

# Project 5

## 3D Reconstruction

Name: Muhammad Shahzaib Waseem

### Sparse Reconstruction

#### Eight Point Algorithm

The original F matrix looks like this:

```
>> runner  
  
F =  
  
-0.0000    0.0000   -0.0000  
 0.0000    0.0000   -0.0015  
-0.0000    0.0015    0.0064
```

Figure 1: Recovered Fundamental Matrix  $F$

The output of `eightpoint.m` looks like this:

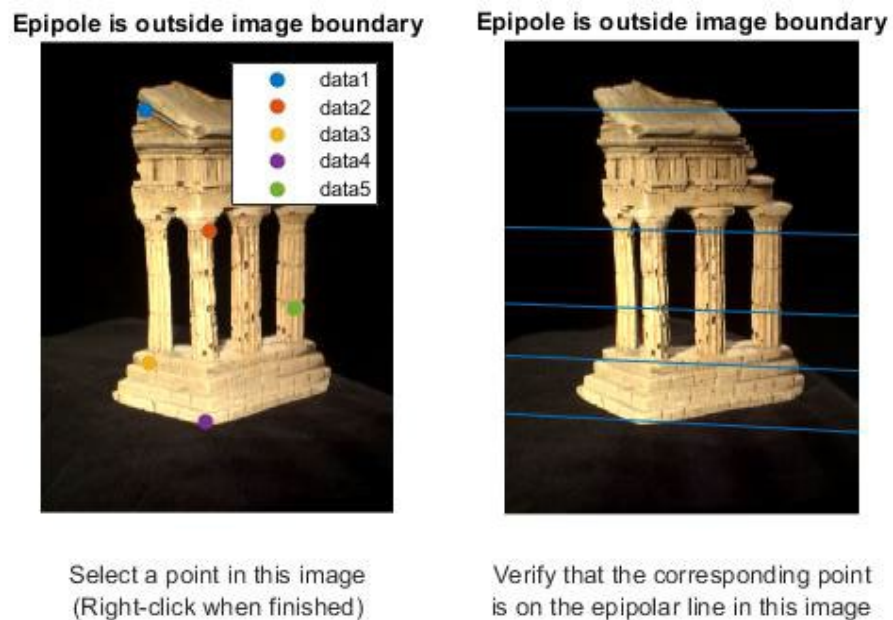


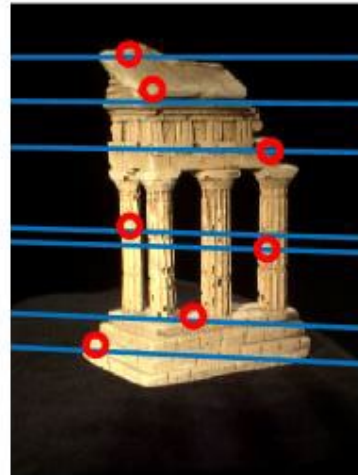
Figure 2: Epipolar Lines

## Epipolar correspondences

The epipolar correspondences from the temple image 1 to temple image 2 look like the following:



