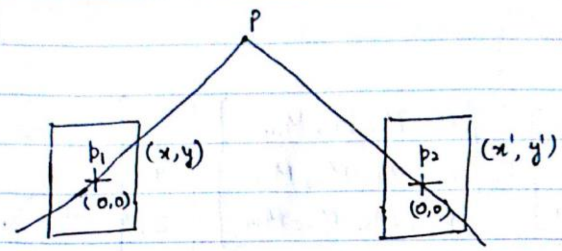
16-720 Computer Vision: Homework 4
3D Reconstruction

Abhishek Bhatia (abhatia1@andrew.cmu.edu)

3/24/2016

1) Theory

1.1)



We know that,

$$p_1\text{-homogenous} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad p_2\text{-homogenous} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x & y & 1 \end{bmatrix} F \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = 0$$

$$\Rightarrow \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} — F_1 — \\ — F_2 — \\ — F_3 — \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

$$\Rightarrow \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} F_{13} \\ F_{23} \\ F_{33} \end{bmatrix} = 0$$

$$\Rightarrow \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{13} \\ F_{23} \\ F_{33} \end{bmatrix} = 0$$

Hence, if image coordinates are normalized so that the coordinate Origin (0,0) coincides with principal point,

$$\Rightarrow \boxed{F_{33} = 0}$$

the F_{33} element of fundamental matrix is zero.

1.2) We know that,

The epipolar line is computed as:

$$l_2 = E x_1 \quad \left\{ \text{where } l_2 = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \text{ such that } ax+by+c=0 \right\}$$

Now,

$$E = \hat{T}R = T\times R = \underbrace{\begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}}_{[T]_\times} \underbrace{\begin{bmatrix} & & \\ & R & \\ & & \end{bmatrix}}_{R}$$

where, T and R is translation and Rotation of one camera with respect to the other.

Now, since there is pure translation parallel to x-axis

$$\Rightarrow t_2 = t_3 = 0, \quad t_1 = t_x \ \& \ R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow E = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_x \\ 0 & t_x & 0 \end{bmatrix} [I]$$

Now,

$$l_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_x \\ 0 & t_x & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -t_x \\ y_1 t_x \end{bmatrix}$$

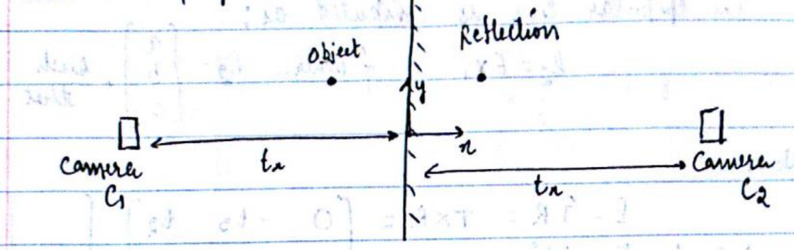
Hence, Epipolar line 'l_2' is : $-\cancel{t_x} y + c\,\cancel{t_x} y_1 = 0$

$$\Rightarrow \boxed{y = c\,y_1} \Rightarrow y \text{ is a constant}$$

$\Rightarrow$ Epipolar lines in the two cameras are also parallel to x-axis.

The situation such that a camera views object and its reflection in plane mirror, is similar to, having 2 camera's looking

1.3) at the same scene, as shown below, the 2 camera's obviously have same properties.



From the figure, we can see that Translation & Rotation of Camera 2 with respect to Camera 1 is:

$$R = \begin{bmatrix} +1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} 2t_x \\ 0 \\ 0 \end{bmatrix}$$

Now, Essential Matrix $(E) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_x \\ 0 & t_x & 0 \end{bmatrix} \begin{bmatrix} +1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$\Rightarrow E = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_x \\ 0 & t_x & 0 \end{bmatrix} \quad \& \quad E^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & t_x \\ 0 & -t_x & 0 \end{bmatrix}$$

Since, $E^T = -E$, E is a skew symmetric matrix

Now, $F = K_2^T E K_1$ {Fundamental Matrix}

$F^T = (K_2^T E K_1)^T$

$= (E K_1)^T K_2$

$= K_1^T E^T K_2$

& $K_1 = K_2 = K$ {∵ same camera}

$\Rightarrow F^T = K^T E^T K$

$= -(K^T E K)$

$= -F$

Since, $\boxed{F^T = -F}$

$\Rightarrow$ Fundamental matrix is skew-symmetric.

