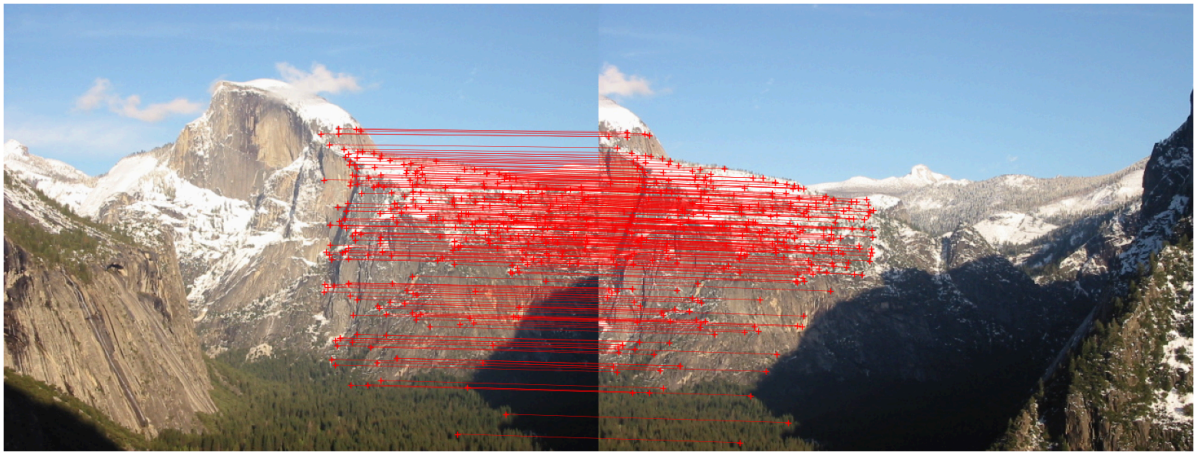
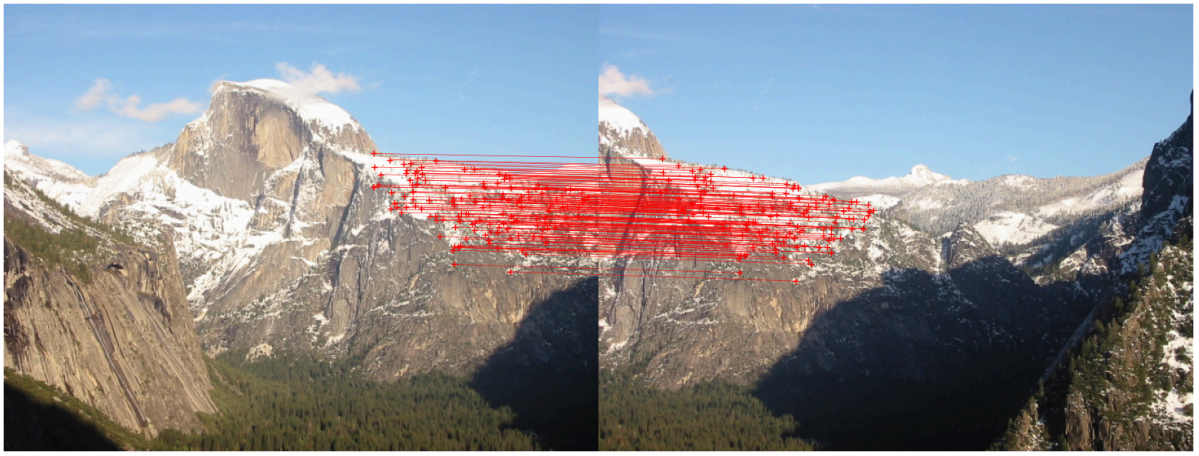
1-3



