Mini-Project 4
Camera Calibration

in this mini project we were able through image points and their equivalents to obtain the projection matrix thus defining how to project object coordinates from the real world to the image domain. The projection matrix is the result of multiplication of both the intrinsic and extrinsic matrices.

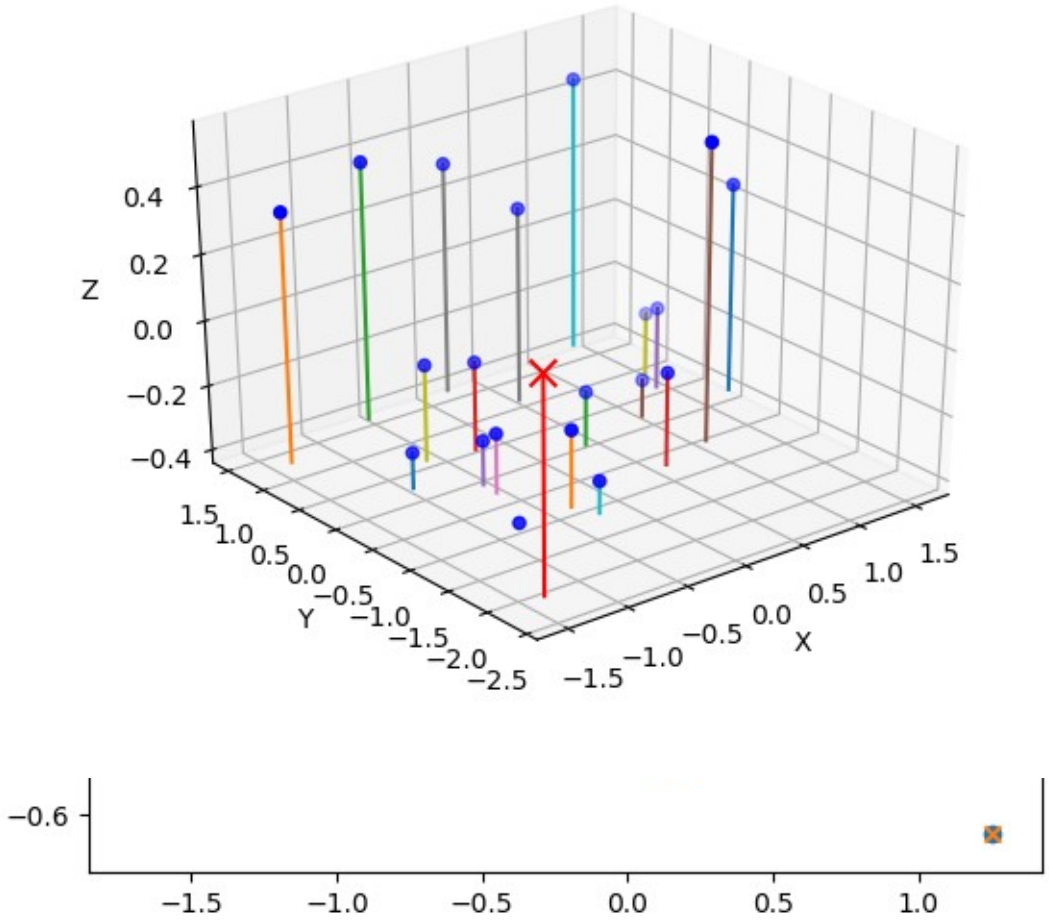
$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \cong \begin{pmatrix} su \\ sv \\ s \end{pmatrix} = M \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

As shown in the image above, it is clear that the projection matrix consists of 12 parameters and to have a unique solution we can fix the scale by setting element m34 to 1 and solve for the rest of the elements using a normal linear approach.

$$AU = B$$

another approach to solve for the projection matrix is the total linear least square. In this approach we do not need to set element M34 to 1 but we will need to use single value decomposition.

The projection method part results:



the projected image resulted form the points given was extremely close to the normalized projection matrix in the project description PDF.

```
The projection matrix is:
 [[ 0.76785834 -0.49384797 -0.02339781  0.00674445]
 [-0.0852134  -0.09146818 -0.90652332 -0.08775678]
 [ 0.18265016  0.29882917 -0.07419242  1.        ]]

The total residual is:
 0.04453499394931152

'
```

later in this project we used the projection matrix to obtain the center of the image

$$M = (Q|m_4)$$

$$C = -Q^{-1}m_4$$

Also, Our camera center is nearly identical to the one in the project description PDF.

```
The estimated location of the camera is:
 [-1.51263977 -2.35165965  0.28266502]
```

Latter in this project we calculated the fundamental matrix which defines lines of interest on which a correspondents to a points in Image A lies in Image B. to solve for the fundamental matrix we used an 8 point algorithm, in which we used 8 match points, single value decomposition and setting the rank of the fundamental matrix to two.

$$(f_{11}uu' + f_{12}vu' + f_{13}u' + f_{21}uv' + f_{22}vv' + f_{23}v' + f_{31}u + f_{32}v + f_{33}) = 0$$

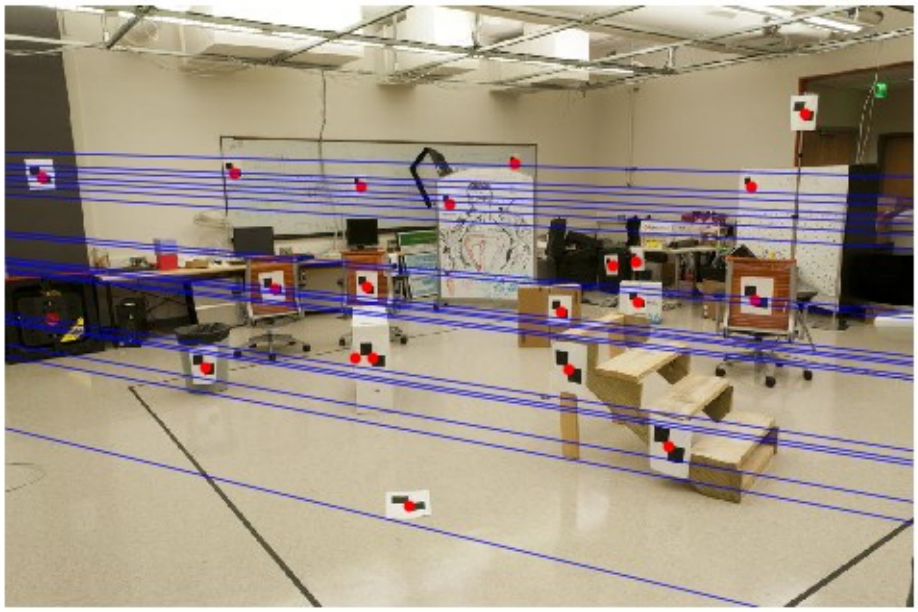
```
u, s, vh = np.linalg.svd(np.array(U),full_matrices=True)
F=vh[-1,:]
F= np.reshape(F,(3,3))

u_f,s_f,vh_f =np.linalg.svd(F)
s_f[-1]=0
F_matrix = np.matmul(np.matmul(u_f,np.diag(s_f)),vh_f)
```

Fundamental Matrix Results:

```
[[ 2.10563974e-07  2.43406151e-06  8.27631182e-06]
 [ 8.05751128e-06  5.29133596e-06  1.18169057e-02]
 [-1.11817413e-03 -2.08984268e-02  9.99711141e-01]]
```

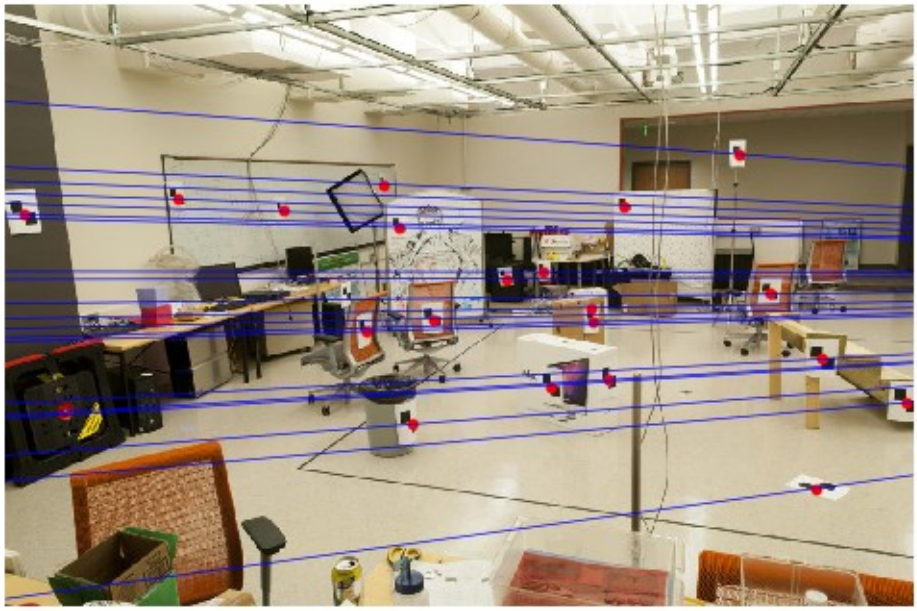
We



also utilized a better Normalized form of the Fundamental matrix which resulted in more accurate outputs which can be deduced from the images as in the normalized forms more epipolar lines path through the points of interest. To calculate the Normalized Fundamental matrix we used a normalized form of 8 image points from image B and Image A, and used single value decomposition along with forcing the matrix to have a rank of 2 as before.

$$Af = \begin{bmatrix} \hat{x}'_1\hat{x}_1 & \hat{x}'_1\hat{y}_1 & \hat{x}'_1 & \hat{y}'_1\hat{x}_1 & \hat{y}'_1\hat{y}_1 & \hat{y}'_1 & \hat{x}_1 & \hat{y}_1 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \hat{x}'_8\hat{x}_8 & \hat{x}'_8\hat{y}_8 & \hat{x}'_8 & \hat{y}'_8\hat{x}_8 & \hat{y}'_8\hat{y}_8 & \hat{y}'_8 & \hat{x}'_8 & \hat{y}_8 & 1 \end{bmatrix} f = 0$$

```
[[-2.02478523e-07  2.78039315e-06 -6.94781145e-04]
 [ 1.92581510e-06 -4.74398800e-07  5.59813726e-03]
 [-4.16157075e-05 -7.69192568e-03  1.78588215e-01]]
```

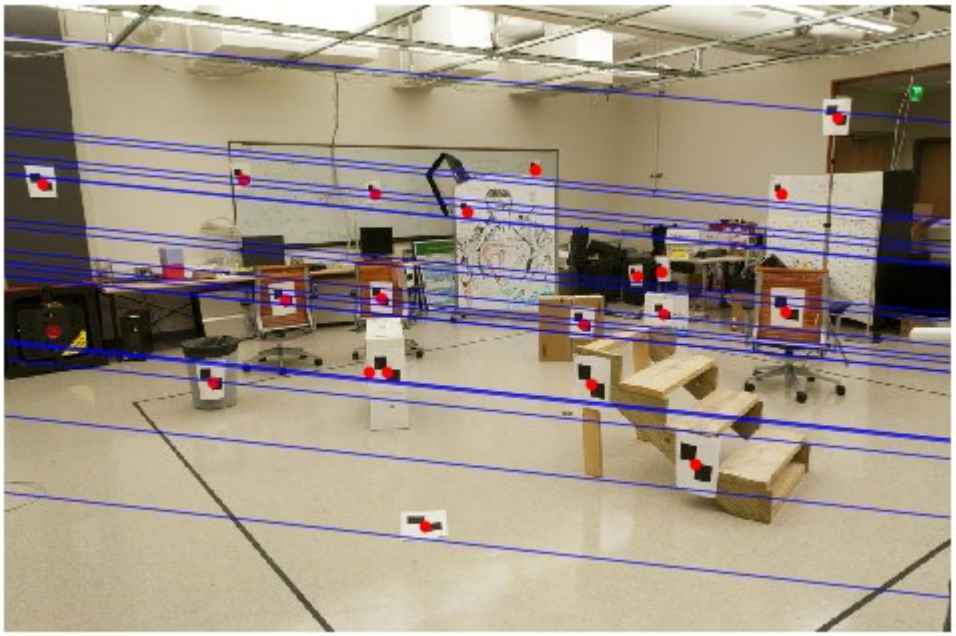


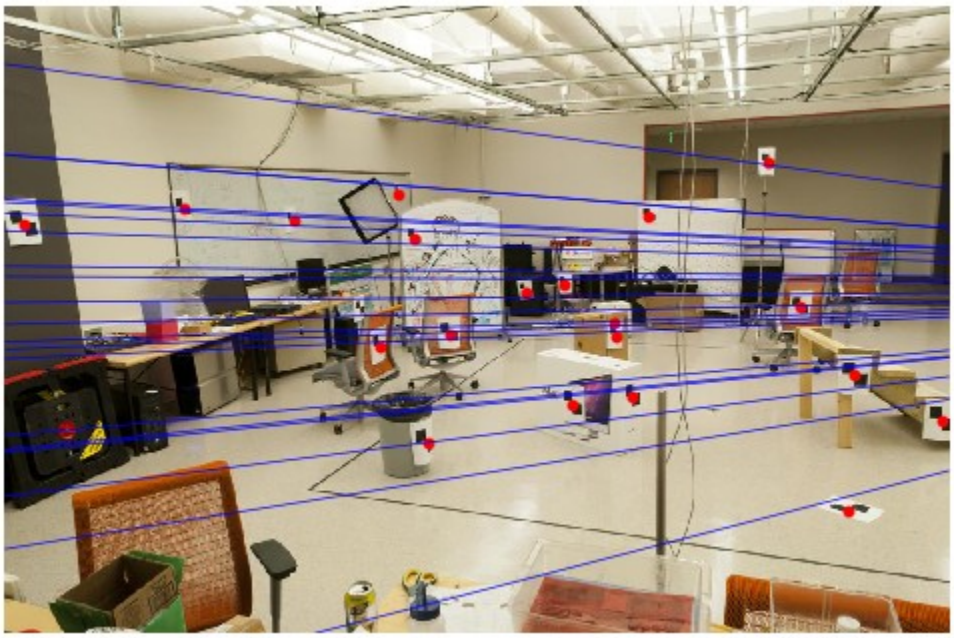
Finally in this project we used Ransac algorithm to estimate a fundamental matrix satisfying as much points as possible. To run our algorithm we need to set a parameter to indicate which points to consider inliers and which points to exclude. Also, we set the number of iterations our code can run and finally we filter the best results. Also, we added noise to our ground truth data to test our code against the data. We used the normal fundamental matrix in the beginning but later we used the normalized fundamental matrix as it produced better results. In addition, increasing the number of iterations and decreasing the tolerance usually leads to better results. Further more, the ground truth for the images had some significant error in them leading to the difficulty in generating a proper estimation for the fundamental matrix. Finally accounting for the count of inliers only resulted in unsatisfactory results similar to adding no noise to the images, both resulted for bad images because the ground truth points were wrong and the count did represent the error within the inliers. These issues were somehow countered by adding noise, increasing iterations and putting in consideration the sum of error of each point in the inliers.

```
if(CurrentCount>=previousCount and currentError<previousError):
```

Estimated Fundamental matrix Results on the boxes image without noise:

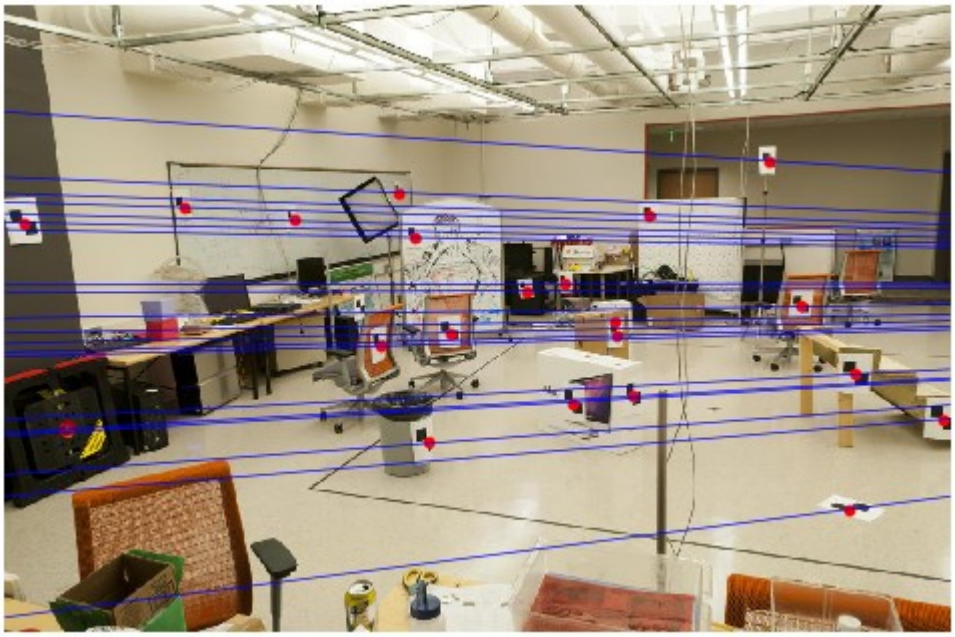
```
Estimated matrix:
[[ 2.98658642e-06 -3.00866941e-05  7.34173565e-03]
 [-3.86968494e-06  2.35381957e-05 -4.71579983e-02]
 [-4.43814513e-03  4.95949541e-02  9.97618596e-01]]
Real matrix:
[[ 2.10563974e-07  2.43406151e-06  8.27631182e-06]
 [ 8.05751128e-06  5.29133596e-06  1.18169057e-02]
 [-1.11817413e-03 -2.08984268e-02  9.99711141e-01]]
```

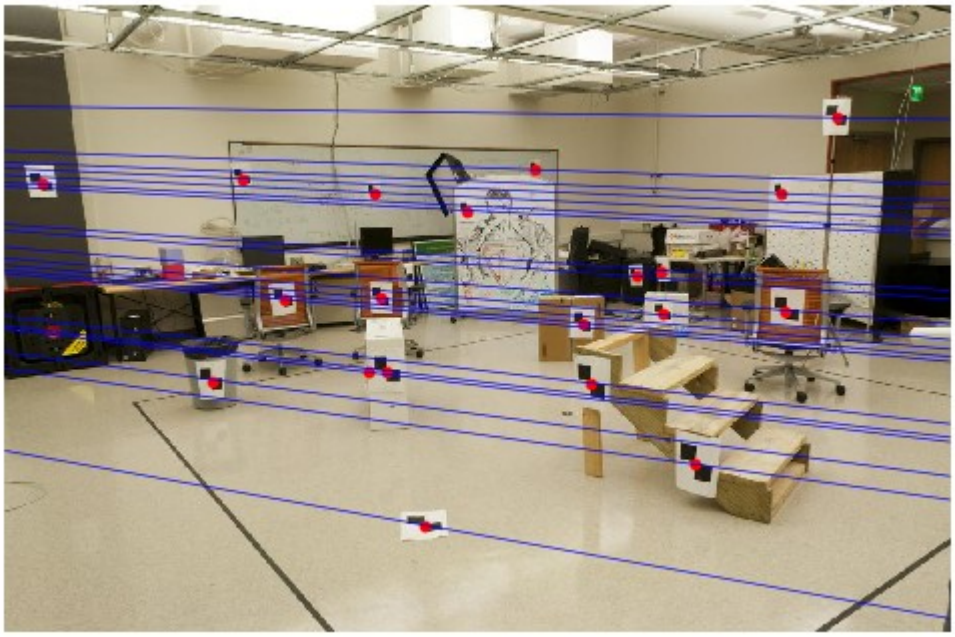




Normalized Estimated Fundamental Matrix Results without noise:

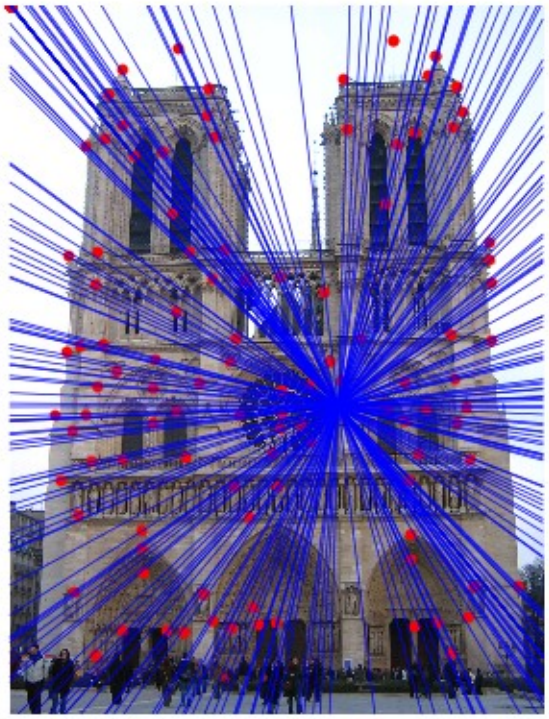
```
Estimated matrix:
[[ 1.34546364e-07 -1.95927180e-06  4.82219443e-04]
 [-1.68617885e-06  1.05243454e-07 -4.02829813e-03]
 [ 1.25008584e-04  5.82074656e-03 -1.81729694e-01]]
Real matrix:
[[ 2.10563974e-07  2.43406151e-06  8.27631182e-06]
 [ 8.05751128e-06  5.29133596e-06  1.18169057e-02]
 [-1.11817413e-03 -2.08984268e-02  9.99711141e-01]]
```

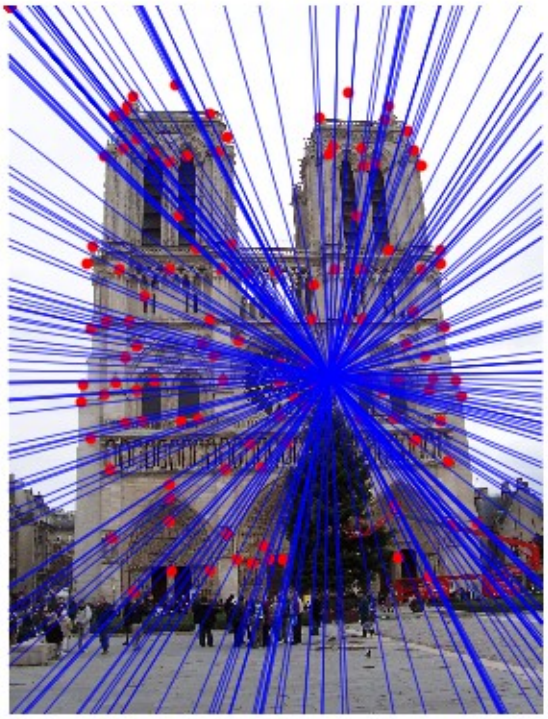


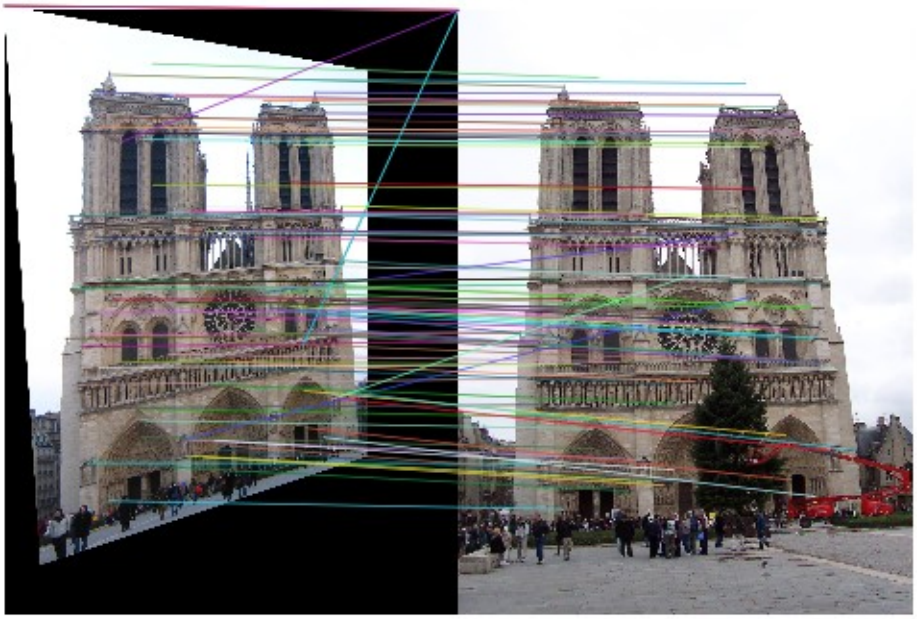


Notre Dam image with noise and using normalized fundamental matrix:

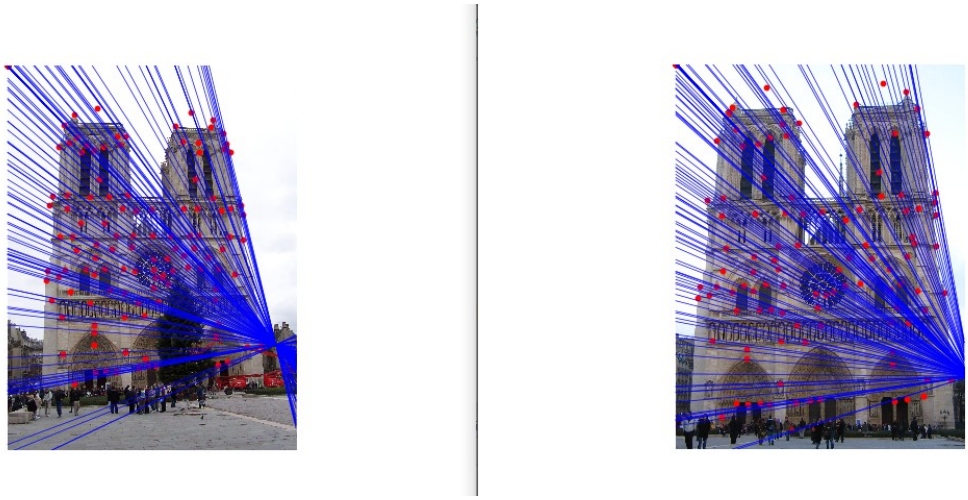
iteration = 3000
tolerance =0.1

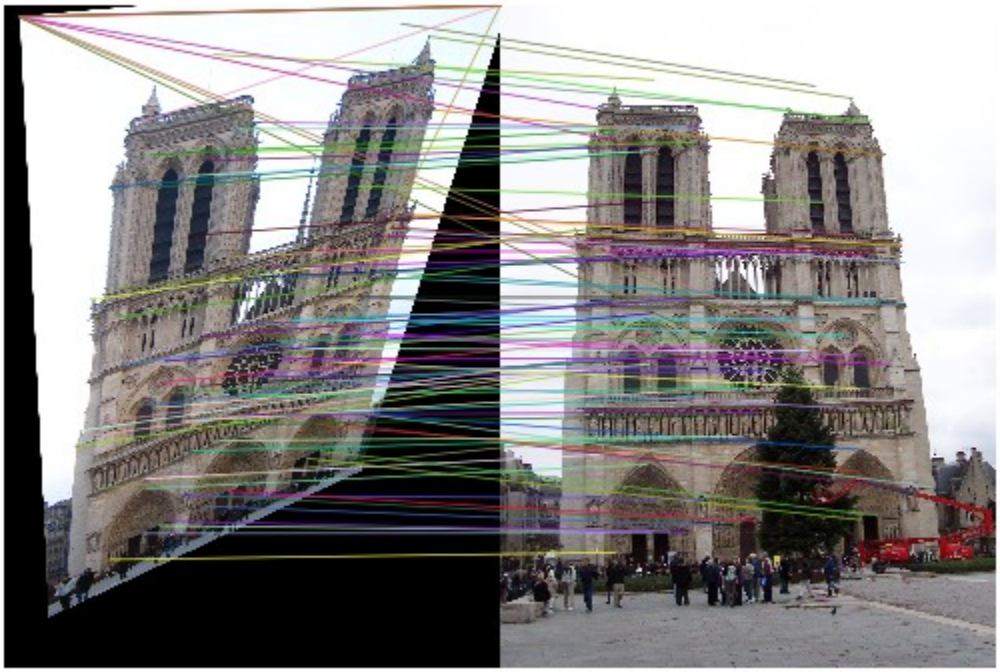




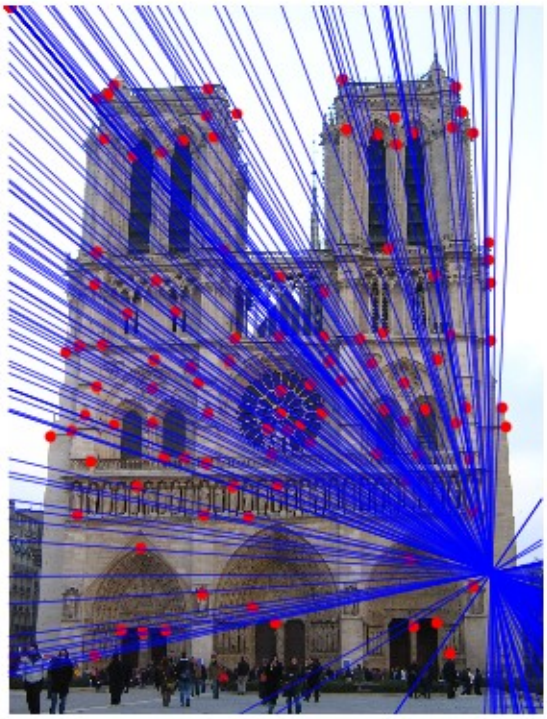


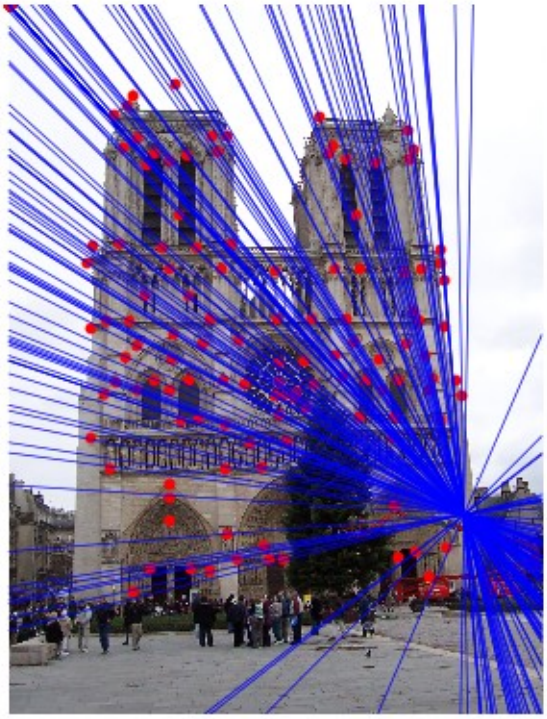
Another trial with iteration = 1000 and tolerance = 0.1 , normalized Fundamental matrix and noise:

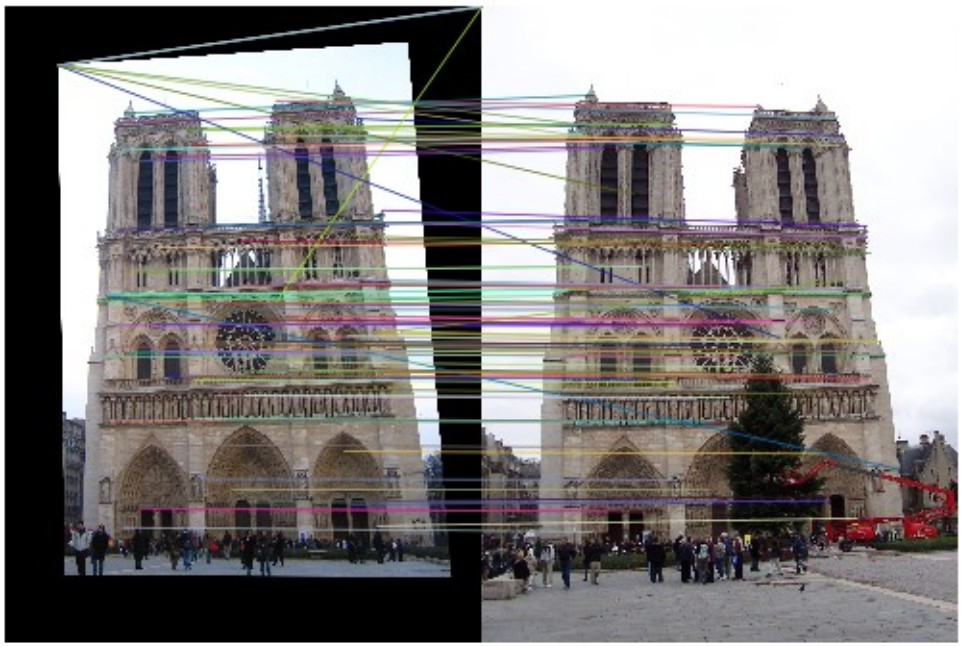




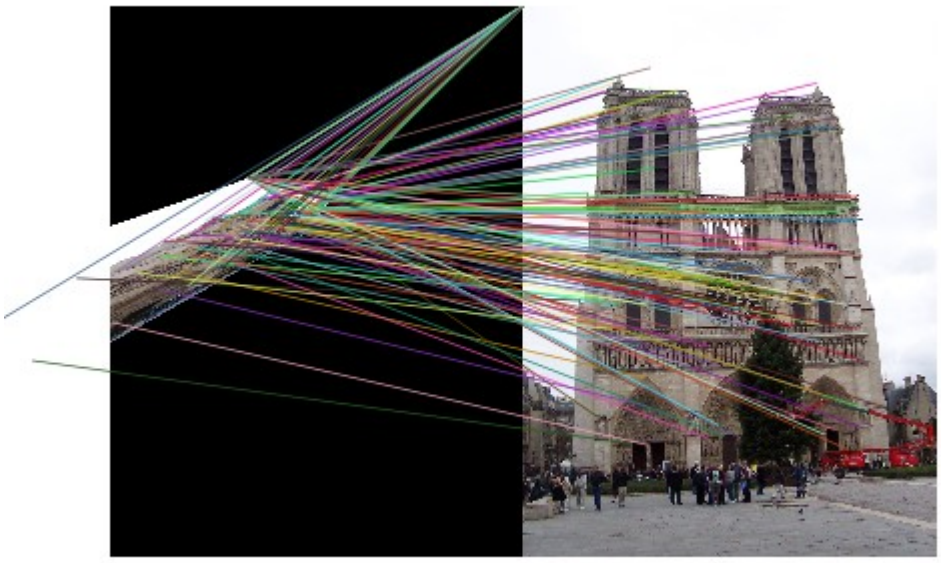
Another trial with iteration = 1500 and tolerance = 0.1 , normalized Fundamental matrix and noise:







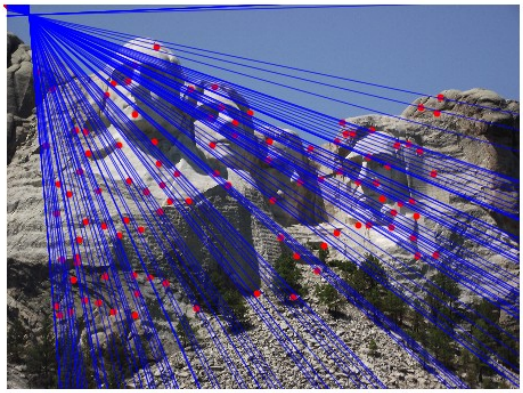
Another trial with iteration = 1000 and tolerance = 0.1 , normalized Fundamental matrix and no noise :

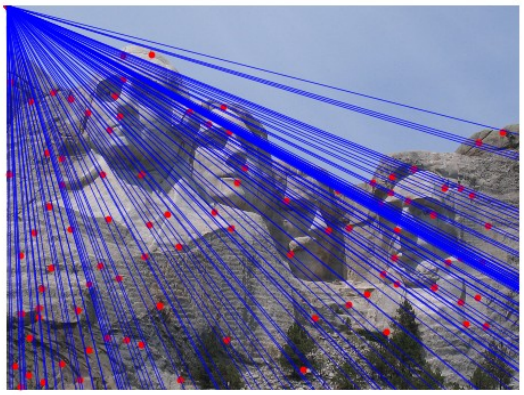


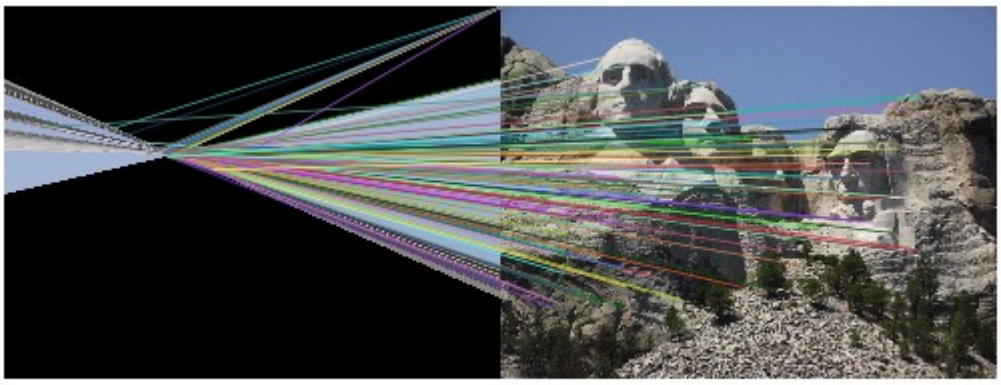
most of the images resulted were distorted and the fundamental matrices estimated were wrong this is deduced from trying to run the code on noise free image and noisy one and noticing that the result were much better in the cases in which the epipolar center lies on the right of the images while the so called ground truth points resulted in an epipolar center near the top left corner. This also may justify why increasing the iterations and decreasing the tolerance did not result in the hoped outcome or made it hard to produce right ones.

Rushmore Results :

tolerance =0.1 , iterations =1500, noise added , normalized fundamental matrix:

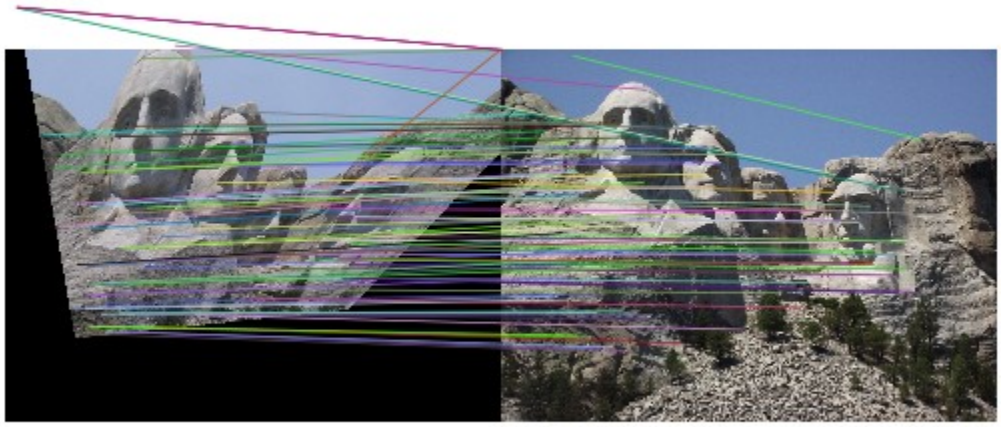




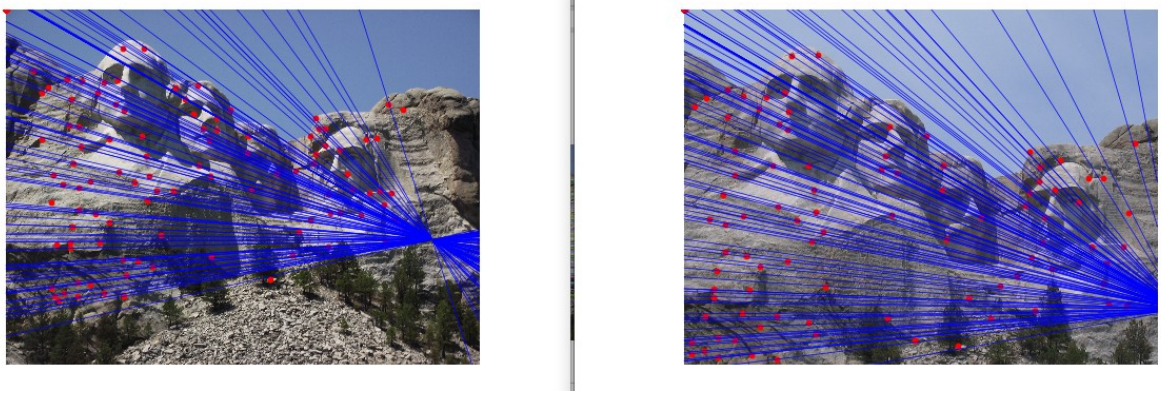


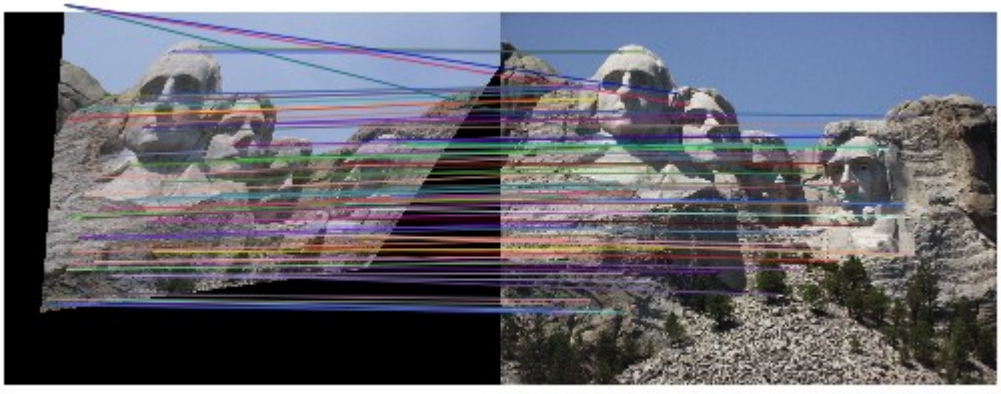
this may give indication that the ground truth points for Rushmore were faulty too as it is clear that epipolar center near the right edge of the image produced much better results than epipolar center near the top left corner.

tolerance =0.1 , iterations =1900, noise added , normalized fundamental matrix:



tolerance =0.07 , iterations =2000, noise added , normalized fundamental matrix:





In the end better images should be resulted by increasing the number of iterations, decreasing tolerance, using normalized fundamental matrix, and not only depending on the inliers count but also putting the inliers error in consideration. Unfortunately this trend was not seen in our images because of faulty ground truth images but with tuning tolerance, iterations and patience one could reach satisfactory results.