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



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

Figure 3: Epipolar Correspondence between the two temple images

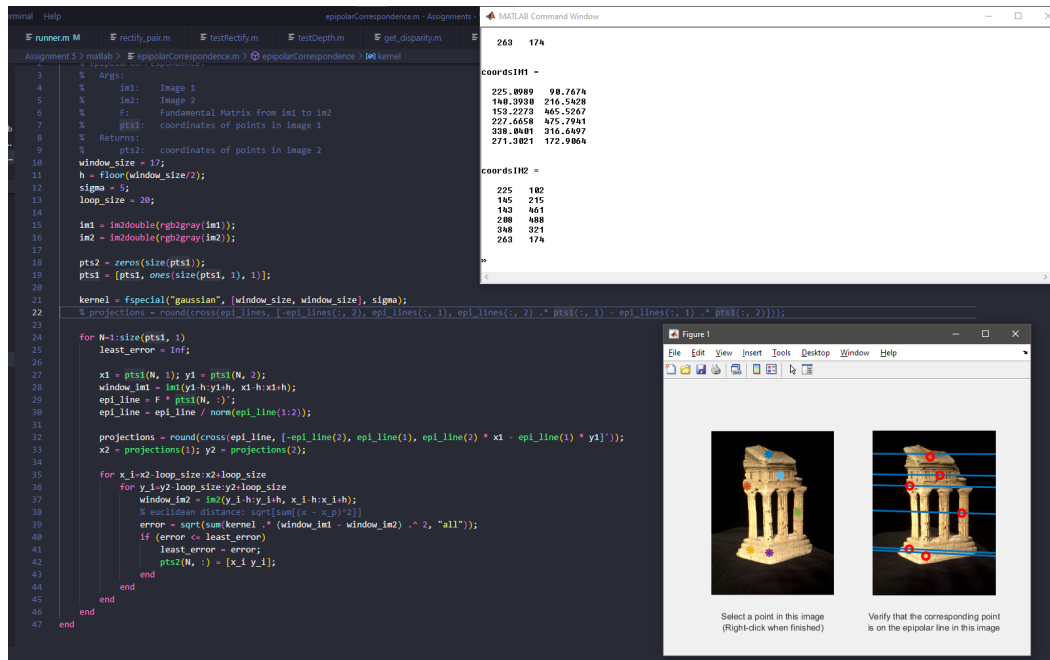


Figure 4: Screenshot with the implementation

## Similarity Metric

The similarity metric I employed for this task was Euclidean distance, which is given by:

$$error(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

or

```
error = sqrt(sum(kernel .* (window_im1 - window_im2).^2, "all"))
```

This will take a subset from both of the images and subtract them and square the residual and sum all of the values in the kernel and take the root of the resulting value. The window size for this is set to be 17 pixels.

## Explanation

I think the algorithm failed when there are a lot of points (or windows) which are very similar along the epipolar line, as it is visible from the [Image 3](#) that the purple point lies on the stairs which can be repeated multiple times throughout the image, so this makes the algorithm fail for these certain cases.

## Compute the Essential Matrix (E)

The essential matrix, E, looks like this:

```
>> runner

E =

   -0.0025    0.4070    0.0476
    0.1863    0.0127   -2.2833
    0.0076    2.3114    0.0026
```

Figure 5: Essential Matrix E

## Triangulation

### Explanation

I ran a loop for all of the four candidate extrinsic matrices (P2) and ran the whole algorithm and measured the positive depth pixels (pixels which are in front of the camera and not behind). The best performing matrix was saved and used.

### Errors

The two reprojection errors are: reproj\_1 = **0.2783** and reproj\_2 = **0.2711** using the points provided in the `templeCoords.mat` file.

If I calculate the reprojection error using the points provided by `someCorresp.mat` I get errors equal to: reproj\_1 = **0.2744** and reproj\_2 = **0.2670**.