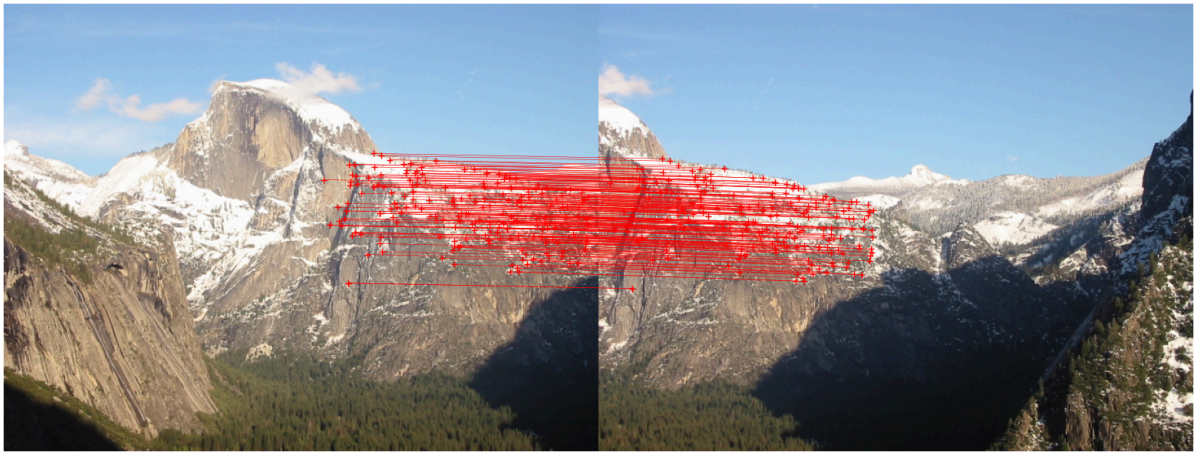
1-4

Experiment on different iteration_times

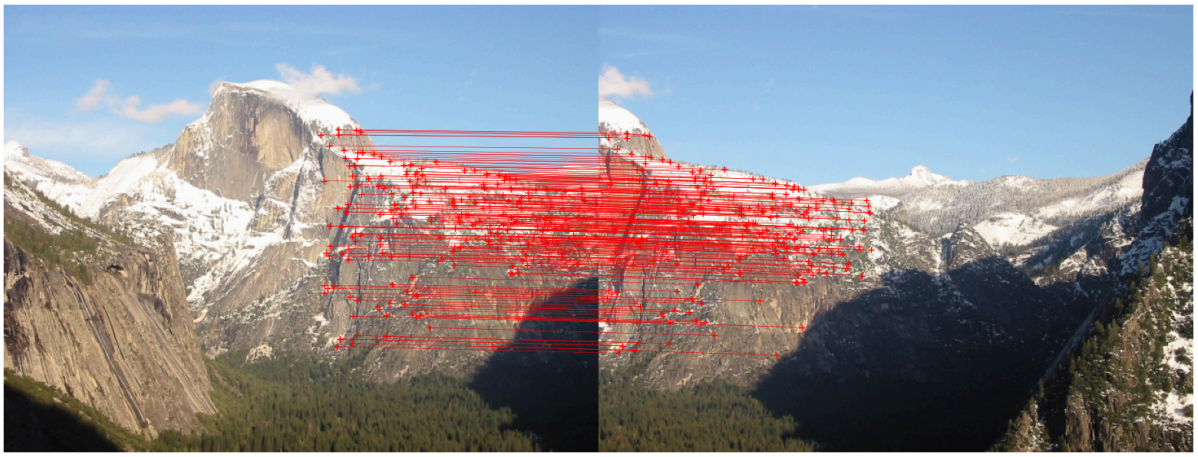
iteration_times = 10, threshold = 0.3, number of inliners = 166, Average residual: 0.031



iteration_times = 100, threshold = 0.3, number of inliners = 240, Average residual: 0.029



