Computer Vision, 16720A - Homework #4

Neeraj Basu
neerajb@andrew.cmu.edu

November 11, 2020

Q 1.1 - Theory

Looking at the equation from the eight point algorithm:

$$P_l^T F P_r = 0$$

$$p_l = \begin{bmatrix} x_l \\ y_l \\ 1 \end{bmatrix}, p_r = \begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix}, F = \begin{bmatrix} F_{11}, F_{12}, F_{13} \\ F_{21}, F_{22}, F_{23} \\ F_{31}, F_{32}, F_{33} \end{bmatrix}$$

Since we have normalized the image coordinates such that the coordinate origin cooresponds with the principle point, the new set of equations will look like:

$$p_l = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, p_r = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, F = \begin{bmatrix} F_{11}, F_{12}, F_{13} \\ F_{21}, F_{22}, F_{23} \\ F_{31}, F_{32}, F_{33} \end{bmatrix}$$

Multiplying these together:

$$\begin{bmatrix} 0, 0, 1 \end{bmatrix} \begin{bmatrix} F_{11}, F_{12}, F_{13} \\ F_{21}, F_{22}, F_{23} \\ F_{31}, F_{32}, F_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

$P_l^T F$

$$\begin{bmatrix} F_{31}, F_{32}, F_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

$F P_r$

$$F_{33} = 0$$

Q 1.2 - Theory

Since the only difference between our two camera views is a pure translation, we can represent the transformation from the first camera to the second as:

$$p_r = \begin{bmatrix} R \; t \end{bmatrix} p_l$$

Where R is the identity matrix (since there is no rotation) and t is a column vector representing translation solely in the x-axis (t_x). We solve for p_r in terms of translation and p_L as such:

$$p_r = \begin{bmatrix} 1, 0, t_x \\ 0, 1, 0 \\ 0, 0, 1 \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ 1 \end{bmatrix}$$

Simplifying:

$$p_r = \begin{bmatrix} x_l + t_x \\ y_l \\ 1 \end{bmatrix}$$

We now have two points, p_L and p_r which the epipolar line crosses through. The only difference is the translation by t_x.

$$p_r - p_l = \begin{bmatrix} x_l + t_x \\ y_l \\ 1 \end{bmatrix} - \begin{bmatrix} x_l \\ y_l \\ 1 \end{bmatrix} = \begin{bmatrix} t_x \\ 0 \\ 0 \end{bmatrix}$$

We can see this resulting difference vector is of the form of a 1-D line in the x-axis, meaning the line is parallel to the x-axis.

Q 1.3 - Theory

For this proof, I will denote T_i, p_i as the current time step and cooresponding point and T_j, p_j as the next time step and corresponding point.

We can use the corresponding translation and rotation vectors for each time step to form the equations at T_i and T_j as follows:

$$p_i = [R_i \; t_i]P$$
$$p_j = [R_j \; t_j]P = [R_{rel} \; t_{rel}]P$$

Since we are interested in the essential and fundamental matrix of the T_j time step we can solve for E using:

$$E = [t_{rel}] \; R_{rel}$$

We can now solve for F using the equation:

$$F = K^{-T}EK^{-1}$$

Since we are using the same camera from T_i to T_j, we will only have one K matrix to deal with. Plugging in our E from above we end with:

$$F = K^{-T}[t_{rel}] \; R_{rel}K^{-1}$$

Here t_{rel} is represented as the difference in translation vectors from T_j to T_i. For R_{rel}, we need to be careful to translate back to the origin before applying the rotation at T_j - other wise we will end up rotating too far.

$$t_{rel} = t_j - t_i$$
$$R_{rel} = R_j R_i^{-1}$$

Q 1.4 - Theory

Please note these papers were used in order to complete this proof:

Planar mirrors for image-based robot localization and 3-D reconstruction

Mirror Symmetry to 2-View Stereo Geometry

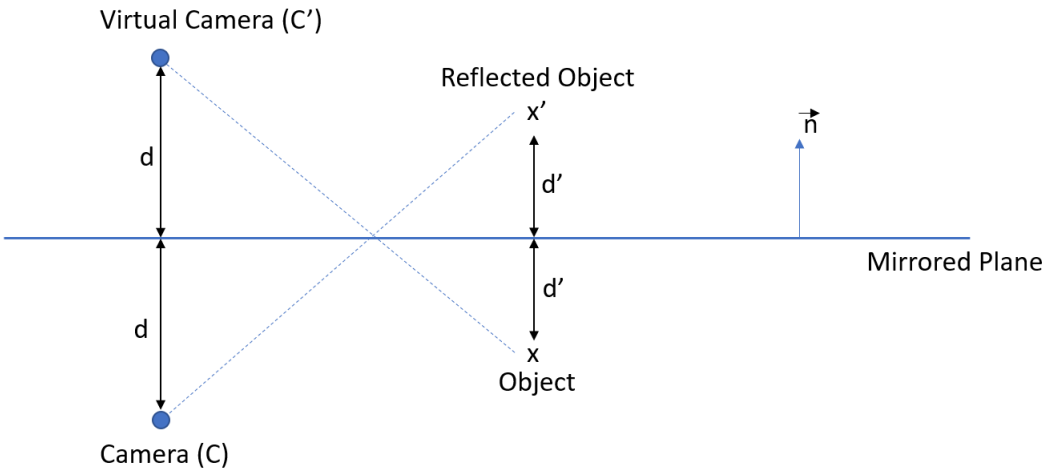

Figure 1: Diagram for Q1.4

Here I am choosing the representation where x' is point x mirrored over the mirrored plane. d represents the distance from the mirrored plane to the camera and virtual camera and d' represents the distance from the mirrored plane to the object and reflected object.

Using the Householder equation defined in the first paper, we can create a matrix R which describes the rotation around a plane. This rotation will define the transformation from C to C' over the mirrored plane.

$$R = I - 2nn^T$$

Where n is the components of the projection vector. For this example, I will define n as:

$$n = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

We can now define our rotation matrix using n and the equation for R above:

$$R = \begin{bmatrix} 1-2x^2, & -2xy, & -2xz \\ -2xy, & 1-2y^2, & -2yz \\ -2xz, & -2yz, & 1-2z^2 \end{bmatrix}$$

We can also define the translation between the real and virtual camera using the equation:

$$t = 2dn^T$$

Representing t as a skew-symmetric matrix we get:

$$[t]_x = 2d \begin{bmatrix} 0, & -z, & y \\ z, & 0, & -x \\ -y, & x, & 0 \end{bmatrix}$$

This allows us to now use the equation of:

$$E = [t]_x R$$

to solve for essential matrix.

$$E = [t]_x R = 2d \begin{bmatrix} 0, & -z, & y \\ z, & 0, & -x \\ -y, & x, & 0 \end{bmatrix} \begin{bmatrix} 1-2x^2, & -2xy, & -2xz \\ -2xy, & 1-2y^2, & -2yz \\ -2xz, & -2yz, & 1-2z^2 \end{bmatrix}$$

Simplifying we see that the resulting E is a skew-symmetric matrix meaning $E^T = -E$:

$$E = 2d \begin{bmatrix} 0, & -z, & y \\ z, & 0, & -x \\ -y, & x, & 0 \end{bmatrix}$$

Knowing that E is a skew-symmetric matrix, we can now use the equation of F to see if the property of $F^T = -F$ holds true:

$$F = K^{-T} E K^{-1}$$
$$F^T = (EK^{-1})^T K^{-1} = K^{-T} E^T K^{-1} = -K^{-T} E K^{-1} = -F$$

We can see that both equations hold true:

$$E^T = -E$$
$$F^T = -F$$

Proving a reflection around a mirrored plane is equivalent to having a skew-symmetric fundamental matrix.

Q 2.1 - Fundamental Matrix Estimation

$$F = \begin{bmatrix} 0 & -0.00000013 & 0.00112586 \\ -0.00000006 & 0 & -0.00001176 \\ -0.00108269 & 0.00003048 & -0.00447033 \end{bmatrix}$$

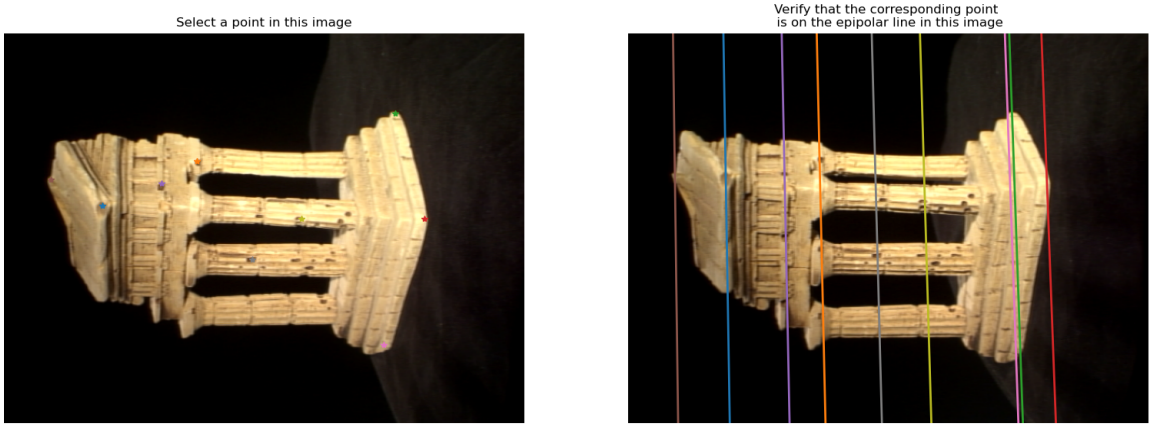

Figure 2: Example Output of Q2.1

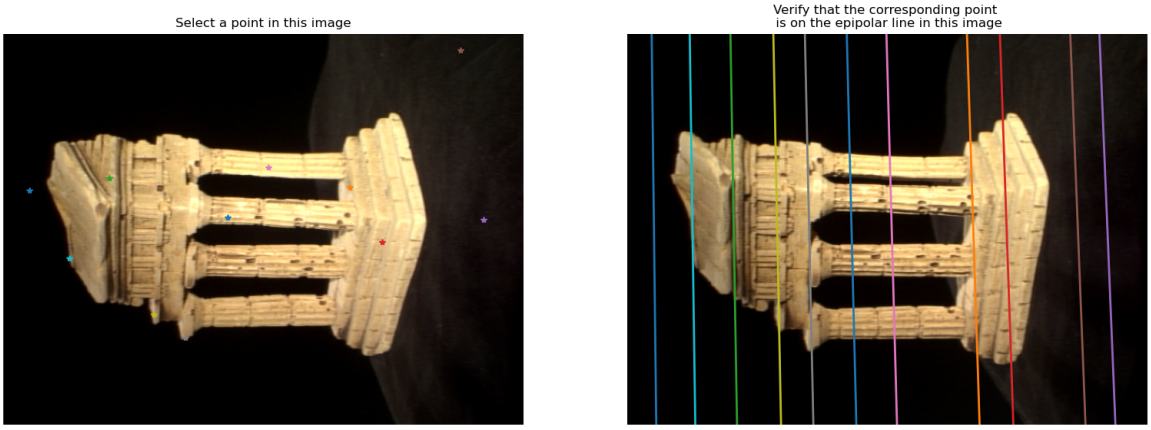

Figure 3: Another Example Output of Q2.1

Q 3.1 - essentialMatrix

The estimated fundamental and essential matrix from my eight-point algorithm are as follows:

Fundamental Matrix:

$$F = \begin{bmatrix} 0 & -0.00000013 & 0.00112586 \\ -0.00000006 & 0 & -0.00001176 \\ -0.00108269 & 0.00003048 & -0.00447033 \end{bmatrix}$$

Essential Matrix:

$$E = \begin{bmatrix} 0.00226269 & -0.3065525 & 1.66260633 \\ -0.13313041 & 0.00691061 & -0.04330034 \\ -1.6672107 & -0.01332104 & -0.00067219 \end{bmatrix}$$

Q 3.2 - triangulate

To solve for our 3D points w_i (4x1) we need to solve for the equation:

$$C_1 w_i = x_i$$

Here C_1 is our camera matrix (3x4), and x_i (3x1) is our correspondence points. Assume we are only looking at a single camera right now. We can rearrange this equation such that:

$$(x_i \times C_1) w_i = 0$$

Let's denote $x_i \times C_1$ as $A1_{temp}$. The resulting cross product will result in a (3x4) vector. However, since there is a linear dependence in this matrix, we can drop one of the duplicate rows. This will ultimately transform $A1_{temp}$ into a (2x4) matrix. However since our objective it to solve for w_i, we do not have enough equations to solve this system of equations.

If we repeat this process again for the second camera, $C2$, we will generate another (2x4) $A2_{temp}$ matrix. By stacking these two vectors on top of each other, we will have an A_i (4x4) vector which will be sufficient to solve for w_i.

$$\begin{bmatrix} A1_{temp} \\ A2_{temp} \end{bmatrix} w_i = 0$$

$$\begin{bmatrix} A1_{temp} \\ A2_{temp} \end{bmatrix} = A_i$$

$$A_i w_i = 0$$

By taking the SVD of A and taking the eigen vector cooresponding to the lowest eigenvalue, we can solve for our w_i (4x1) vector. By taking the projection of w_i with the respective camera matrix, we can get our 3D coordinate estimate in the world plane.

Q 4.1 - epipolarCoorespondence

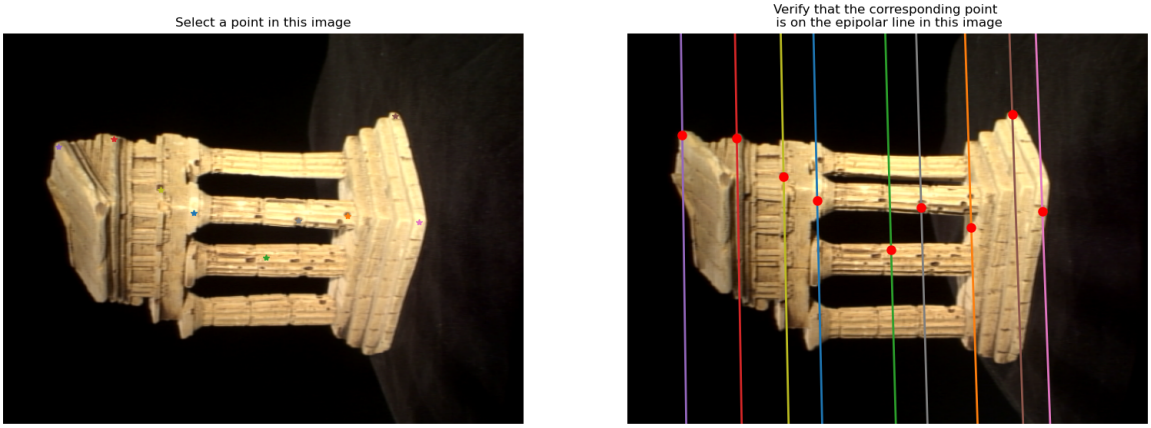

Figure 4: Example Output of Q4.1

Please note for 4.1, I printed the input (x,y) and output (x,y) to console and manually wrote a script to save this to the npz file.

Q 4.2 - 3D Visualization

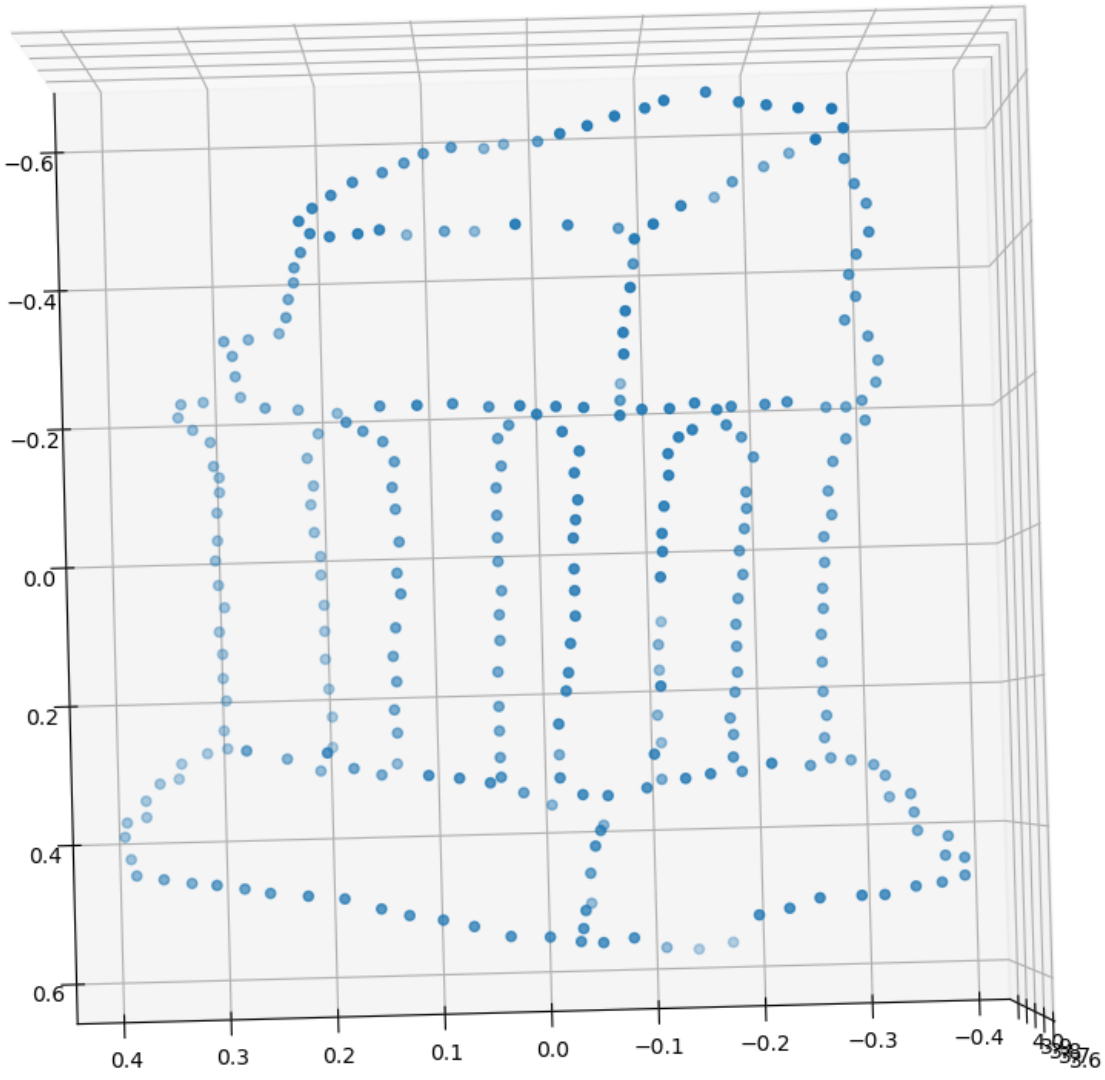

Figure 5: Example Output of Q4.2

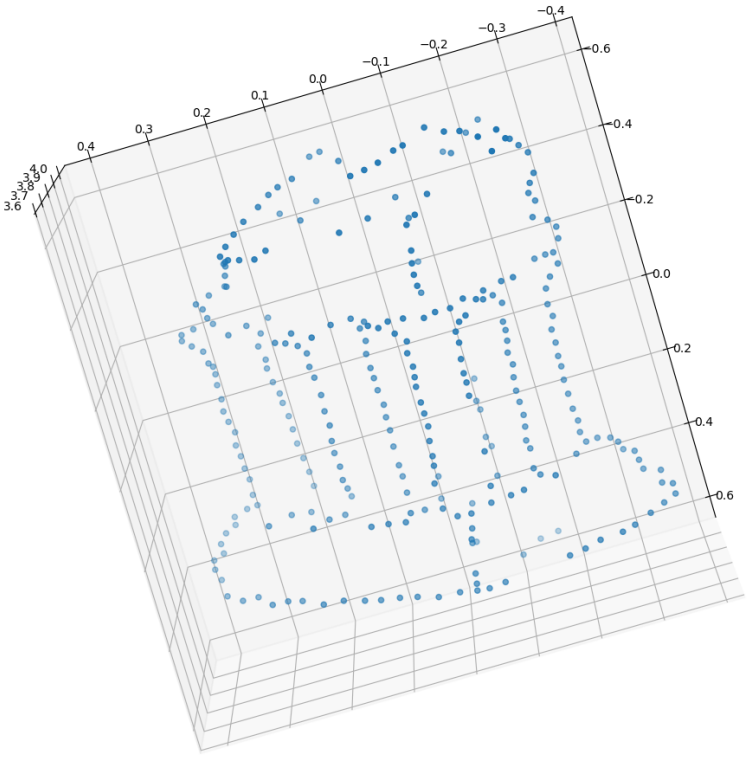

Figure 6: Another Example Output of Q4.2

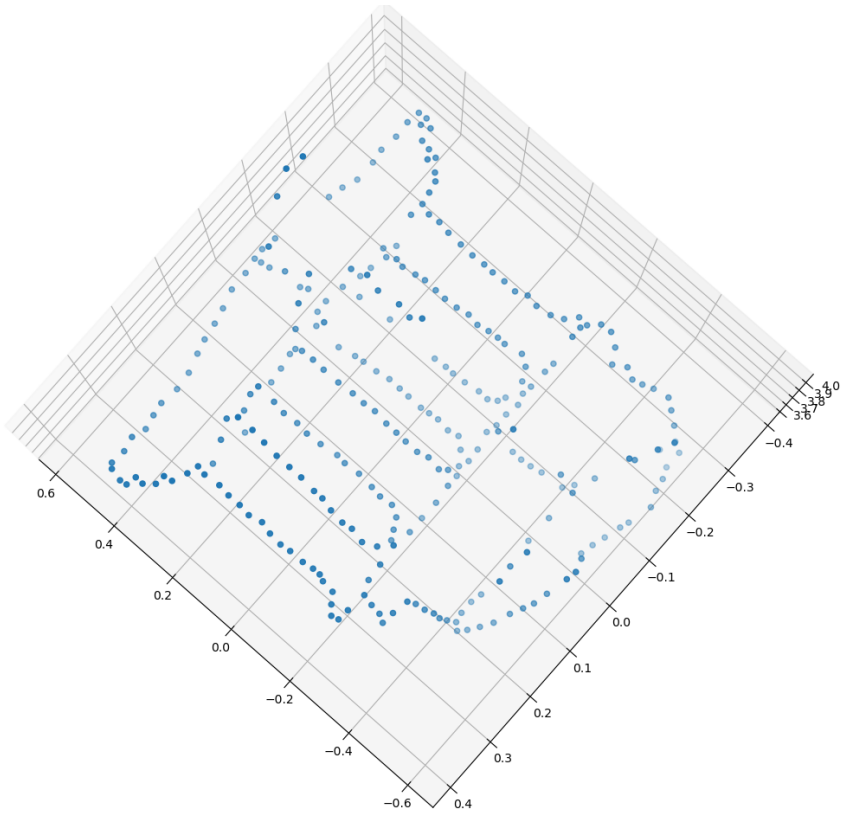

Figure 7: Another Example Output of Q4.2

Q 5.1 - Bundle Adjustment (Extra Credit)

Testing the noisy correspondences matrix we can see we fail to produce an effective F.

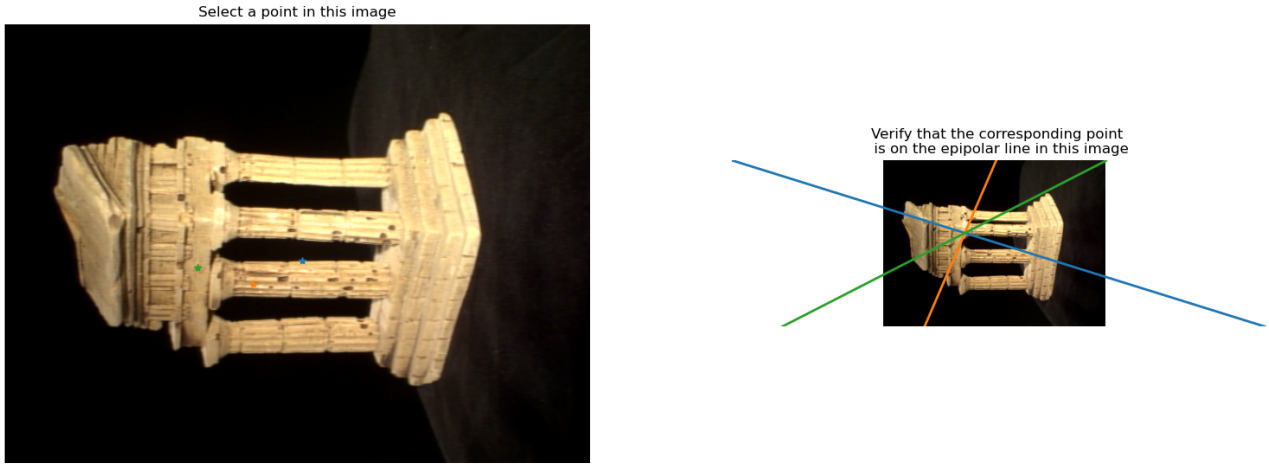

Figure 8: Example Output of Noisy Correspondences Q5.1

RANSAC significantly improves the performance of the generation of epipolar lines and their corresponding point. I was able to improve the performance by increasing the number of iterations. This gives RANSAC more opportunity to find the best F. The error term was calculated by taking the geometric distance between the point and the epipolar line as shown in the lecture slides. If this distance was below the tolerance threshold, that point is added as an inlier.

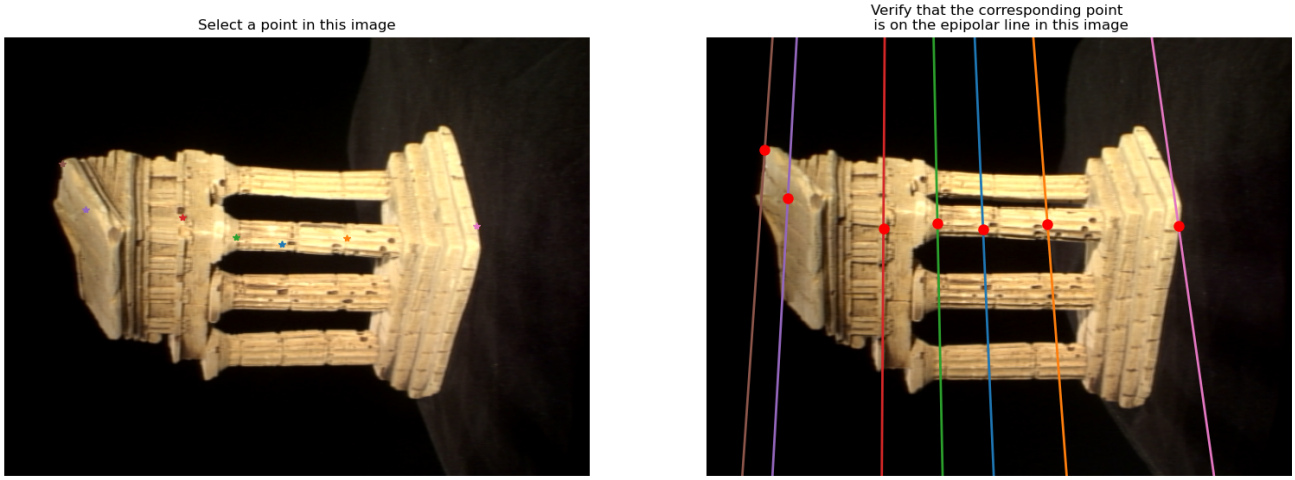

Figure 9: Example Output of Q5.1 (tol = .42, nIter = 100)

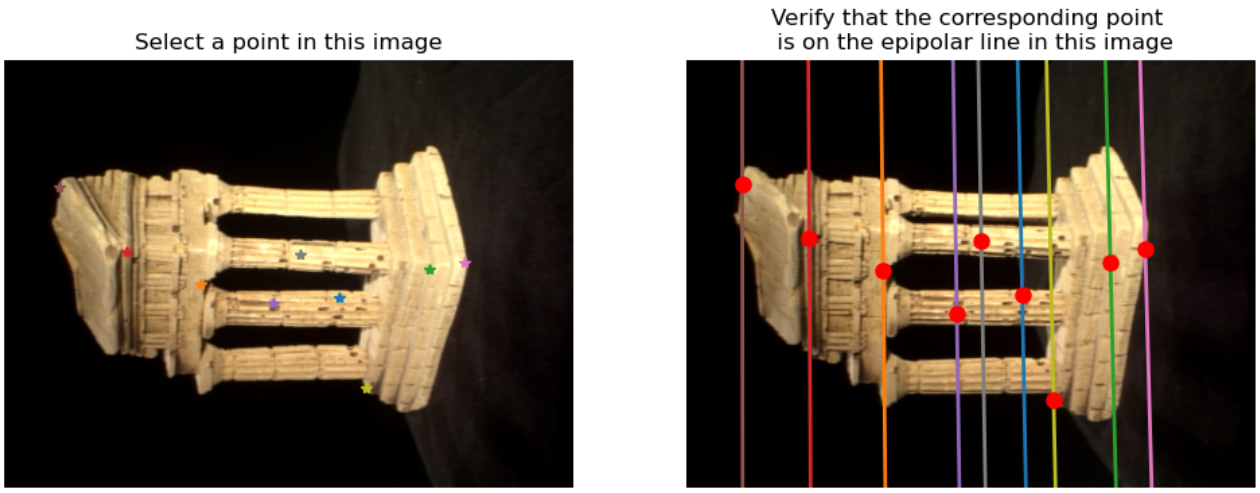

Figure 10: Improved Output of Q5.1 (tol = .42, nIter = 200)

Q 5.2 - rodrigues, invRodrigues (Extra Credit)

I was able to implement the rodrigues and invRodrigues function although I was not able to test the whole system. What I was able to prove was that when passing a vector:

$$r = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

I can first pass in the vector r to rodrigues and return R (3x3). Then by passing R into invRodrigues I can get the same r back out.

As an example:

$$input\ r = \begin{bmatrix} 1/2 \\ 1/3 \\ 1/4 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.91621582, -0.15228682, 0.37061746 \\ 0.31315245, 0.84918847, -0.42522286 \\ -0.24996824, 0.50565568, 0.8257289 \end{bmatrix}$$

$$output\ r = \begin{bmatrix} 1/2 \\ 1/3 \\ 1/4 \end{bmatrix}$$