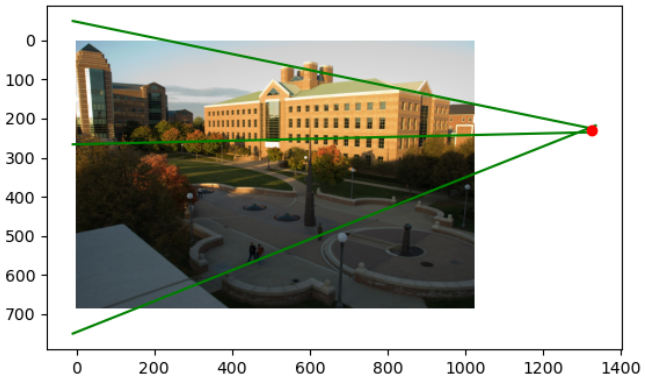
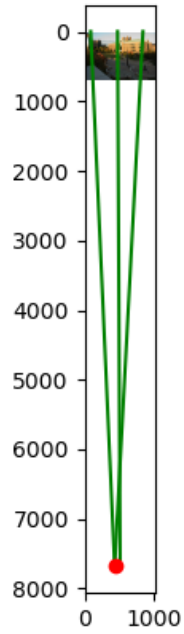
CS543 Assignment 4

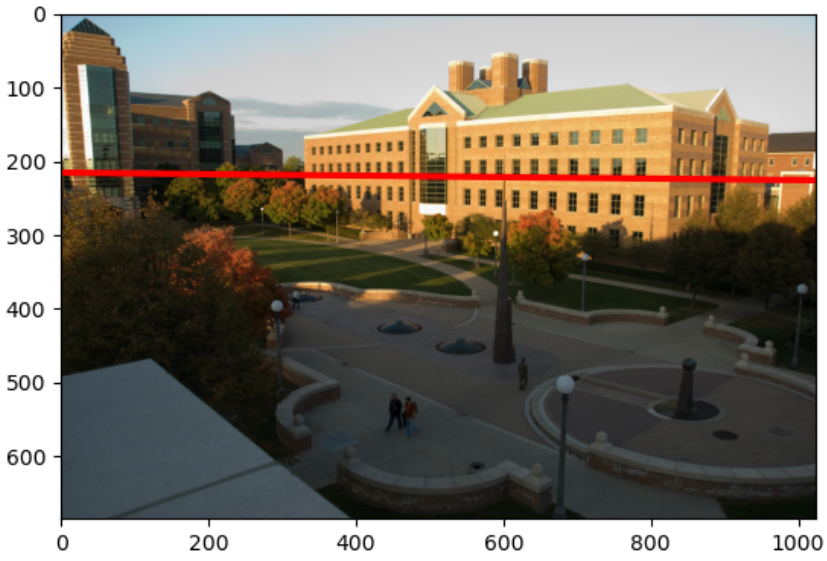
Your Name: hongqing liu
Your NetId: hl85

Part 1 Single-View Geometry:

Plot the VPs and the lines used to estimate them on the image plane using the provided code.


0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200
1400


0
1000
2000
3000
4000
5000
6000
7000
8000
0
1000


0
100
200
300
400
500
600
0
200
400
600
800
1000

Specify the VP pixel coordinates.

VP1:[-2.45010665e+02, 2.12768188e+02,1]

VP2: [1.32478877e+03, 2.28540401e+02, 1]

VP3: [4.50809524e+02, 7.67276174e+03, 1]

Plot the ground horizon line and specify its parameters in the form a * x + b * y + c = 0. Normalize the parameters so that: a^2 + b^2 = 1.

horizon_line: [-1.00467722e-02 9.99949530e-01 -2.15219016e+02]

Line:

$$-1.00467722e-02 * x + 9.99949530e-01 * y + (-2.15219016e+02) = 0$$

Using 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.