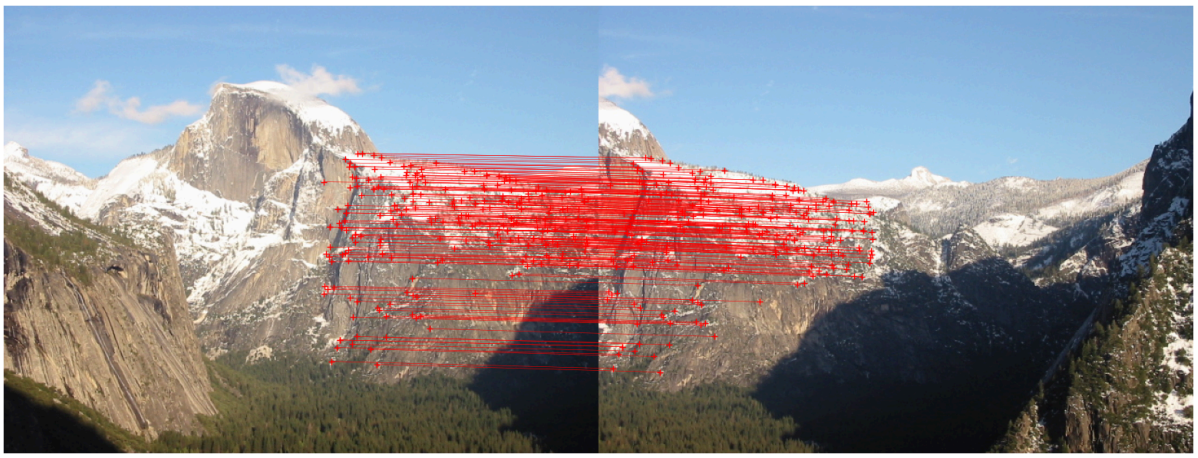
iteration_times = 500, threshold = 0.3, number of inliners = 255, Average residual: 0.028



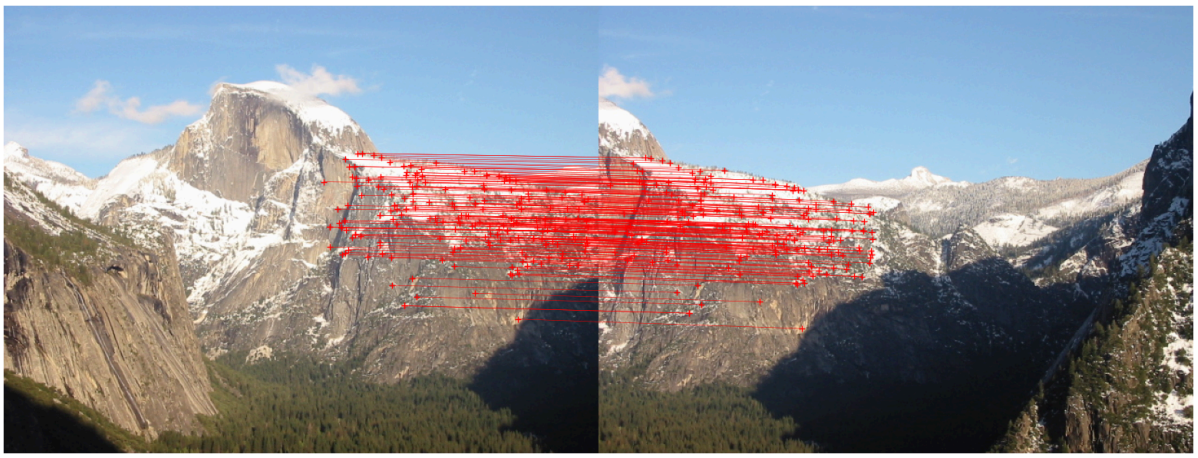
Description: With more iteration times, the residual slightly declines, and the number of inliners increases.

Experiment on different thresholds

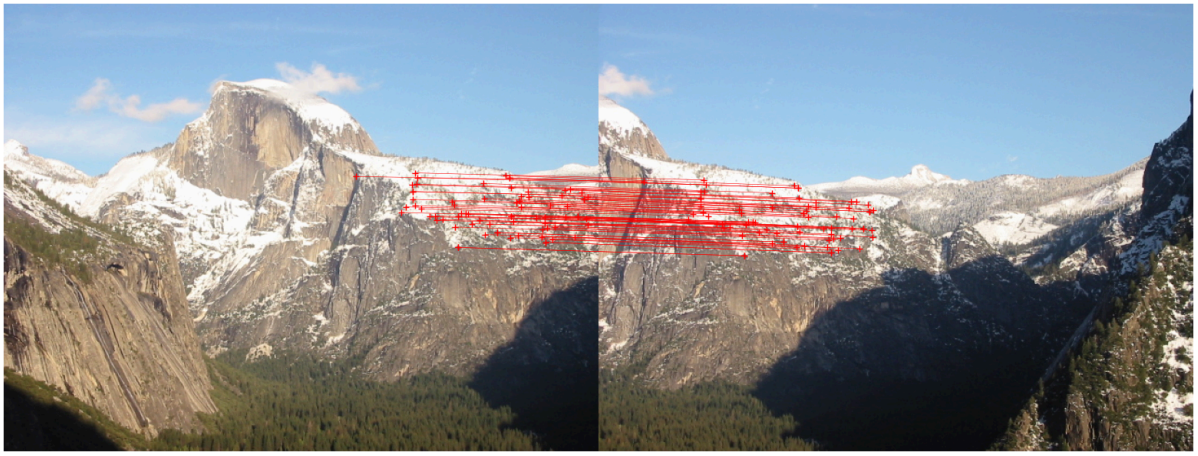
iteration_times = 100, threshold = 0.3, number of inliners = 240, Average residual: 0.029