All together

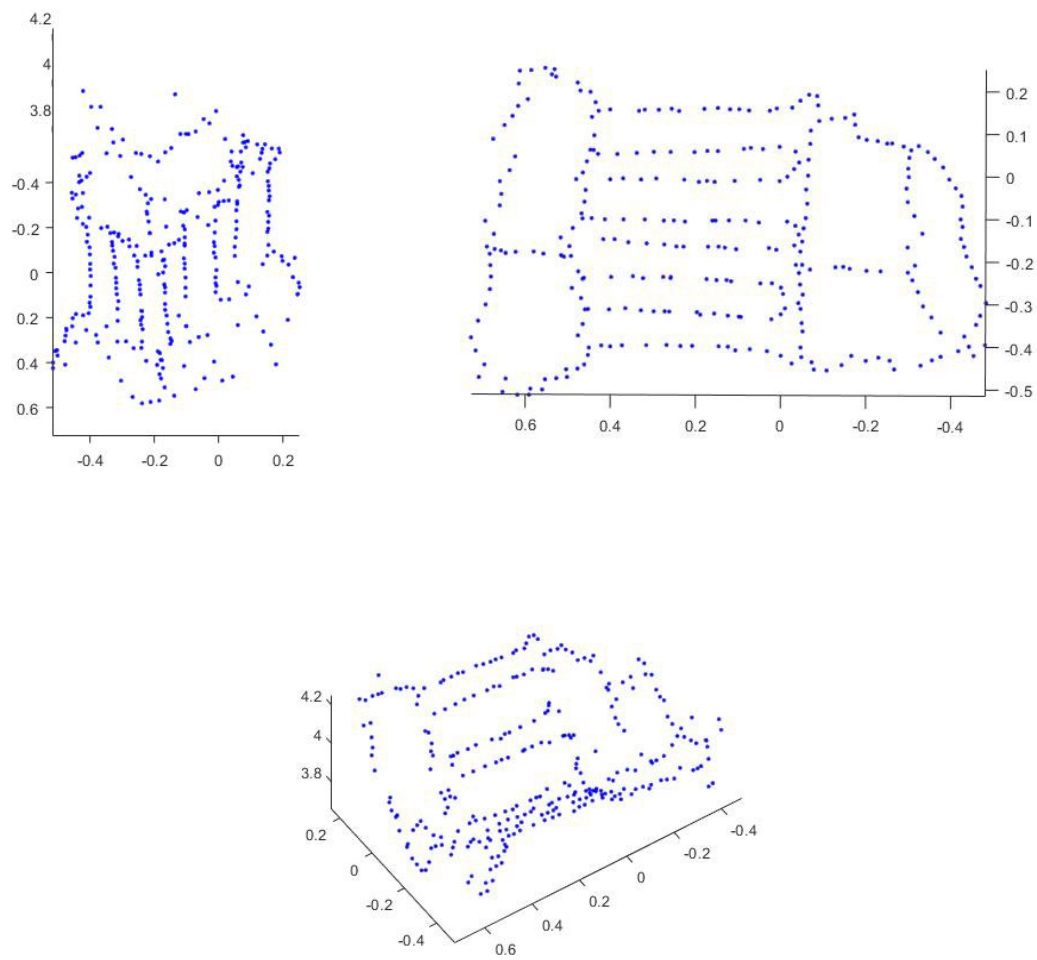


Figure 6: Sparse reconstruction of templeCoords

# Dense Reconstruction

## Image rectification

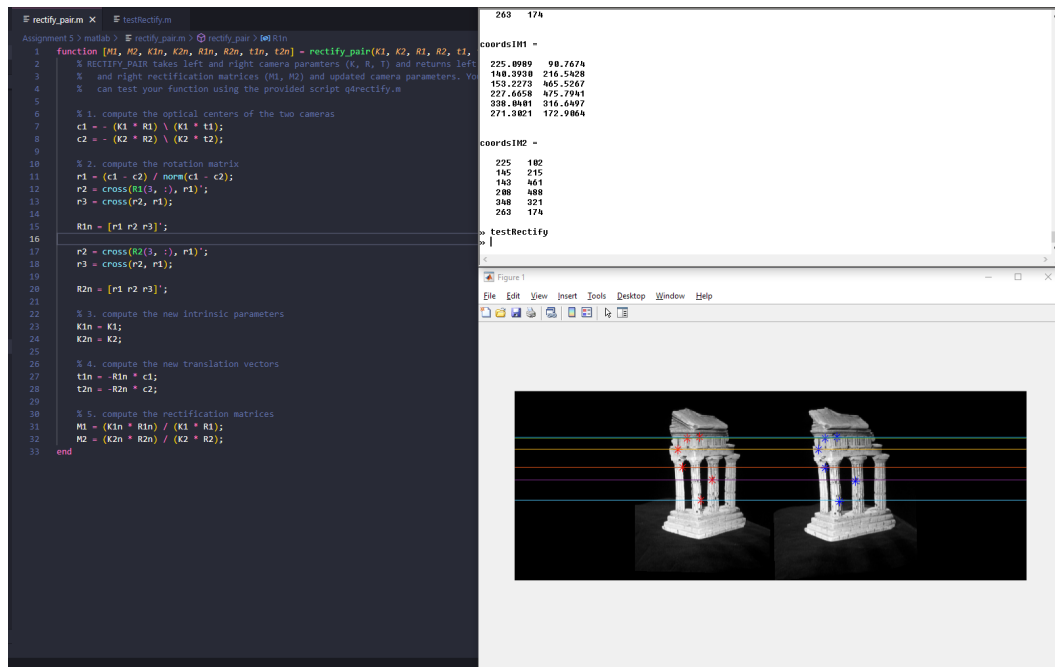