2) Practice

2.1) Fundamental Matrix Estimation: Eight Point Algorithm

F =

0.0000 -0.0000 0.0165

-0.0000   0.0000   -0.0003

-0.0158   0.0006   -0.0659

M =

0.0031   0   -1.0000

0   0.0042   -1.0000

0   0   1.0000

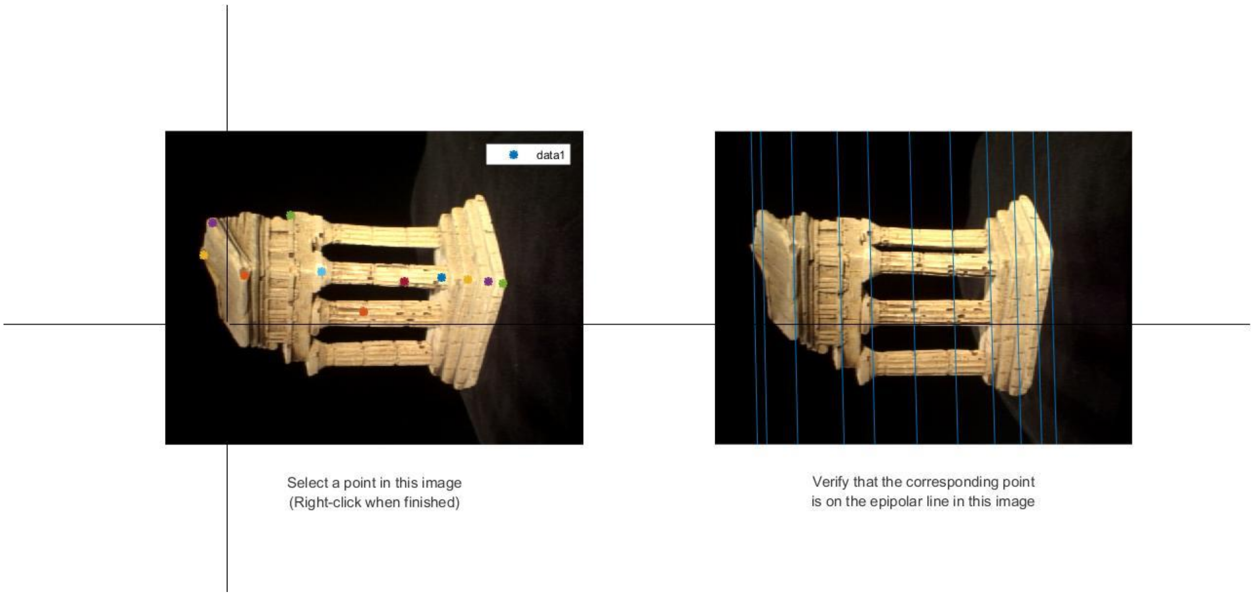


Figure 1: Output of displayEpipolar.m for 8-point algorithm

2.2) Fundamental Matrix Estimation: Seven Point Algorithm

The seven-point algorithm gives average results corresponding to 7 randomly selected points from the 'some_corresp.mat' provided. There can be 2 reasons behind results not being as accurate as above (8-point, over-constrained system):

(i) The seven-point algorithm is very sensitive to outliers. While randomly selecting the 7 points, we may not select the 7 best points that should give us the accurate result.

(ii) We are only using 7 points to generate the Fundamental Matrix, unlike the 8-point algorithm case where we are using all the points (making an over-constrained system) to generate F.

Below are the 3 fundamental matrix outputs from the seven-point algorithm (each F corresponding to a different value of alpha).