My code:

```
]: def get_camera_parameters(vpts):
    '''
    Computes the camera parameters. Hint: The SymPy package is suitable for this.
    '''
    vp1 = vpts[:, 0].reshape(3,1)
    vp2 = vpts[:, 1].reshape(3,1)
    vp3 = vpts[:, 2].reshape(3,1)

    f, px, py = symbols('f, px. py')
    K_i_T = Matrix([[1/f, 0, 0], [0, 1/f, 0], [-px/f, -py/f, 1]])
    K_i = Matrix([[1/f, 0, -px/f], [0, 1/f, -py/f], [0, 0, 1]])

    eq1 = Eq((vp1.T @ K_i_T @ K_i @ vp2)[0], 0)
    eq2 = Eq((vp1.T @ K_i_T @ K_i @ vp3)[0], 0)
    eq3 = Eq((vp2.T @ K_i_T @ K_i @ vp3)[0], 0)

    f, u, v = solve((eq1,eq2,eq3), (f,px,py))[0]

    return f, u, v
```

focal length = -780.447080083736
optical center = (524.854039253718,303.153165312346)

Compute the rotation matrix for the camera.

Rotation matrix = [[-0.71418181 0.00999091 0.69988893]

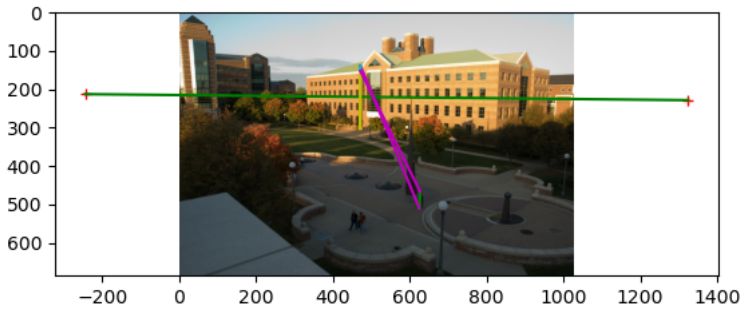
 [0.06661428 -0.99438961 0.08216956]

 [0.69678323 0.10530661 0.70950944]]

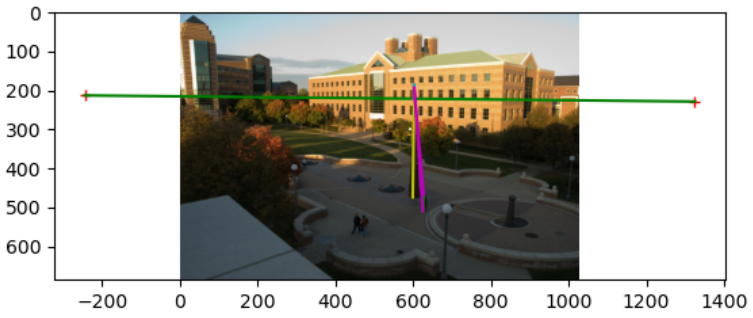
[9]:
```
def get_rotation_matrix(vpts, f, u, v):
    """
    Computes the rotation matrix using the camera parameters.
    """
    Z = vpts[:, 0].reshape(3,1)
    X = vpts[:, 1].reshape(3,1)
    Y = vpts[:, 2].reshape(3,1)
    K = np.array([[f, 0, u], [0, f, v], [0, 0, 1]], dtype=float)
    K_inv = np.linalg.inv(K)

    r1 = K_inv.dot(X)
    r2 = K_inv.dot(Y)
    r3 = K_inv.dot(Z)

    r1 = r1 / np.linalg.norm(r1)
    r2 = r2 / np.linalg.norm(r2)
    r3 = r3 / np.linalg.norm(r3)

    R = np.concatenate((r1, r2, r3), axis=1)
    return R
```

Estimate the heights of (a) the CSL building, (b) the spike statue, and (c) the lamp posts assuming that the person nearest to the spike is 5ft 6in tall. In the report, show all the lines and measurements used to perform the calculation.

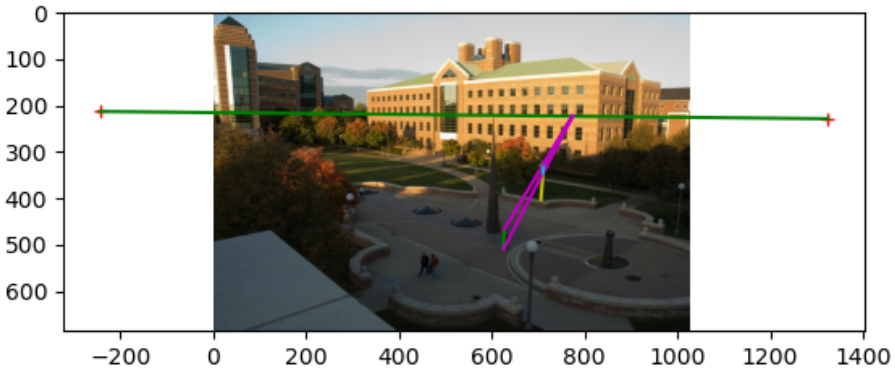
(a) the CSL building: 1985.2394605032034cm



(b) the spike statue: 6910.046590565572cm



(c) the lamp posts: 659.9263279669311cm



My codes:

[10]:
```
def estimate_height(im,person, obj, vpts, horizon_line, ref_height):
    """
    Estimates height for a specific object using the recorded coordinates. You might need to plot additional i
    your report.
    """
    vp1 = vpts[:, 0].reshape(3,1)
    vp2 = vpts[:, 1].reshape(3,1)
    vp3 = vpts[:, 2].reshape(3,1)
    #可视化地平线
    fig, ax = plt.subplots()
    ax.set_aspect('equal')
    ax.imshow(im)
    ax.plot(vp1[0],vp1[1],'+r')
    ax.plot(vp2[0],vp2[1],'+r')
    ax.plot([vp1[0], vp2[0]], [vp1[1], vp2[1]], 'g')
    #可视化人和物
    ax.plot([person[0][0], person[0][1]], [person[1][0], person[1][1]], 'g')
    ax.plot([obj[0][0], obj[0][1]], [obj[1][0], obj[1][1]], 'y')
    b0 = np.array([person[0][0], person[1][0], 1])
    t0 = np.array([person[0][1], person[1][1], 1])
    b = np.array([obj[0][0], obj[1][0], 1])#bt可能反了
    r = np.array([obj[0][1], obj[1][1], 1])
    #计算目标高度
    img_height_tar = abs(obj[1][0] - obj[1][1])
    #求bottom_line
    bottom_line = np.real(np.cross(b0.T, b.T))
    v = np.real(np.cross(bottom_line, horizon_line))#这里可能有问题horizonline
    v = v / v[2]
    #求top_line
    top_line = np.real(np.cross(v.T, t0.T))
    vertical_line = np.real(np.cross(r.T, b.T));
    t = np.real(np.cross(top_line.T, vertical_line.T))#这里可能有问题horizonline
    t = t / t[2]
    ax.plot([v[0], b0[0]], [v[1], b0[1]], 'm')
    ax.plot([v[0], t0[0]], [v[1], t0[1]], 'm')
    ax.plot([v[0], t[0]], [v[1], t[1]], 'm')
    ax.plot([b[0], t[0]], [b[1], t[1]], 'c')
    ax.plot([b[0], v[0]], [b[1], v[1]], 'm')
    #计算高度差
    left_up = np.linalg.norm(t - b) * np.linalg.norm(vp3.T - r)
    left_down = np.linalg.norm(r - b) * np.linalg.norm(vp3.T - t)
    H = ref_height
    height_tar = (H * left_down) / left_up
    plt.show()
    return height_tar
```

How do the answers change if you assume the person is 6ft tall?

(a) the CSL building: 2165.715775094404cm

(b) the spike statue: 7538.23264425335cm

(c) the lamp posts: 719.9196305093795cm

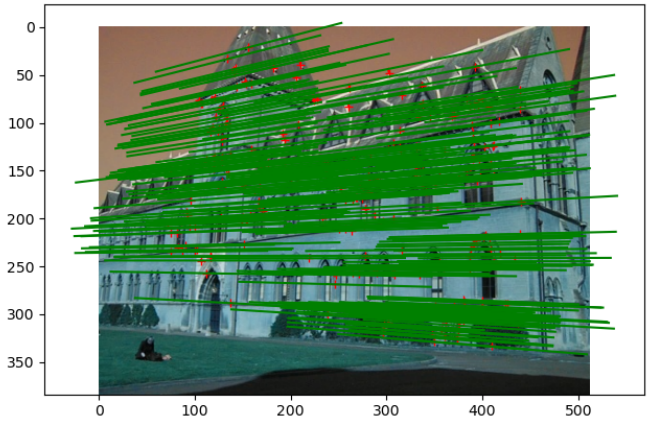
The assume height change linearly, as shown below:

the spike statue(6)/the spike statue(5.6) - the CSL building(6)/the CSL building(5.6) = 0

Part 2 Fundamental Matrix Estimation, Camera Calibration, Triangulation:

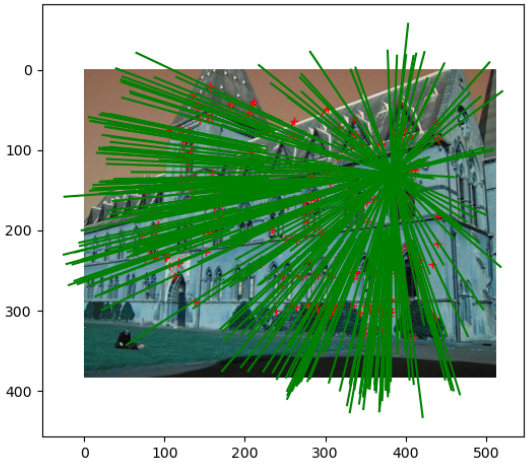
For the lab and library image pairs, display your result (points and epipolar lines) and report your residual for both unnormalized and normalized fundamental matrix estimation.
When set iteration of RANSAC = 3000,
Unormalized: residual = [6.93725205e-05]

Result:



Normalized: residual = [0.00855667]

Result:

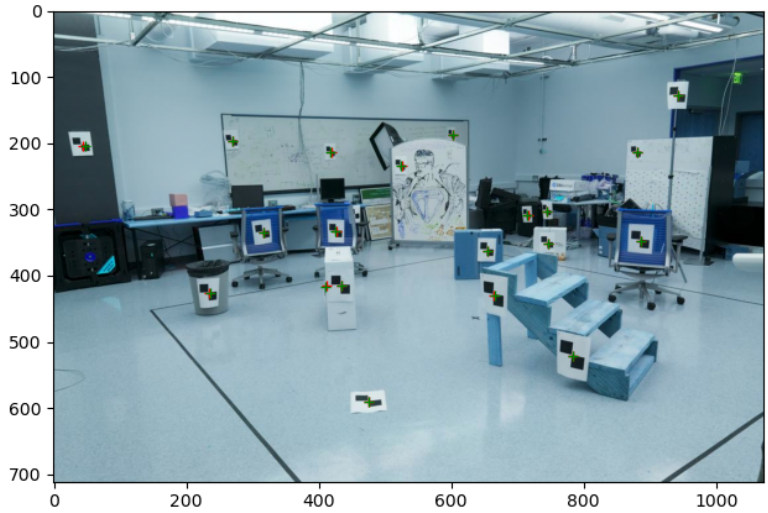


For the lab image pair, show your estimated 3x4 camera projection matrices. Report the residual between the projected and observed 2D points.
For lab1
Residual = 17.55419775809577

Projection_matrix:

[[-2.33997834e+00 -9.61260519e-02 2.79807855e-01 7.36286936e+02]
 [-2.23799709e-01 -4.79474906e-01 2.09041057e+00 1.51230956e+02]
 [-1.24876982e-03 -2.07623720e-03 4.41071003e-04 1.00000000e+00]]

plot:



Camera center: [[305.76673193 304.13974173 30.15050006]]

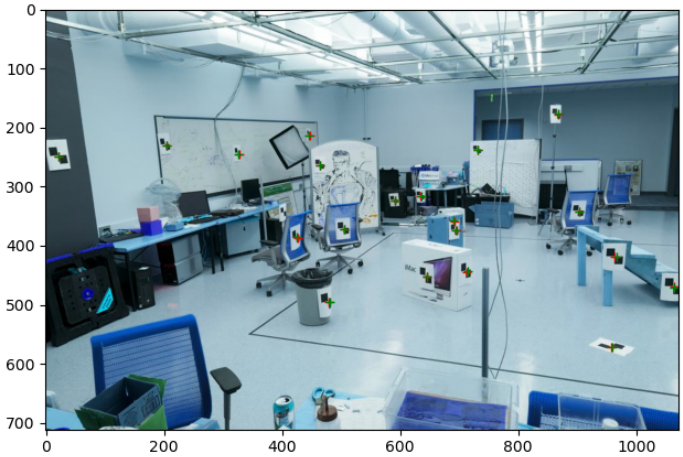
For lab2

Residual = 18.211882498207014

Projection_matrix:

[[-2.03842556e+00 1.17808019e+00 4.22851725e-01 2.43118127e+02]
 [-4.54871950e-01 -3.05003277e-01 2.15688057e+00 1.65966280e+02]
 [-2.24148032e-03 -1.10433637e-03 6.14930281e-04 1.00000000e+00]]

plot:

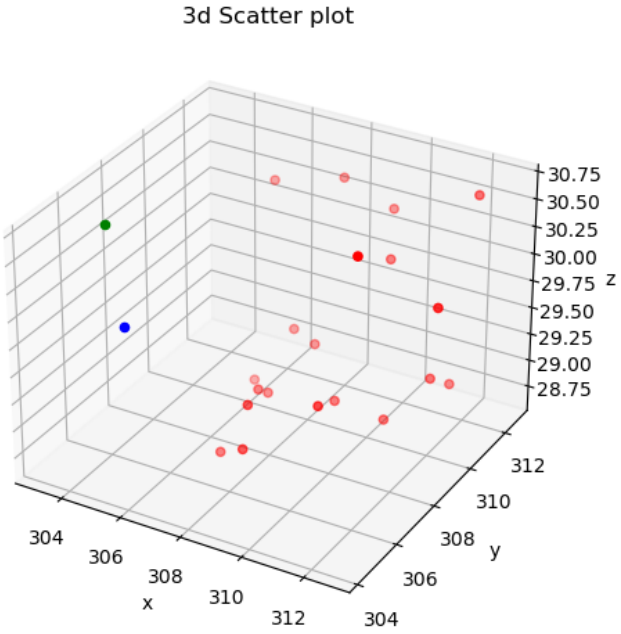


Camera center: [[303.12434646 307.20700648 30.4215388]]

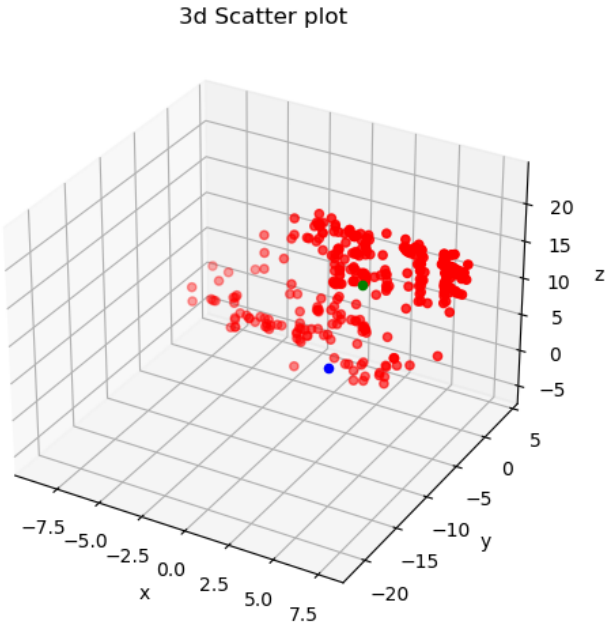
Library 1 camera center: [[7.28863053 -21.52118112 17.73503585]]
Library 2 camera center: [[6.89405488 -15.39232716 23.41498687]]

For the lab and library image pairs, visualize 3D camera centers and triangulated 3D points.

For the lab: red points are triangulated 3D points, the blue one is the camera center for lab1 and the green point is the camera center for lab2.

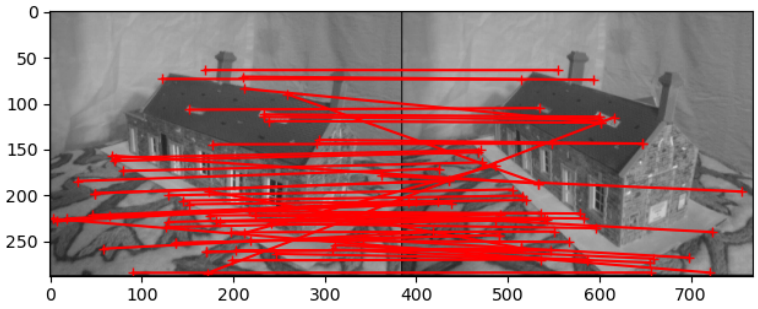


For the library: red points are triangulated 3D points, the blue one is the camera center for library1 and the green point is the camera center for library2.

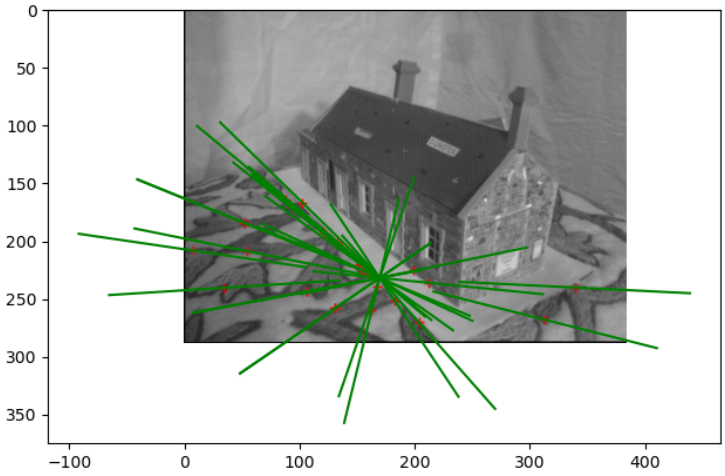


For the house and gaudi image pairs, display your result and report your number of inliers and average inlier residual for normalized estimation without ground truth matches.

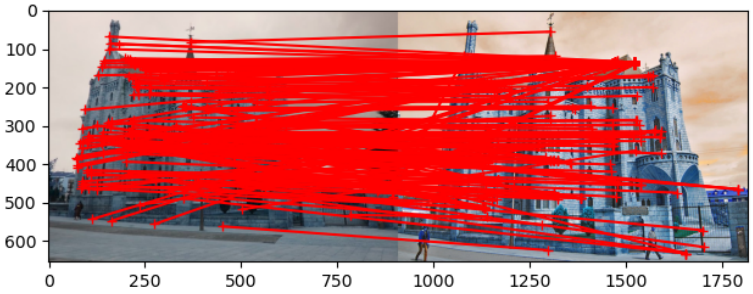
For house image pairs, when I set SIFT descriptor distance threshold = 20000, I got 54 pairs of match points, When I set RANSAC iteration = 10000 and cost threshold = 0.001, I got 25 pairs of inliers and the average residual = 0.00028113981619660224
The SIFT pairs before RANSAC is shown below:



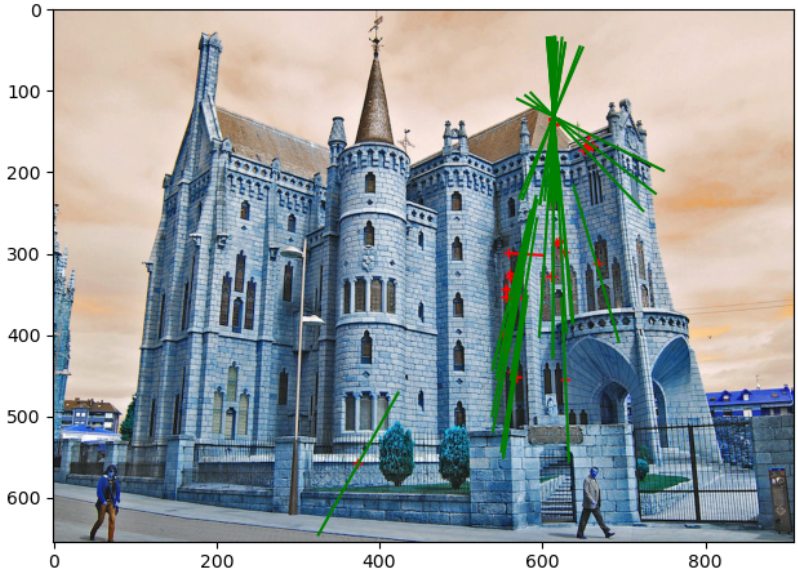
The result is shown below:



For guadi image pairs, when I set SIFT descriptor distance threshold = 20000, I got 201 pairs of match points, When I set RANSAC iteration = 10000 and cost threshold = 0.001, I got 54 pairs of inliers and the average residual = 0.0003097428014970789
The SIFT pairs before RANSAC is shown below:



The process result is shown below:



Extra Credit:

Don’t forget to include references, an explanation, and outputs to receive credit. Refer to the assignment for suggested outputs.

Part 1

Part 2