Figure 7: Screenshot of the whole screen while running testRectify.m

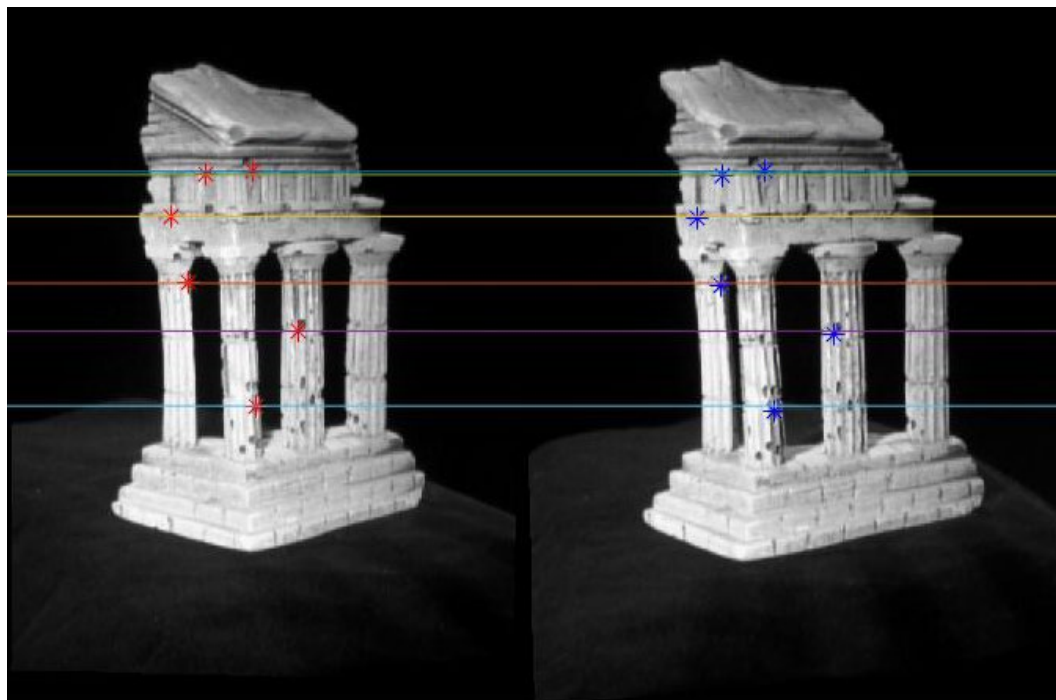


Figure 8: Rectify Image (Enlarged Image)

## Dense window matching



Figure 9: Disparity Map

## Depth map

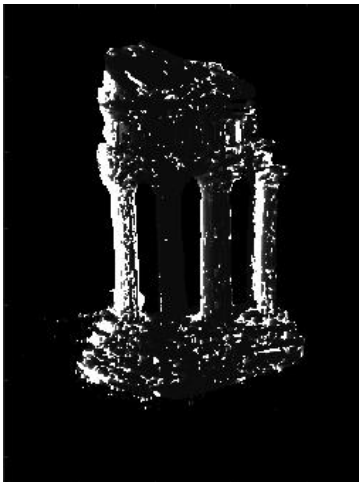


Figure 11: Depth Map

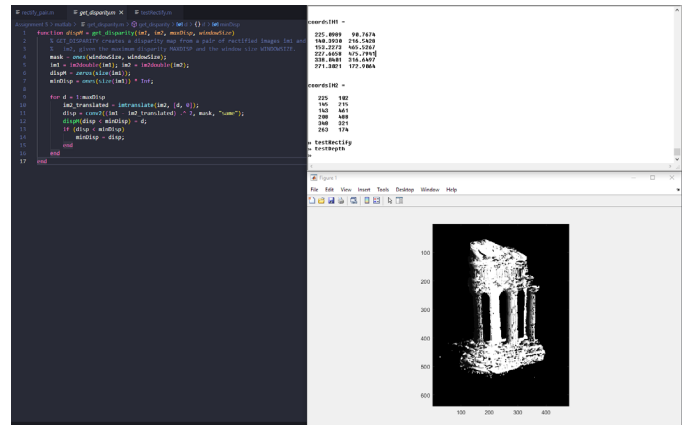


Figure 10: Screenshot of the whole screen

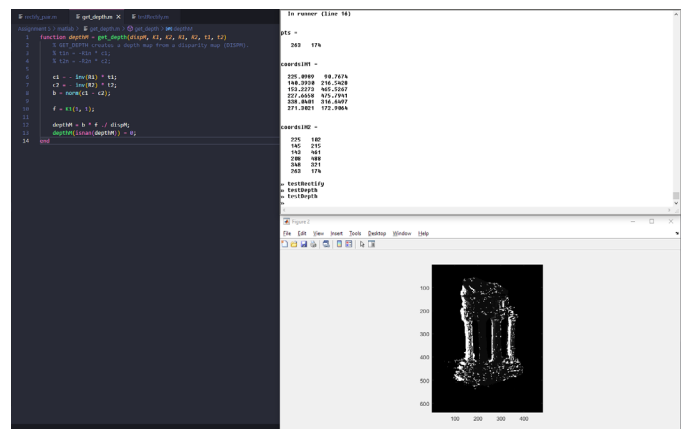


Figure 12: Screenshot of the whole screen

## Pose Estimation

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 2.3516
Pose Error with noisy 2D points is 0.1831
```

Figure 13: Estimate Camera Matrix  $P$  (testPose.m script)

Estimate intrinsic/extrinsic parameters

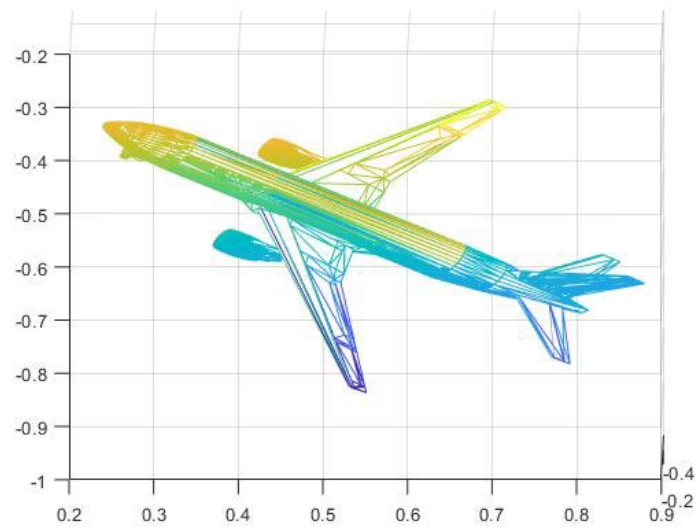
```
>> testKRt
Intrinsic Error with clean 2D points is 0.0000
Rotation Error with clean 2D points is 0.0000
Translation Error with clean 2D points is 0.0000
-----
Intrinsic Error with clean 2D points is 0.5802
Rotation Error with clean 2D points is 0.1829
Translation Error with clean 2D points is 0.2097
```

Figure 14: Estimate intrinsic/extrinsic parameters (testKRt.m script)

Project a CAD model to the image



Figure 15: 3D and 2D projections on the airplane



*Figure 16: trimesh graph of the CAD model*



*Figure 17: CAD model overlayed on the airplane image*