F(:,:,1) =

-0.0000   0.0000   0.0025

-0.0000 -0.0000 0.0000

-0.0024 -0.0000 -0.0101

F(:,:,2) =

0.0000 -0.0000 -0.0077

0.0000 0.0000 -0.0004

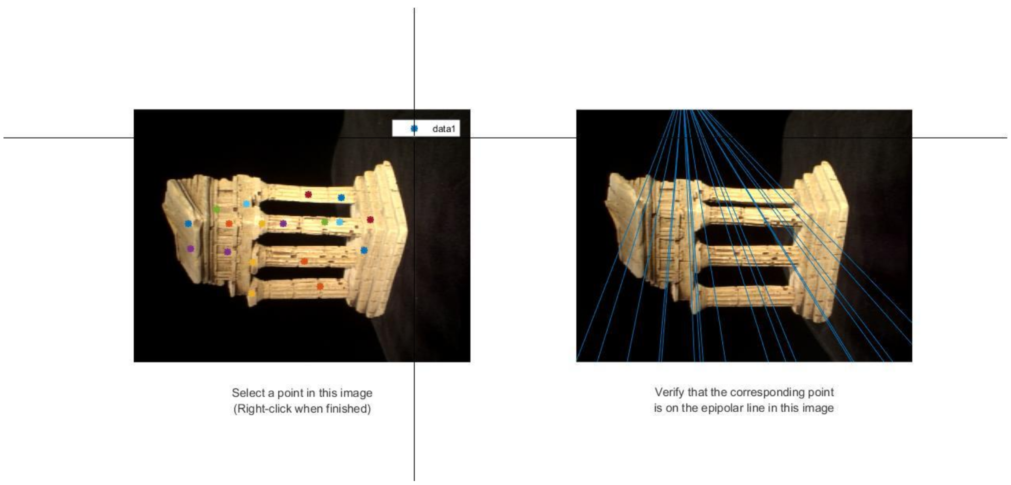
0.0076 0.0005 -0.0064

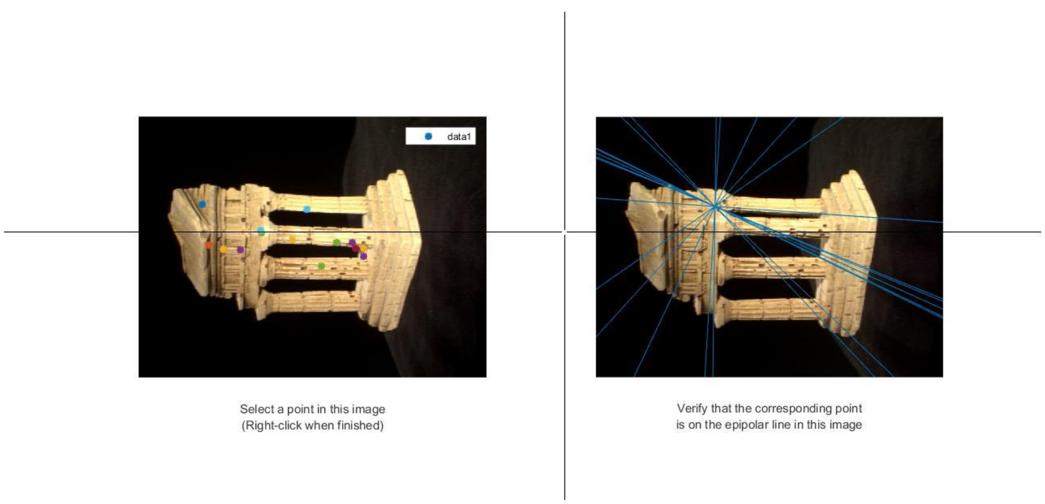
F(:,:,3) =

-0.0000 0.0000 0.0226

-0.0000 -0.0000 0.0005

-0.0217 -0.0001 -0.0925





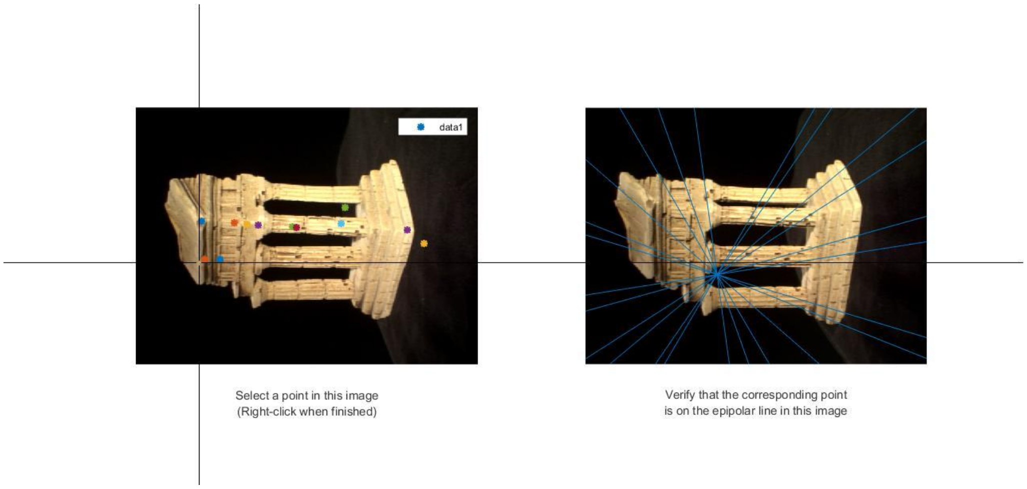


Figure 2: Output of displayEpipolar.m for 7-point algorithm (each image corresponding to 3 different values of F, as explained above)

But, if we use all the points from ‘some_corresp.mat’ with seven-point, we get very accurate results. Below are the 3 fundamental matrix outputs from the seven-point algorithm (each F corresponding to a different value of alpha).

