



3.3 POSE ESTIMATION

3.3.1 Estimate camera matrix P

Output:

```
Command Window
>> 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 1.9665
Pose Error with noisy 2D points is 0.0720
fx >>
```

3.3.2 Estimate intrinsic/extrinsic parameters

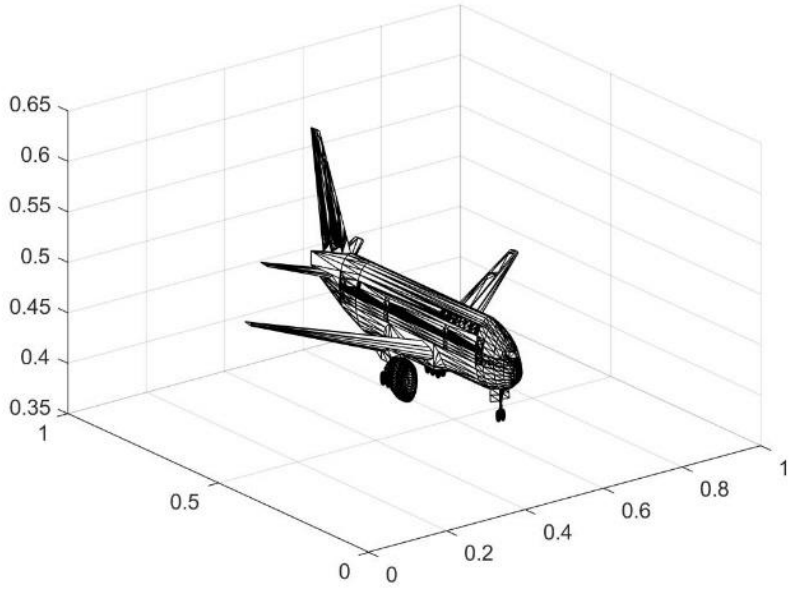
Output:

```
Command Window
>> 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 0.6566
Rotation Error with clean 2D points is 0.1098
Translation Error with clean 2D points is 0.2478
fx >>
```

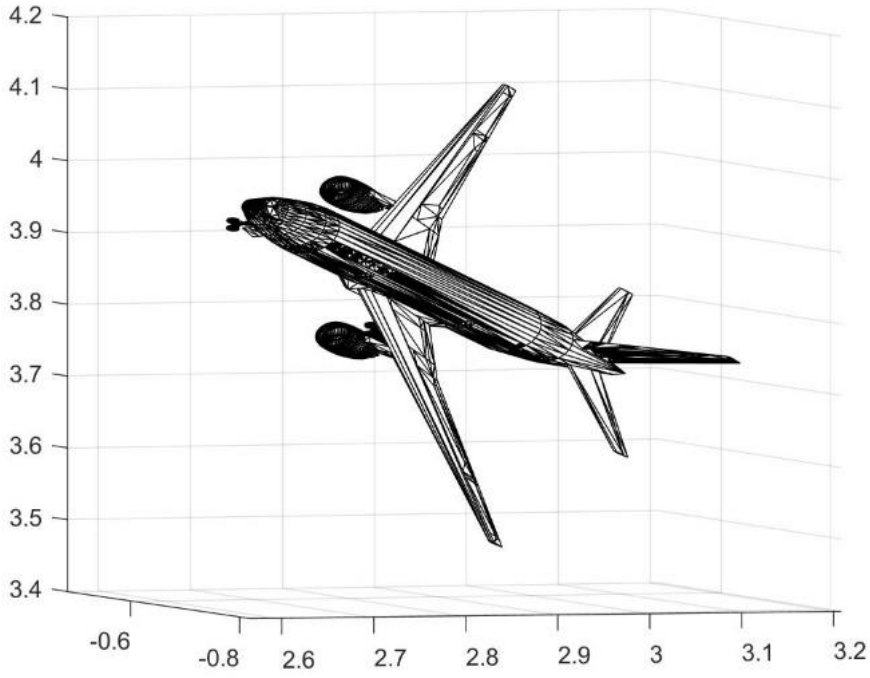
3.3.3 Project a CAD model to the image

1. 2d points are indicated by red dots and 3d points are indicated by blue circles.

2. 3D image with vertices



3. 3D image with rotation:



4. CAD model overlapping with 2D image: