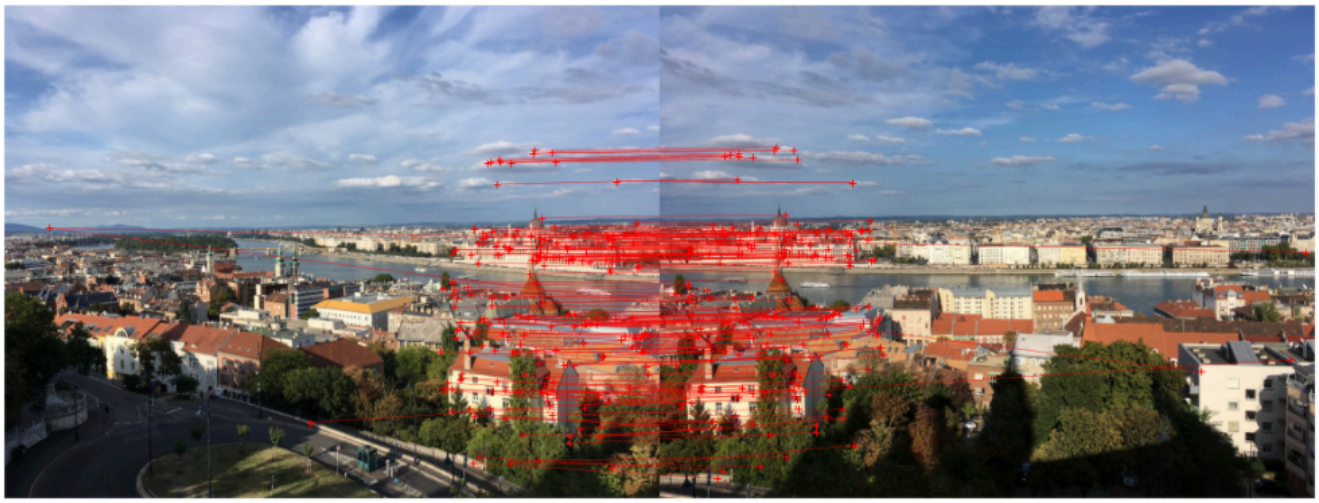
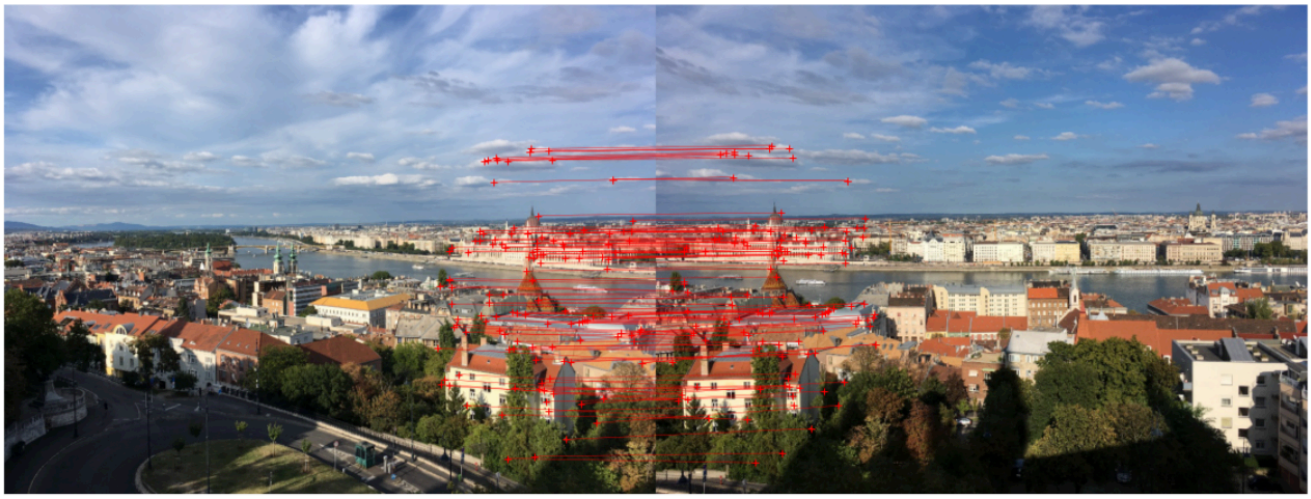
Problem 1

1.3 Putative Matches

I set the threshold = 20000:



I set the threshold = 15000:



I set the threshold = 10000:

1.4 Homography Estimation and RANSAC

Description of implementation details:

For the RANSAC

- In each iteration, I found 4 matchs and to calculate the homography of it.
- If the rank of H is smaller than 3, continue to choose another 4 matchs.
- Then, we will get all inliers fitted the current model and the errors of each homography we have
- We will find the best model by comparing the errro s. If total number of inliers is larger than current one and the error is smaller than current one, we update the model.

For Homography,

- Homography fitting calls for homogeneous least squares.
- Find the SVD of A by the singular vector corresponding to the smallest singular value.
- Use `U, S, V = numpy.linalg.svd(A)` performs the singular value decomposition (SVD).
- `V[-1,:]` gives the right singular vector corresponding to the smallest singular value.

Case1

When I choose the threshold == 20000, and thres_ransac == 0.3, num_iterations == 10000

```
Average residual: 0.1000233935871485
Inliers: 111
best_H is : [[ 3.27128525e+00  8.86084032e-02 -2.17782960e+03]
 [ 7.49312204e-01  2.87107375e+00 -6.34010068e+02]
 [ 2.23000288e-03  5.50249000e-05  1.00000000e+00]]
```

Case2

When I choose the threshold == 15000, and thres_ransac == 0.3, num_iterations == 10000

```
Average residual: 0.10506564302024403
Inliers: 74
best_H is : [[ 3.29472759e+00  9.13618935e-02 -2.19377955e+03]
 [ 7.66507347e-01  2.89101880e+00 -6.47604451e+02]
 [ 2.26127124e-03  4.84374643e-05  1.00000000e+00]]
```

Case3

When I choose the threshold == 10000, and thres_ransac == 0.3, num_iterations == 10000

```
Average residual: 0.11172287867764683
Inliers: 32
best_H is : [[ 3.27742212e+00  6.97250229e-02 -2.17332292e+03]
 [ 7.70567881e-01  2.84664863e+00 -6.39010402e+02]
 [ 2.27381189e-03 -1.44701581e-05  1.00000000e+00]]
```

1.5 Image Warping

Description of implementation details:

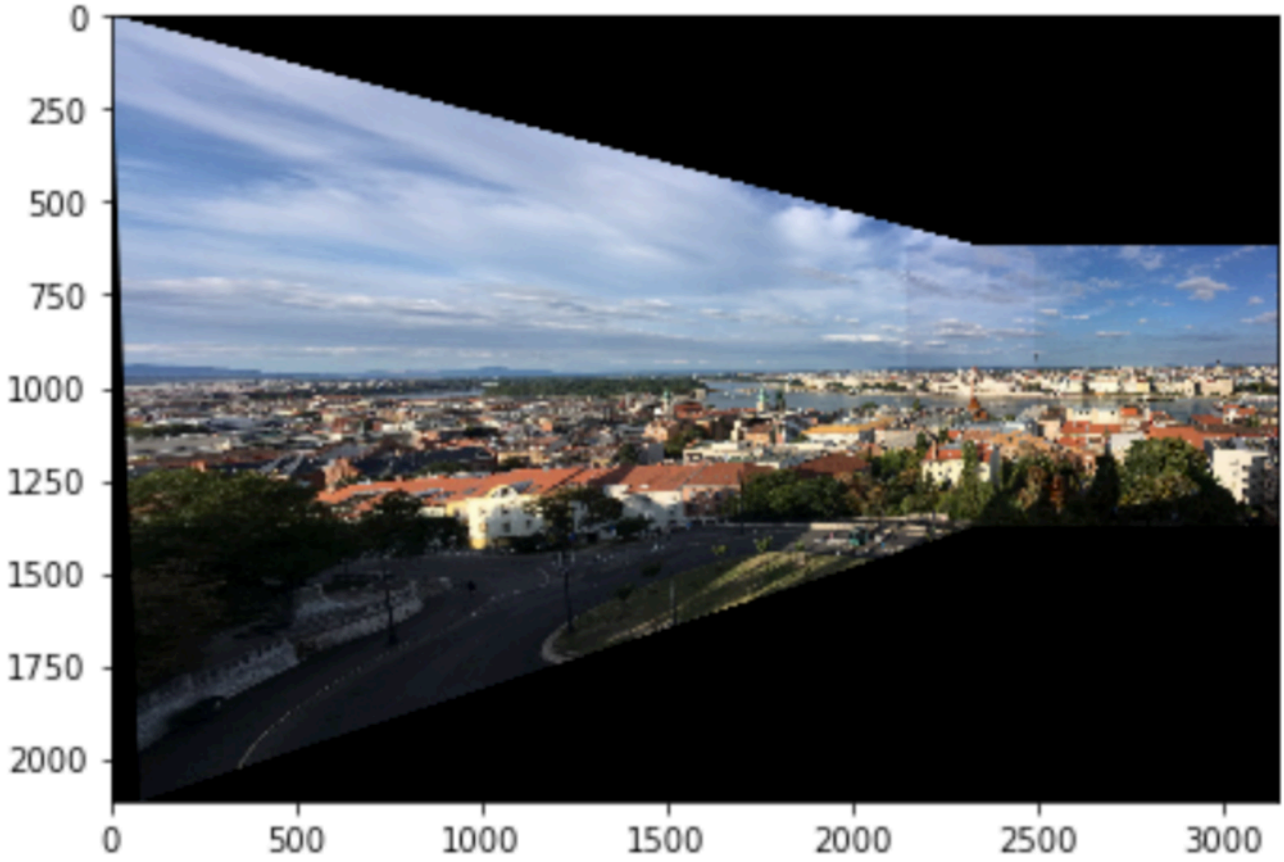
- Use the `skimage.transform.ProjectiveTransform` To find the transform from the best Homography results we got.
- Create a new image big enough to hold the panorama by comparing the size of the pictures and get the min_corner and the max_corner for us. Then, we can calculate the output shape.
- Compose the overlap part by calculating the averaging pixel of the two pictures.

Results1

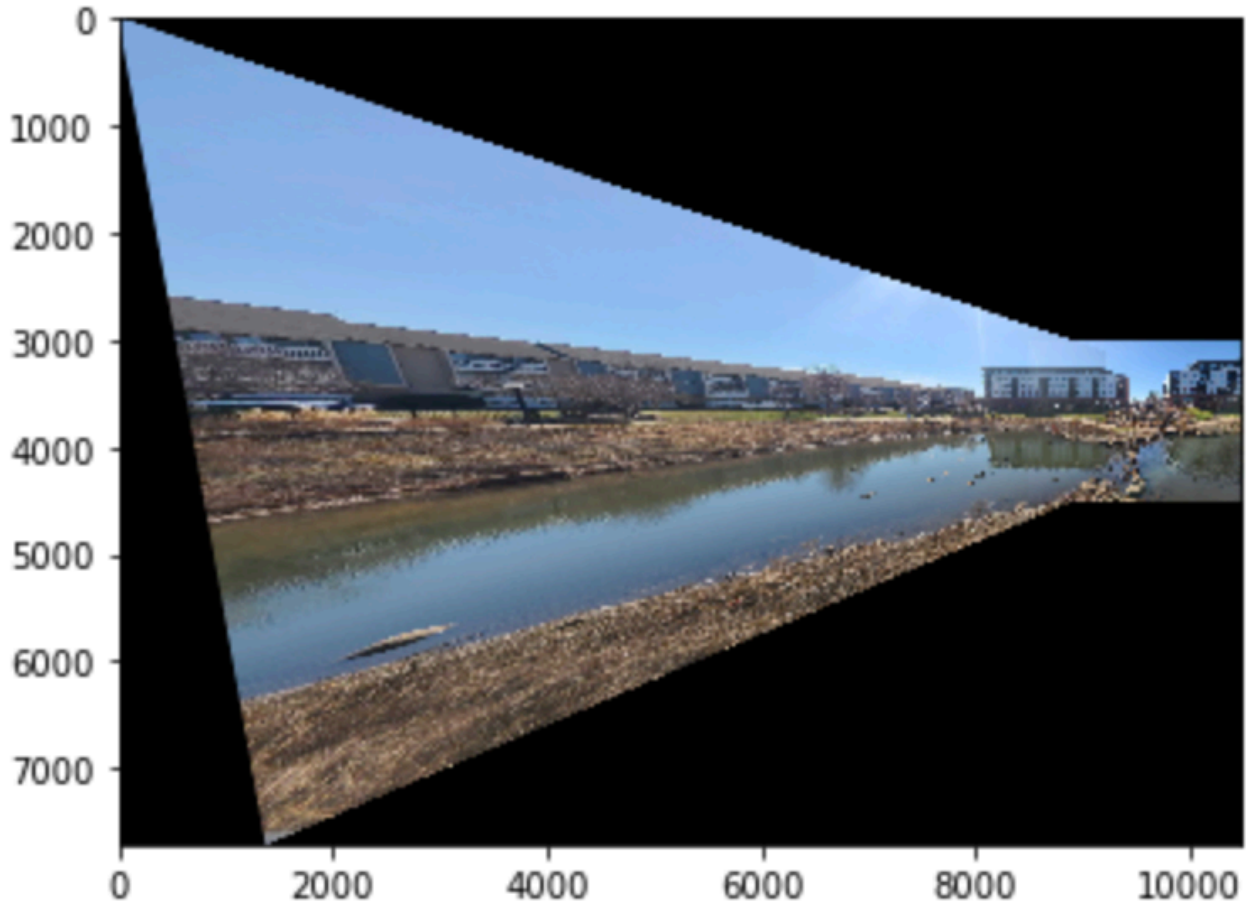
Is we project the right one, we got:

Results2

If we project the left one, we got:



1.6 Extra Credit



Problem 2

2.1 Fundamental Matrix Estimation

Description of implementation details:

For each algorithm and each image pair, report the estimated fundamental matrix, the residual (mean squared distance (in pixels) between points in both images and the corresponding epipolar line), and visualization of epipolar lines and corresponding points.

Non-normalized method

- Estimate the fundamental matrix by the all algorithm. I will randomly use eight points from both p1 and p2.
- Then I calculate as this:

$$\boldsymbol{x} = (u, v, 1)^T, \quad \boldsymbol{x}' = (u', v', 1)$$

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0 \quad \Rightarrow \quad \begin{bmatrix} u'u & u'v & u' & v'u & v'v & v' & u & v & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

Solve homogeneous linear system using eight or more matches

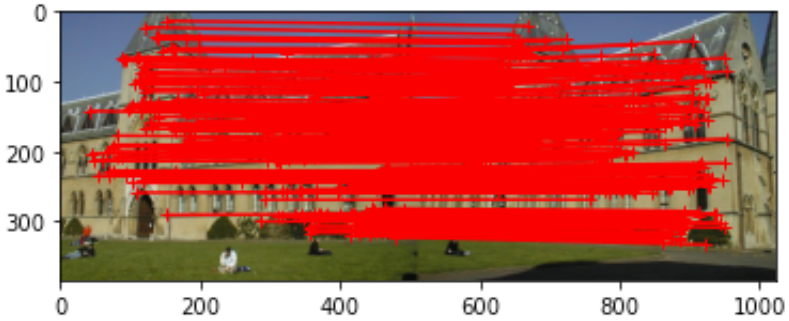
- Enforce the rank-2 constraint by taking SVD of F `U, S, Vt = np.linalg.svd(F)`
- Set the smallest singular value to zero, which is the

```
s_diag = np.diag(S)
s_diag[-1] = 0
F = np.dot(U, np.dot(s_diag, Vt))
```

Normalized method

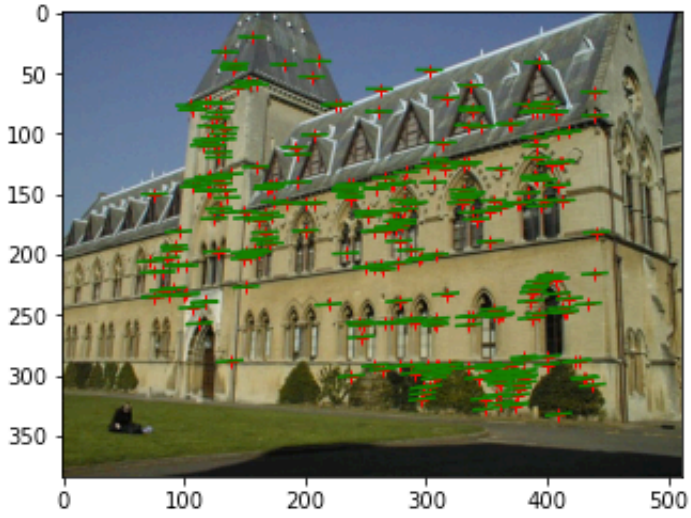
- For the normalized one, I will center the image data at the origin, and scale it so the mean squared distance between the origin and the data points is 2 pixels.
- Then I conducted the eight-point algorithm to compute F from the normalized points and Enforce the rank-2 constraint as the non-normalized one
- At the end, I transform fundamental matrix back to original units. : if T and T' are the normalizing transformations in the two images, than the fundamental matrix in original coordinates is T'.T dot F dot T.

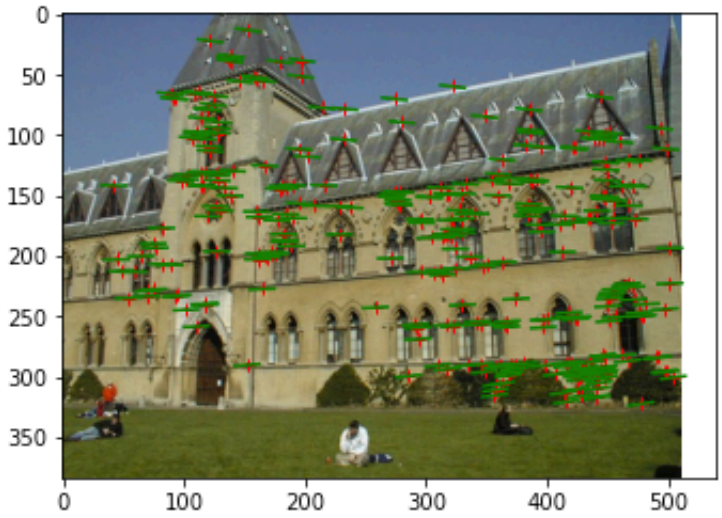
Library



Library-- Non-normalized method

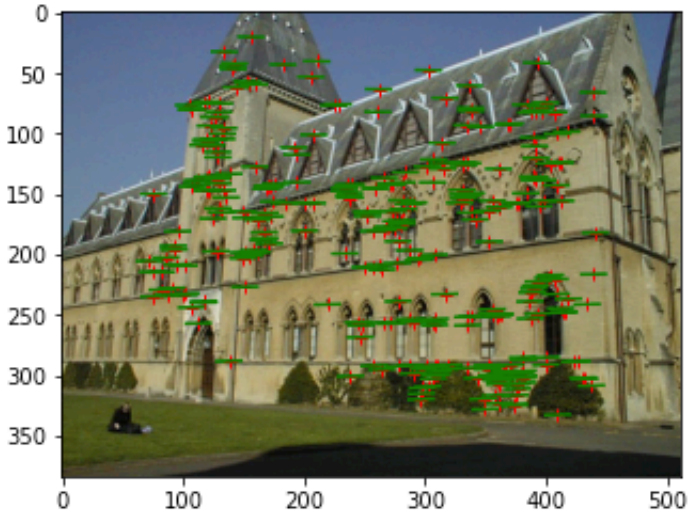
```
library: fundamental matrix (non-normalized method) =
 [[ 1.32546895e-06 -1.36852466e-05  6.83862987e-04]
 [ 2.88625175e-05 -2.66854091e-07 -4.09703775e-02]
 [-5.63235250e-03  3.73349826e-02  1.00000000e+00]]
library: residual in frame 2 (non-normalized method) =  0.17921336678352529
library: residual in frame 1 (non-normalized method) =  0.14912309937208557
library: residual combined   (non-normalized method) =  0.16416823307780543
```

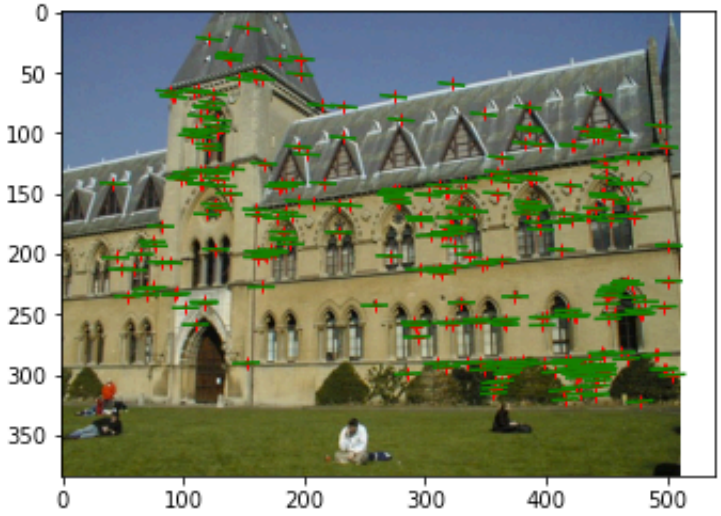




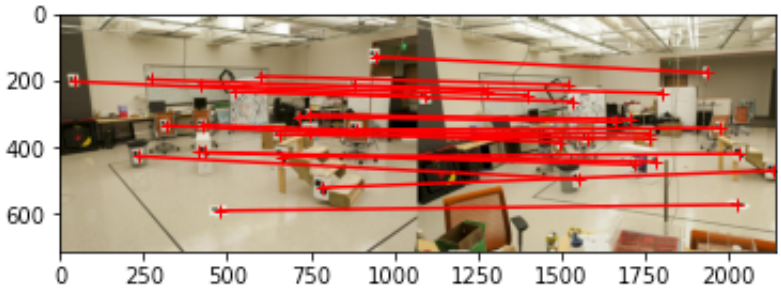
Library--Normalized method

```
library: fundamental matrix (normalized method) =
 [[ 4.01483520e-07 -6.08915845e-06  6.81663146e-04]
 [ 2.39077537e-05  8.34265282e-08 -4.13449118e-02]
 [-5.45063607e-03  3.72779285e-02  1.00000000e+00]]
library: residual in frame 2 (normalized method) =  0.06077736897675137
library: residual in frame 1 (normalized method) =  0.05497126849110569
library: residual combined   (normalized method) =  0.05787431873392853
```



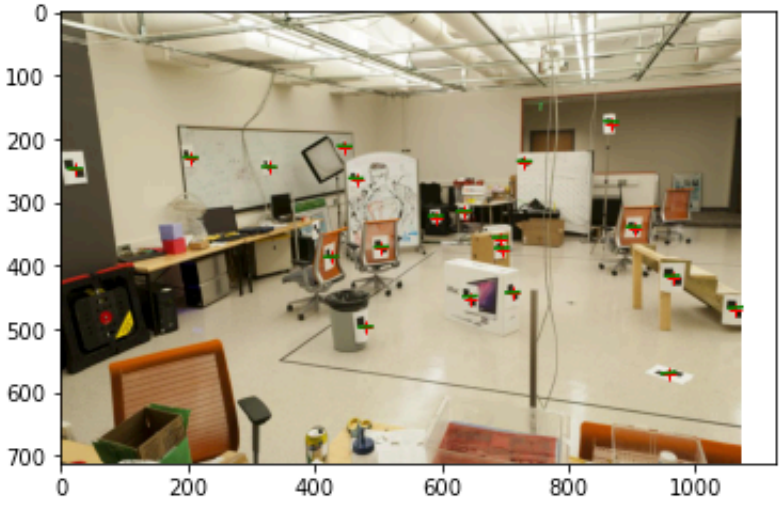


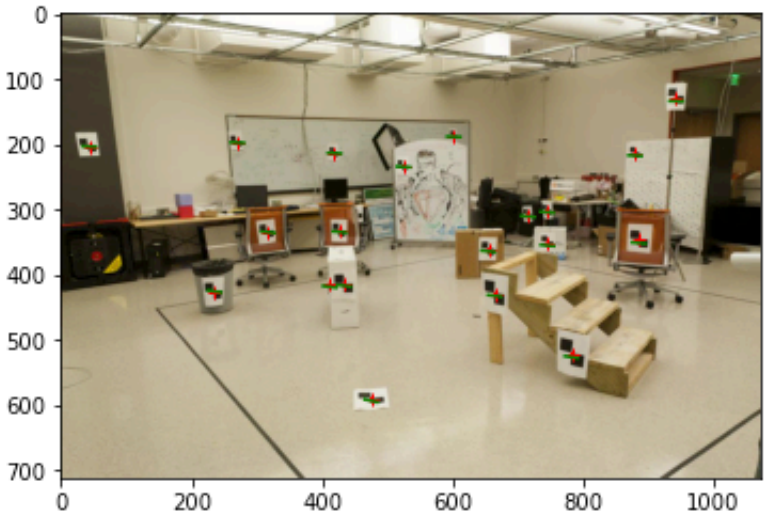
Lab



Lab-- Non-normalized method

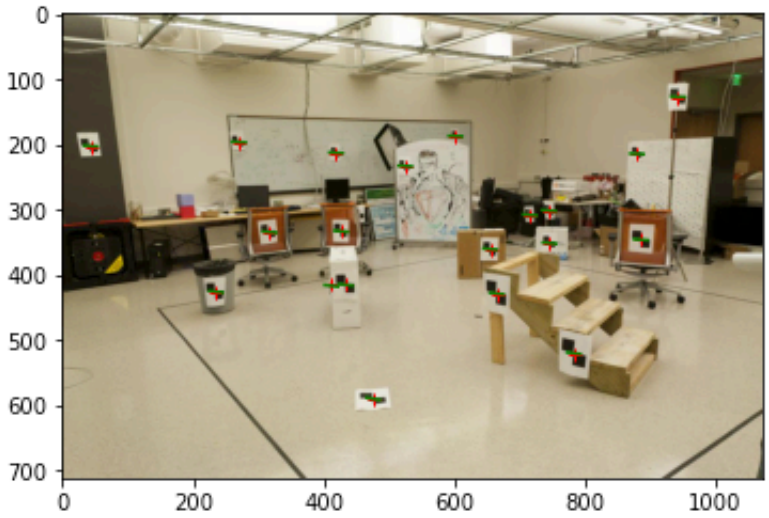
```
lab: fundamental matrix (non-normalized method) =
[[-5.36532415e-07  7.90760078e-06 -1.88694534e-03]
 [ 8.83981093e-06  1.21382365e-06  1.72419095e-02]
 [-9.07836099e-04 -2.64366809e-02  1.00000000e+00]]
lab: residual in frame 2 (non-normalized method) =  6.567091501484055
lab: residual in frame 1 (non-normalized method) =  9.760655423880264
lab: residual combined   (non-normalized method) =  8.16387346268216
```





Lab--Normalized method

```
lab: fundamental matrix (normalized method) =
[[-1.12053075e-06  1.54738807e-05 -3.86547034e-03]
 [ 1.06833421e-05 -2.64439076e-06  3.10978633e-02]
 [-2.40521010e-04 -4.27332749e-02  1.00000000e+00]]
lab: residual in frame 2 (normalized method) =  0.5275054622728044
lab: residual in frame 1 (normalized method) =  0.5670767777912991
lab: residual combined   (normalized method) =  0.5472911200320517
```



2.2 Camera Calibration

Description of implementation details:

- I load the pts of 2d and 3d from the file,
- We calcualte the projection matrix
- And valuate them the residual error from the provided evaluation function (evaluate_points)

Lab

```
lab 1 camera projection
[[-3.09963996e-03 -1.46204548e-04  4.48497465e-04  9.78930678e-01]
 [-3.07018252e-04 -6.37193664e-04  2.77356178e-03  2.04144405e-01]
 [-1.67933533e-06 -2.74767684e-06  6.83964827e-07  1.32882928e-03]]

lab 2 camera projection
[[ 6.93154686e-03 -4.01684470e-03 -1.32602928e-03 -8.26700554e-01]
 [ 1.54768732e-03  1.02452760e-03 -7.27440714e-03 -5.62523256e-01]
 [ 7.60946050e-06  3.70953989e-06 -1.90203244e-06 -3.38807712e-03]]
residuals between the observed 2D points and the projected 3D points:
residual in lab1: 13.545832895466091
residual in lab2: 15.54495345195818
```

Library

```
library1 camera projection
[[-4.5250208e+01  4.8215478e+02  4.0948922e+02  3.4440464e+03]
 [ 4.8858466e+02  2.7346374e+02 -1.3977268e+02  4.8030231e+03]
 [-1.9787463e-01  8.8042214e-01 -4.3093212e-01  2.8032556e+01]]
library2 camera projection
[[-5.9593834e+01  5.5643970e+02  2.3093716e+02  3.5683545e+03]
 [ 4.6419679e+02  2.2628430e+02 -1.9605278e+02  4.8734171e+03]
 [-1.9116708e-01  7.2057697e-01 -6.6650130e-01  2.8015392e+01]]
residuals between the observed 2D points and the projected 3D points:
residual in lib1: 15206.598973282029
residual in lib2: 19529.922450055652
```

2.3 Camera Centers

Description of implementation details:

- We can calculate the camera centers by the projection matrix we get above.
- Find out the SDV of the projection matrix got the VT.
- The center would be the last row of VT normalized.

Lab

```
lab1 camera center [305.83276769 304.20103826  30.13699243   1.        ]
lab2 camera center [305.83276769 304.20103826  30.13699243   1.        ]
```

Libary

```
library1 camera center [  7.28863053 -21.52118112  17.73503585   1.        ]
library2 camera center [  6.89405488 -15.39232716  23.41498687   1.        ]
```

2.4 Triangulation

Description of implementation details:

Triangulation: Linear approach

$$\lambda_1 \mathbf{x}_1 = \mathbf{P}_1 \mathbf{X} \qquad \mathbf{x}_1 \times \mathbf{P}_1 \mathbf{X} = \mathbf{0} \qquad [\mathbf{x}_{1\times}]\mathbf{P}_1 \mathbf{X} = \mathbf{0}$$
$$\lambda_2 \mathbf{x}_2 = \mathbf{P}_2 \mathbf{X} \qquad \mathbf{x}_2 \times \mathbf{P}_2 \mathbf{X} = \mathbf{0} \qquad [\mathbf{x}_{2\times}]\mathbf{P}_2 \mathbf{X} = \mathbf{0}$$

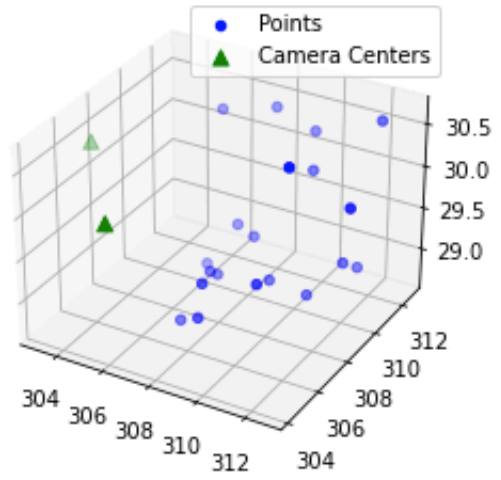
Cross product as matrix multiplication:

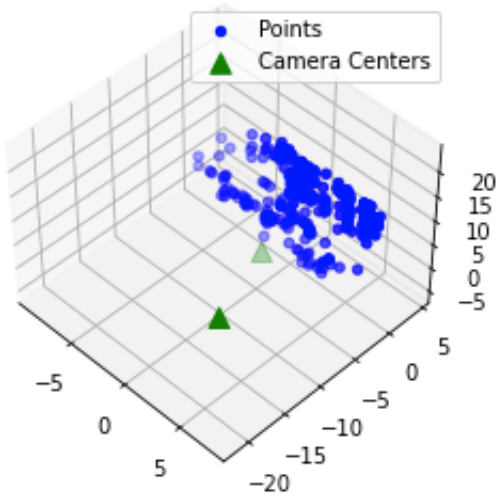
$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [\mathbf{a}_\times]\mathbf{b}$$

Results

```
Mean 3D reconstuction error for the lab data:  0.00025
2D reprojection error for the lab 1 data:  10.89944601416041
2D reprojection error for the lab 2 data:  1.5485148072796253
2D reprojection error for the library 1 data:  24.662071196869096
2D reprojection error for the library 2 data:  28.649537735259983
```

```
<matplotlib.legend.Legend at 0x7fd7502d6b50>
```

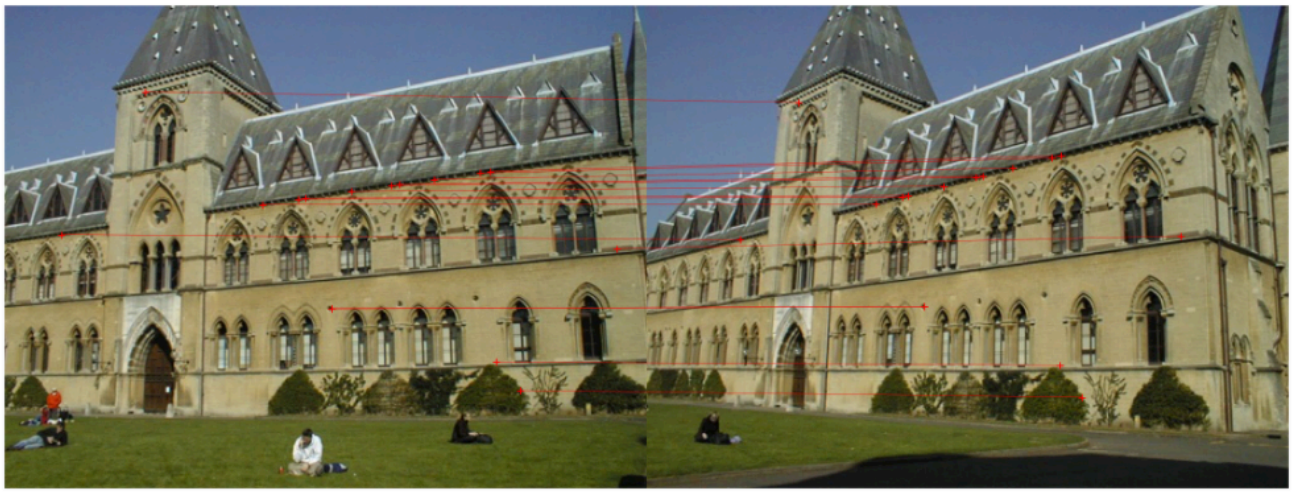




2.5 Extra Credits

Libary

```
Average residual: 0.06766314513759808
Inliers: 16
```



Compare the quality of the result with the one get from ground-truth matches:
library: residual in frame 2 (normalized method) = 0.06077736897675137
library: residual in frame 1 (normalized method) = 0.05497126849110569
library: residual combined   (normalized method) = 0.05787431873392853

The residual is larger in ground-truth matches

Lab

```
Average residual: 0.07207980171915132
Inliers: 27
```

Compare the quality of the result with the one get from ground-truth matches:

lab: residual in frame 2 (non-normalized method) = 6.567091501484055
lab: residual in frame 1 (non-normalized method) = 9.760655423880264
lab: residual combined   (non-normalized method) = 8.16387346268216

The residual is larger in ground-truth matches

Problem3

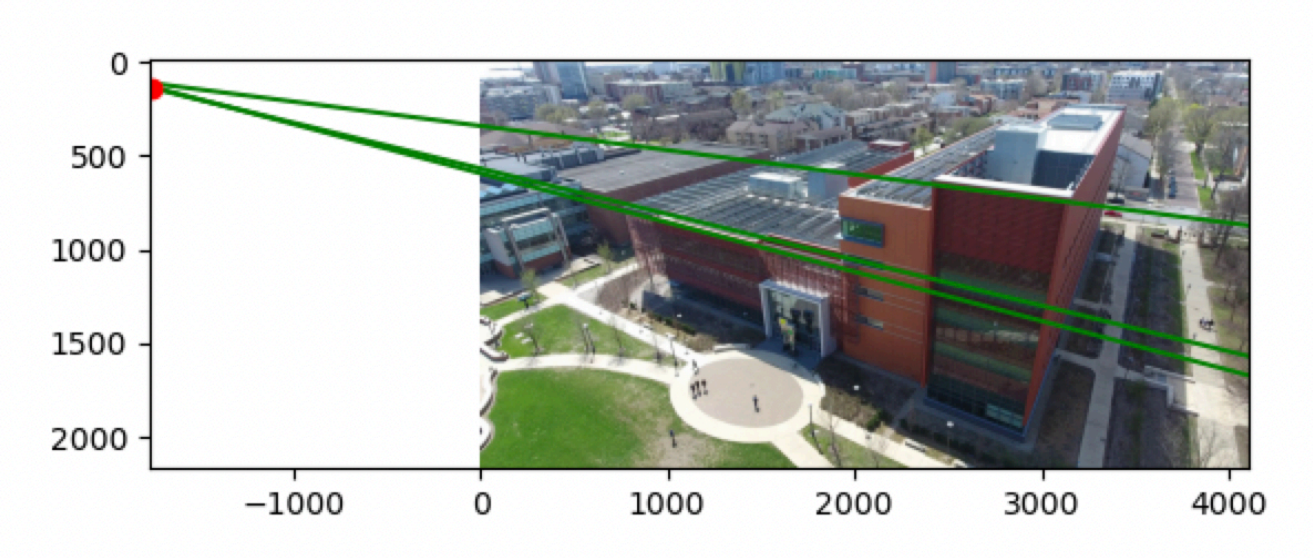
3.1 Vanishing Points

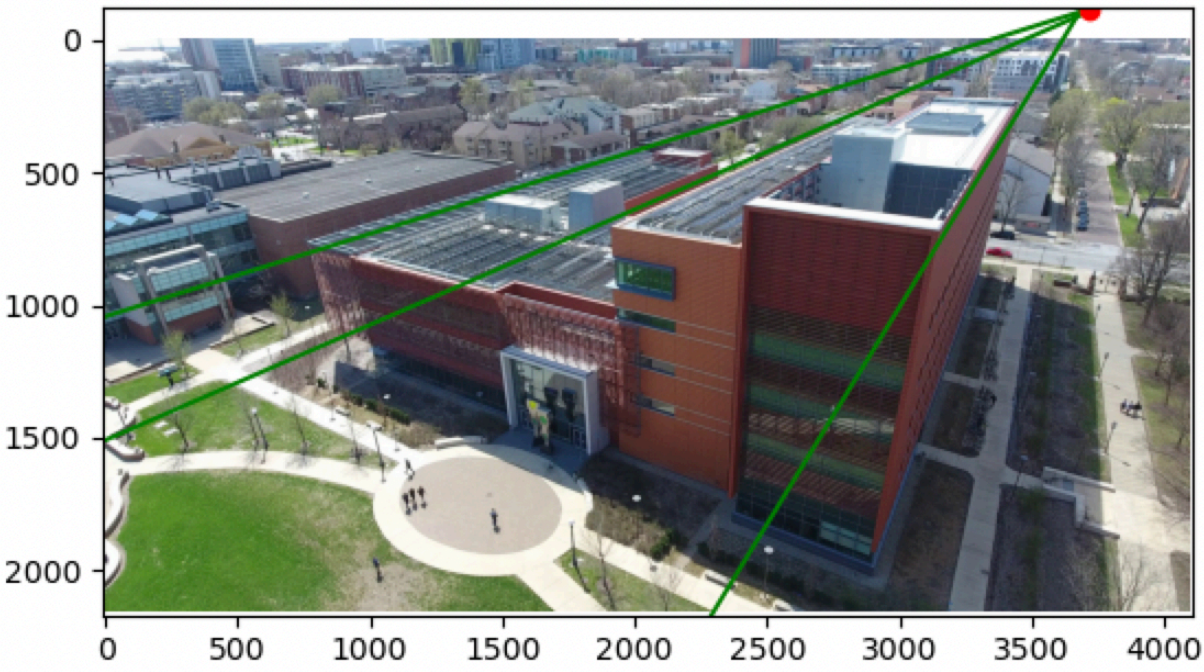
Description of implementation details:

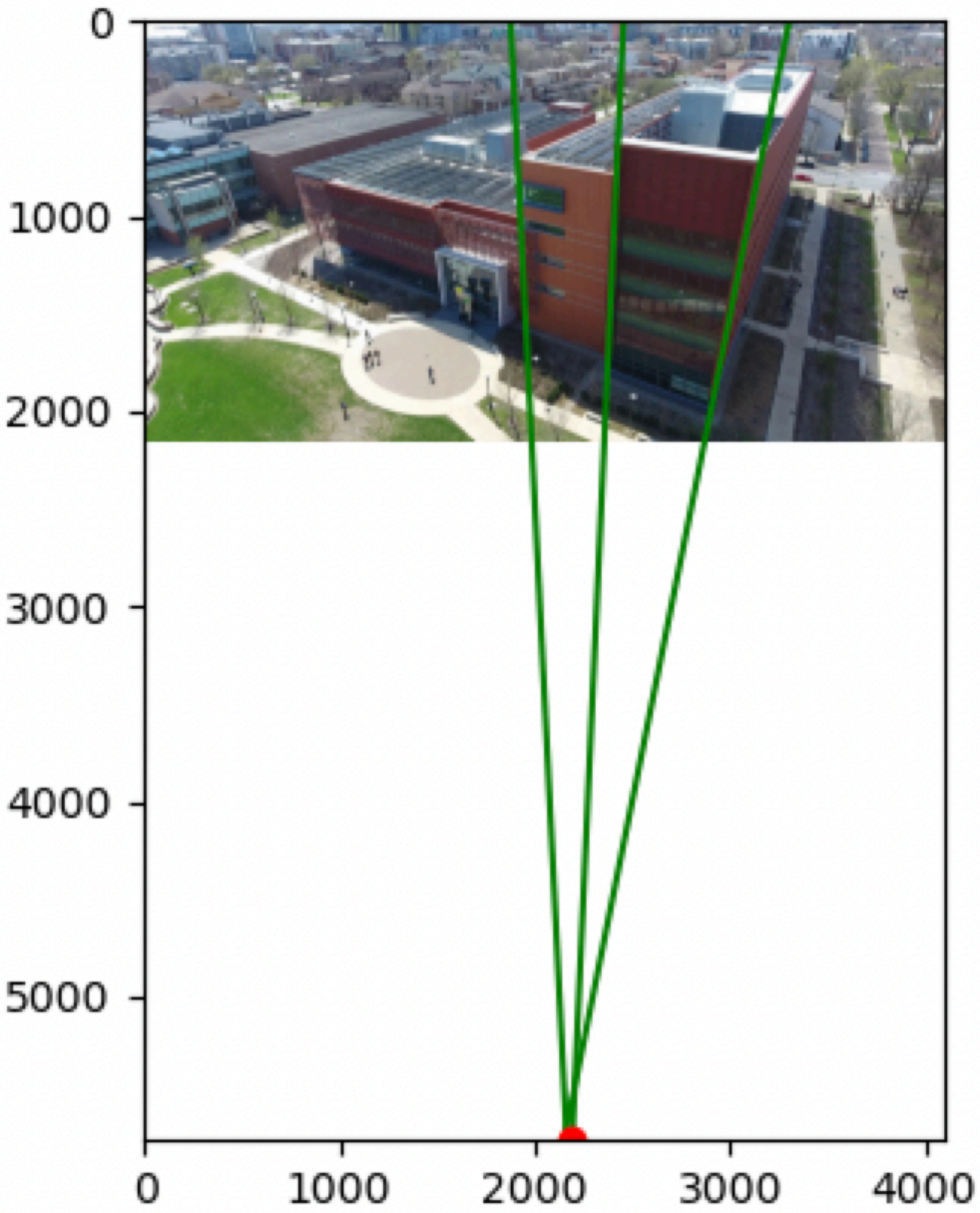
- The vanishing lines can get ftom cross of all the lines
- We normalize all teh cross points
- Stack the answers togethere

Results:

```
vanishing point: [-1.93040643e+03  9.16102077e+01  1.00000000e+00]
vanishing point: [ 3.67634539e+03 -9.95653021e+01  1.00000000e+00]
vanishing point: [2.17464930e+03 5.66480795e+03 1.00000000e+00]
```




0
500
1000
1500
2000
0
500
1000
1500
2000
2500
3000
3500
4000



3.2 Horizon

Description of implementation details:

- The horizional line is the cross of two valishing points x and y
- The horizional line is the cross of two valishing points x and y

Results:

Horizontal lines in the form ax+by+c=0

03407757 x +99941919 y -25.77344097 = 0.

```
Horizon Line: [  0.03407757   0.99941919 -25.77344097]
```

3.3 Camera Calibration

Description of implementation details:

Since the fact that the vanishing directions are orthogonal , solve for the focal length and optical center (principal point) of the camera. Show all your work, and report the values you find.

- We can calibrate from vanishing points. Let us align the world coordinate system with three orthogonal vanishing directions in the scene.
- From the vpts, we can get the three vanishing points in three directions.
- Then, we calculate the K and the [f,px,py]

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\lambda_i \mathbf{v}_i = \mathbf{K}\mathbf{R}\mathbf{e}_i$$

$$\mathbf{e}_i = \lambda_i \mathbf{R}^T \mathbf{K}^{-1} \mathbf{v}_i$$

- Orthogonality constraint: $\mathbf{e}_i^T \mathbf{e}_j = 0$

$$\mathbf{v}_i^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{v}_j = 0$$

- Then, we can solve the equation we the SymPy.solve function.
- So, we can get the [f,px,py] and the according K.

Results:

```
f is: -2299.61367284199
px is: 2019.70603363198
py is: 1120.66717441262
K is: [[-2.2996138e+03  0.0000000e+00  2.0197061e+03]
 [ 0.0000000e+00 -2.2996138e+03  1.1206671e+03]
 [ 0.0000000e+00  0.0000000e+00  1.0000000e+00]]
```

3.4 Rotation Matrix

Description of implementation details:

After we got the vanishing points, f, px, py. We can get that the rotation matrix is the dot product the the inverse of K and the vanishing points in each direction.

We also need to normolize our answer at the end.

Results:

```
Rotation matirx is : [[-0.53687352 -0.03040942  0.84311451]
 [ 0.39544545 -0.8918408   0.21964255]
 [ 0.74524474  0.4513261   0.49083101]]
```