F(:,:,1) =

```
  -0.0000    0.0000   -0.0023
   0.0000   -0.0000   -0.0000
   0.0023   -0.0000    0.0094
```

F(:,:,2) =

```
   0.0000   -0.0000    0.0053
   0.0000   -0.0000   -0.0009
  -0.0051    0.0020   -0.1318
```

F(:,:,3) =

```
   0.0000   -0.0000    0.0011
   0.0000   -0.0000   -0.0008
  -0.0012    0.0012   -0.0318
```

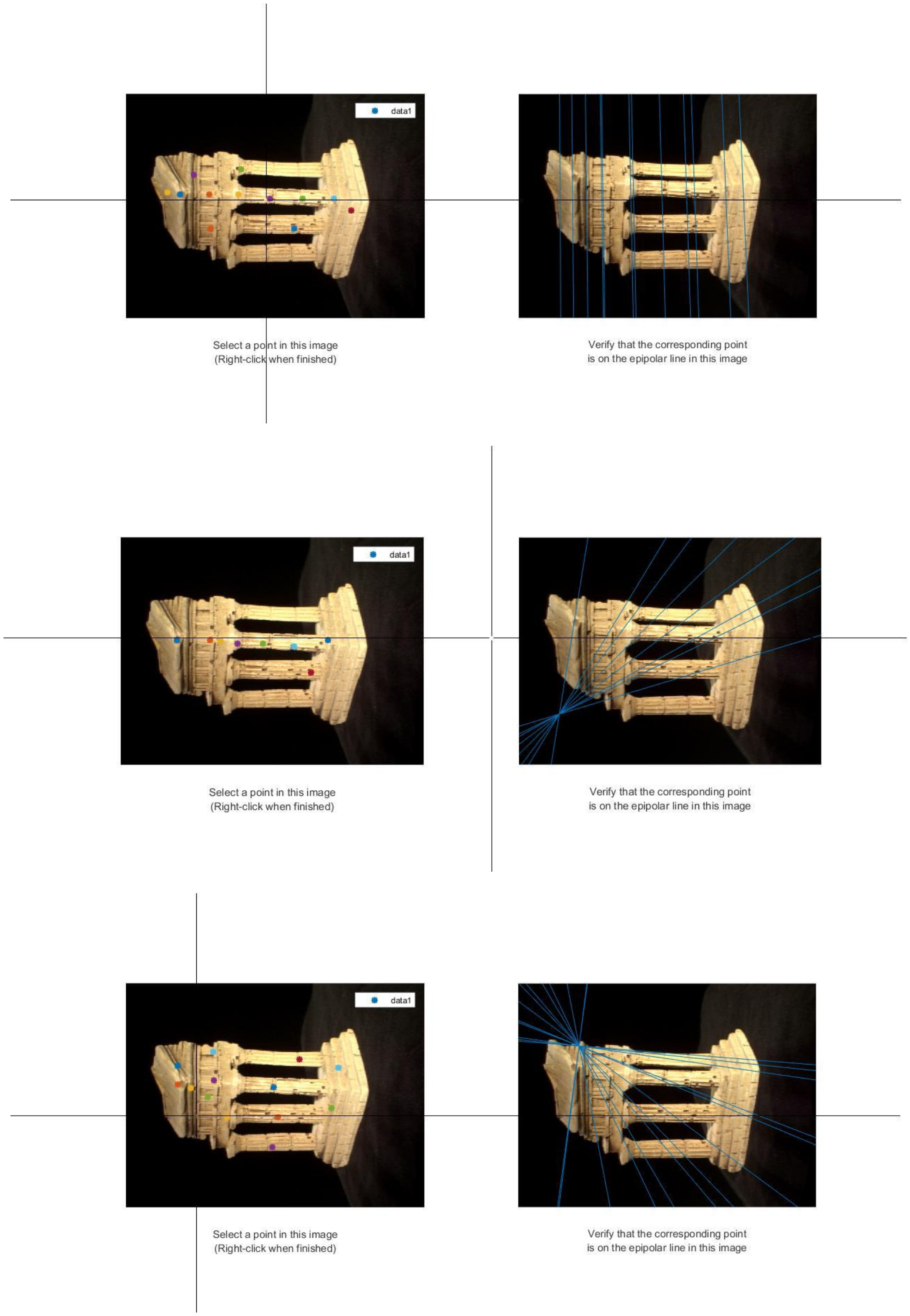


Figure 3: Output of displayEpipolar.m for 7-point algorithm, all points considered to evaluate F, making it an over-constrained system (each image corresponding to 3 different values of F, as explained above)

2.X) Extra Credit, Fundamental Matrix Estimation: Seven Point Algorithm (RANSAC)

We use RANSAC and the seven-point algorithm to find the Fundamental Matrix with maximum inliers, where points are randomly selected from 'some_corresp_noisy.mat'. Below are the 3 fundamental matrix outputs from the seven-point algorithm (each F corresponding to a different value of alpha). All the 3 different Fundamental Matrices mentioned below gave very accurate results.

F(:,:,1) =

-0.0000 0.0000 0.0025

-0.0000 -0.0000 0.0000

-0.0024 -0.0000 -0.0101

F(:,:,2) =

0.0000 -0.0000 -0.0077

0.0000 0.0000 -0.0004

0.0076 0.0005 -0.0064

F(:,:,3) =

-0.0000 0.0000 0.0226

-0.0000 -0.0000 0.0005

-0.0217 -0.0001 -0.0925

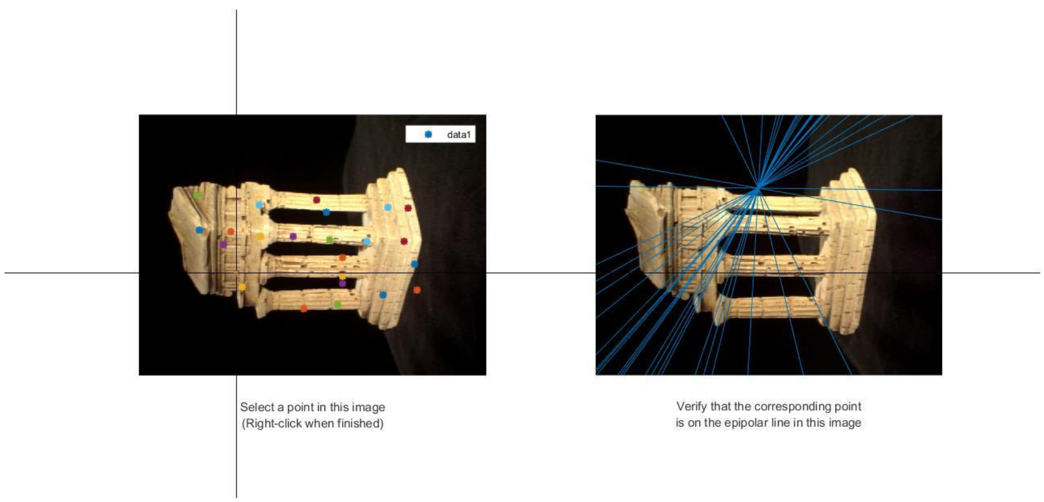


Figure 4: Output of displayEpipolar.m for 7-point algorithm, with noisy data, without RANSAC

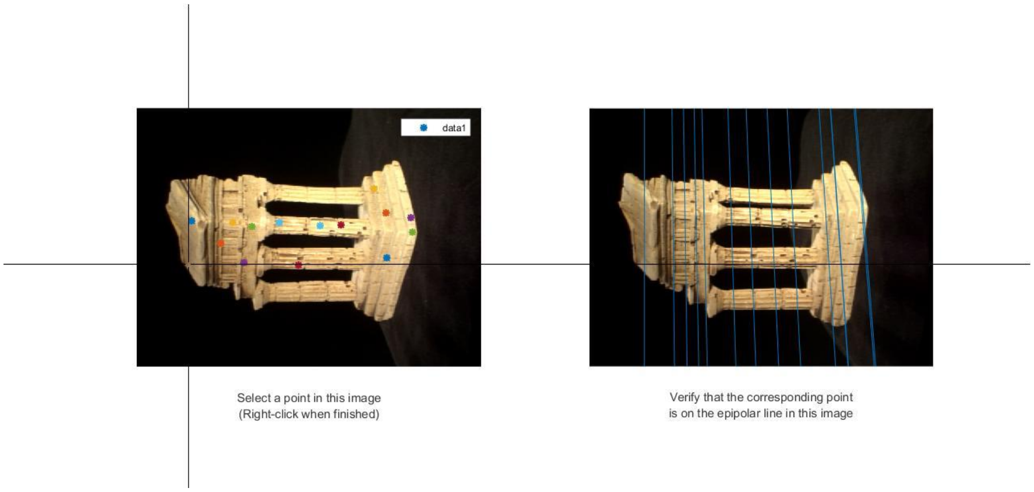


Figure 5: Output of displayEpipolar.m for 7-point algorithm, with noisy data, with RANSAC

RANSAC: For the implementation of RANAC I used an algorithm similar to what we had implemented for HW-2. I randomly select 7-points and generate a Fundamental Matrix and compute the expression '$x_1^T*F*x_2$' for all the correspondence points. Ideally, this expression should evaluate to zero for all the inliers, but I consider all the points that generate the value of this expression below my tolerance (0.01) as inliers. I rerun this for '100' iterations and save the best F and all the inlier points. Later, I re-generate F using all the inliers, that gives me very accurate results even for the noisy data.

2.3) Essential Matrix

F =

0.0000	-0.0000	0.0165
-0.0000	0.0000	-0.0003
-0.0158	0.0006	-0.0659

E =

0.0387	-5.7210	24.1139
-0.6796	0.1333	-0.5897
-24.1692	-0.2318	-0.0089

2.5) Best M2

M2 =

1.0e+03 *

$$\begin{matrix} 1.5202 & -0.0275 & 0.3019 & -0.0287 \\ -0.0510 & 1.4096 & 0.6321 & -1.5190 \\ 0.0000 & -0.0003 & 0.0010 & 0.0000 \end{matrix}$$

2.6) Epipolar Correspondences

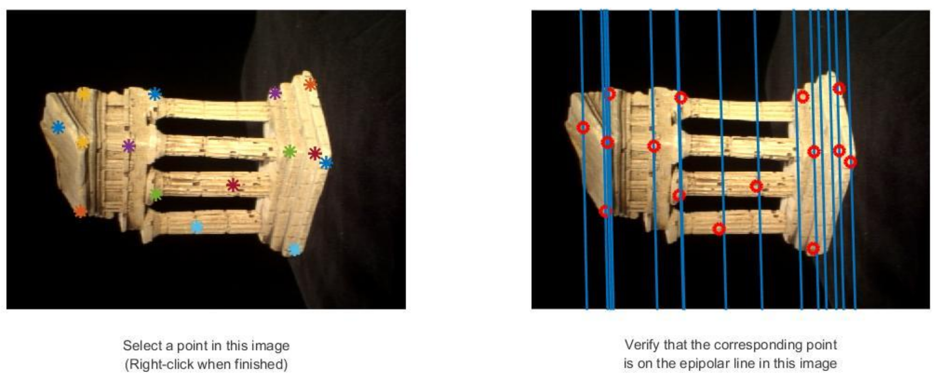


Figure 6: Output of epipolarMatchGUI.m for manually selected points and detected correspondences

To generate the Epipolar Correspondences, I used a Gaussian Weighting window to provide greater influence on the center. Besides, I also searched on a limited window on the epipolar line, considering the images provided are not too different. The values of standard deviation for the Gaussian window, window and, patch size have been tuned manually to provide best results.

2.7)

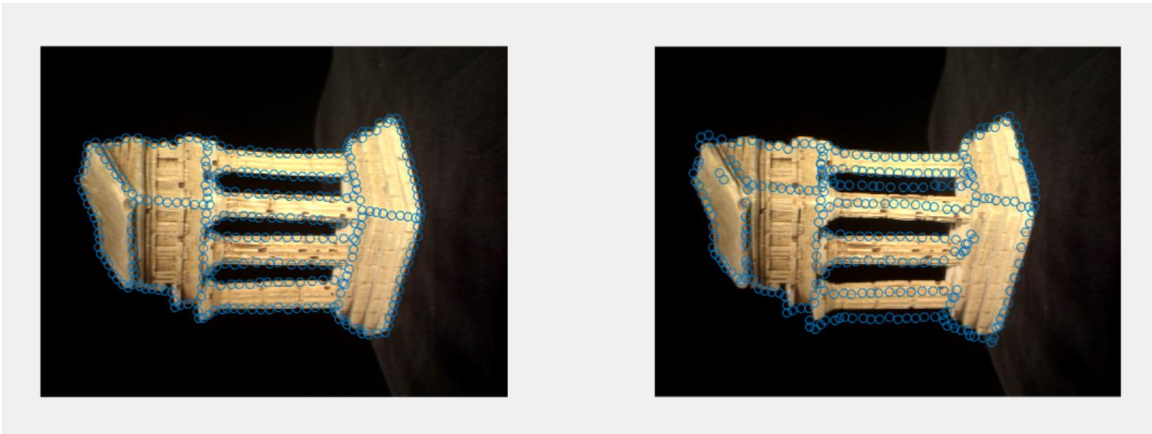


Figure 7: Image on the left is im1 and scatter of selected points pts1 and image on the right is the im2 scatter of correspondences generated

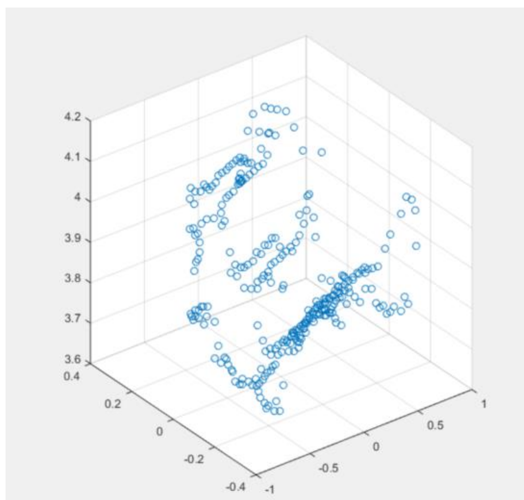


Figure 8: 3-D visualization of the point cloud generated

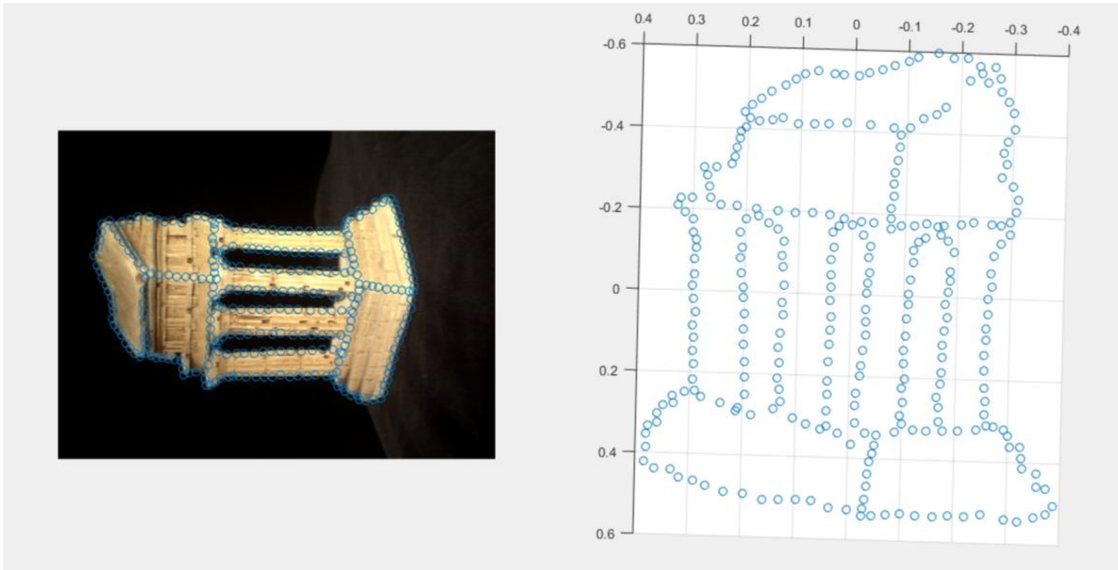


Figure 9: Image on the left is im1 and on right is the 3-D visualization of the point cloud generated, front view

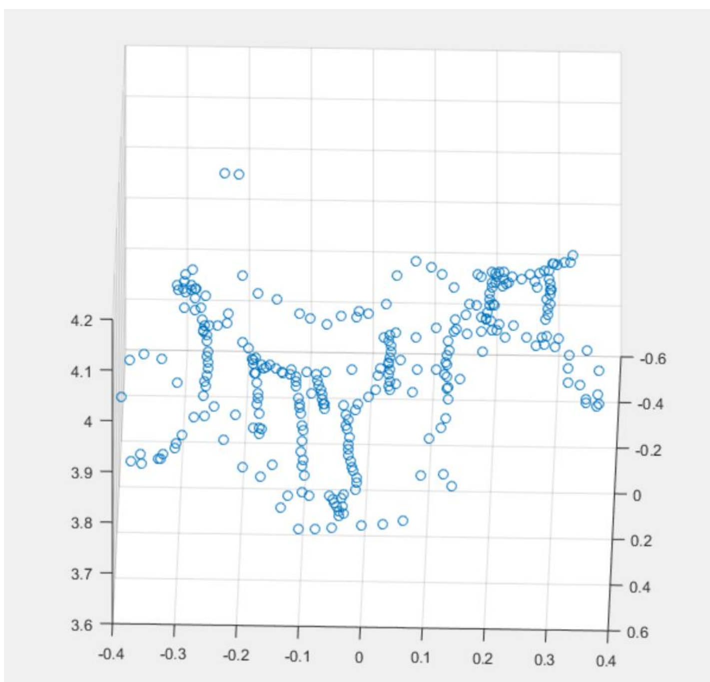


Figure 10: 3-D visualization of the point cloud generated, view from the bottom

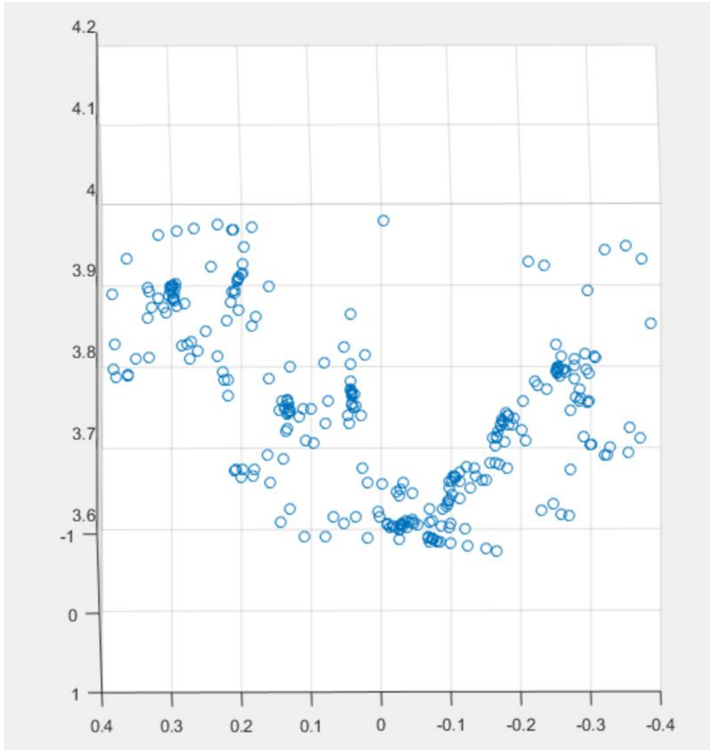


Figure 11: 3-D visualization of the point cloud generated, top view