iteration_times = 100, threshold = 0.2, number of inliners = 183, Average residual: 0.013



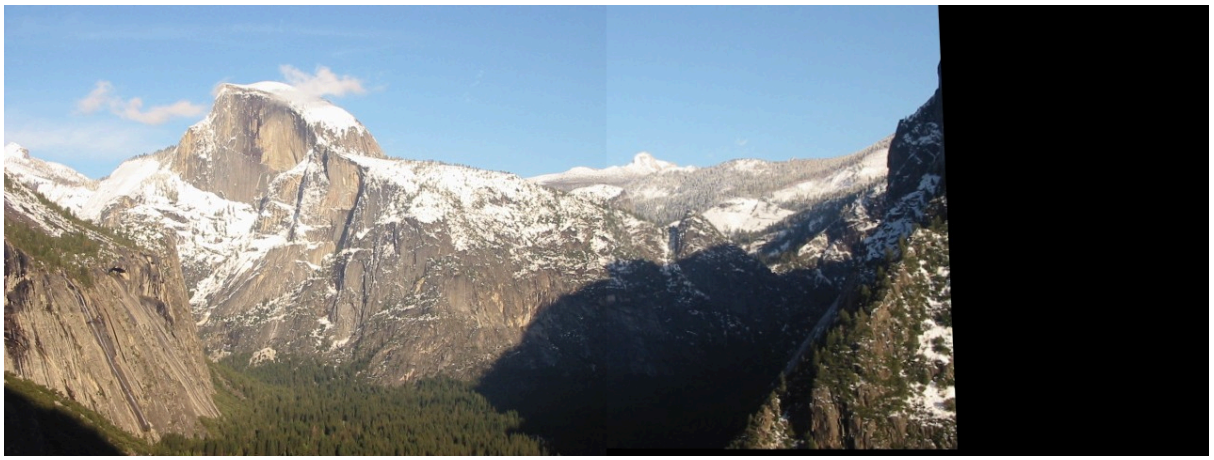
iteration_times = 100, threshold = 0.1, number of inliners = 74, Average residual: 0.003



Description: With a tighter (smaller) threshold, fewer keypoint pairs will be selected as inliners. Also, since only pairs with less distances will be selected as inliners, the residual will also decrease.

Implementation details:
We run # of iteration times of a loop. For each time running in the loop, we randomly pick 4 pairs of keypoints to calculate the H matrix. Then, we transform (x1, y1) using H and subtract it with (x2, y2) to get the distance of point1 and point2. If the distance is lower than the threshold we set, then this pair can be counted as an inliner. After each iteration, if the number of inliners of the new H is larger than the current best model, the H will become the best model.

1-5
stitched image



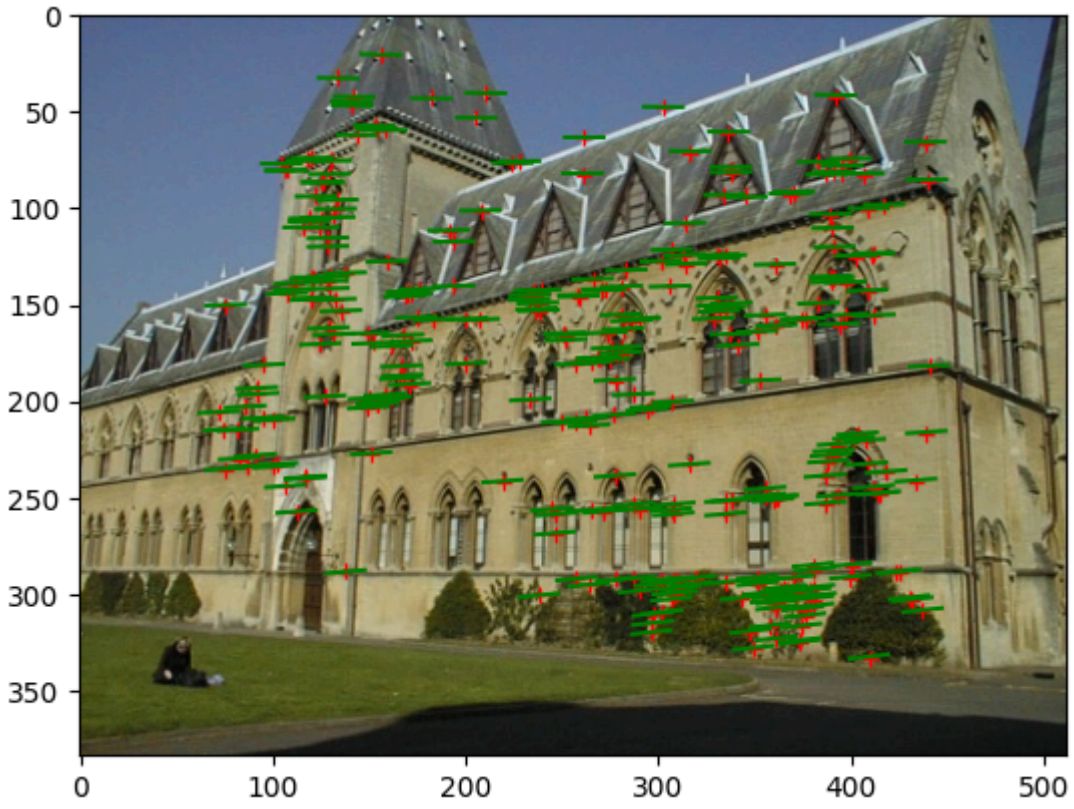
2-1

estimated fundamental matrix

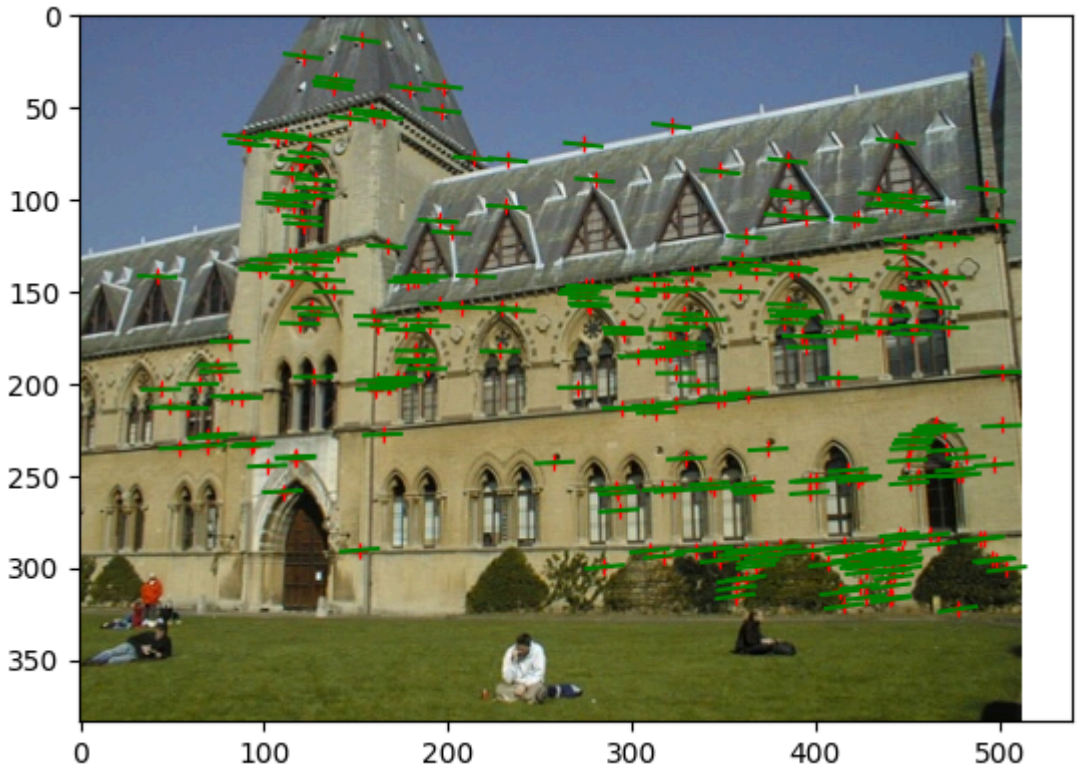
the residual (mean squared distance (in pixels) between points in both images and the corresponding epipolar line)
library: residual in frame 2 (non-normalized method) = 0.17921336680691263
library: residual in frame 1 (non-normalized method) = 0.14912309939068902
library: residual combined (non-normalized method) = 0.16416823309880083
library: residual in frame 2 (normalized method) = 0.060252440893610676
library: residual in frame 1 (normalized method) = 0.05482266577363878
library: residual combined (normalized method) = 0.05753755333362473

visualization of epipolar lines and corresponding points

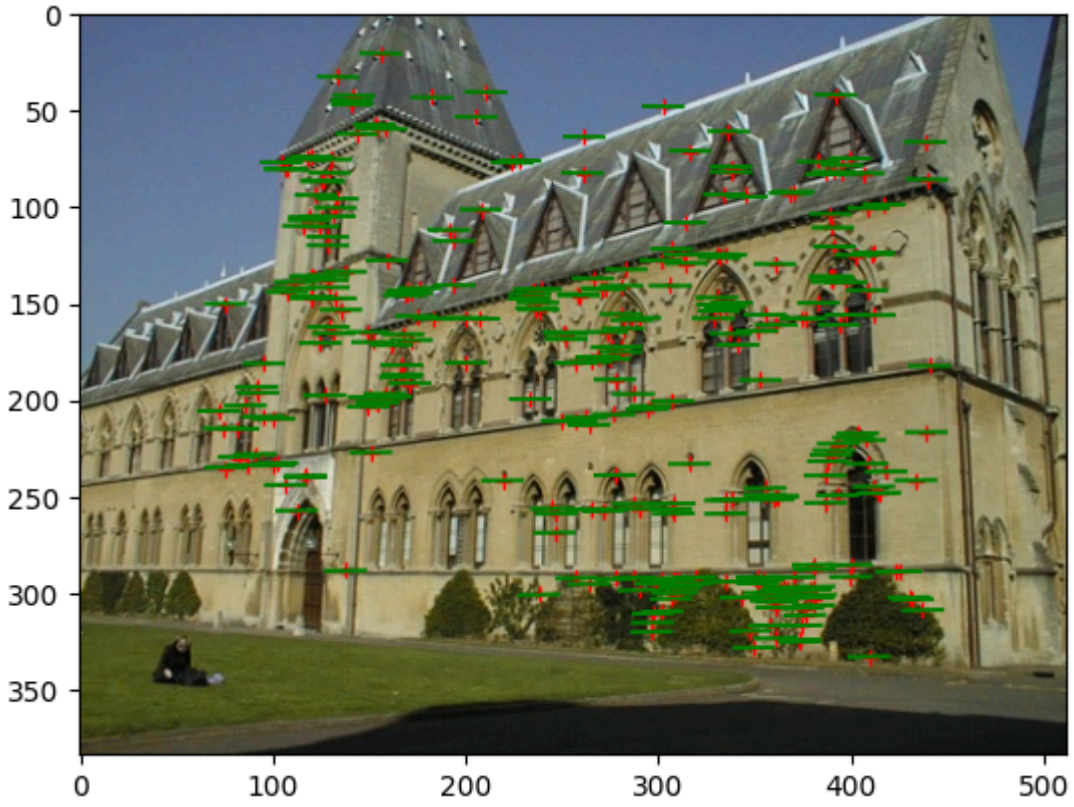
non normalized - frame 2



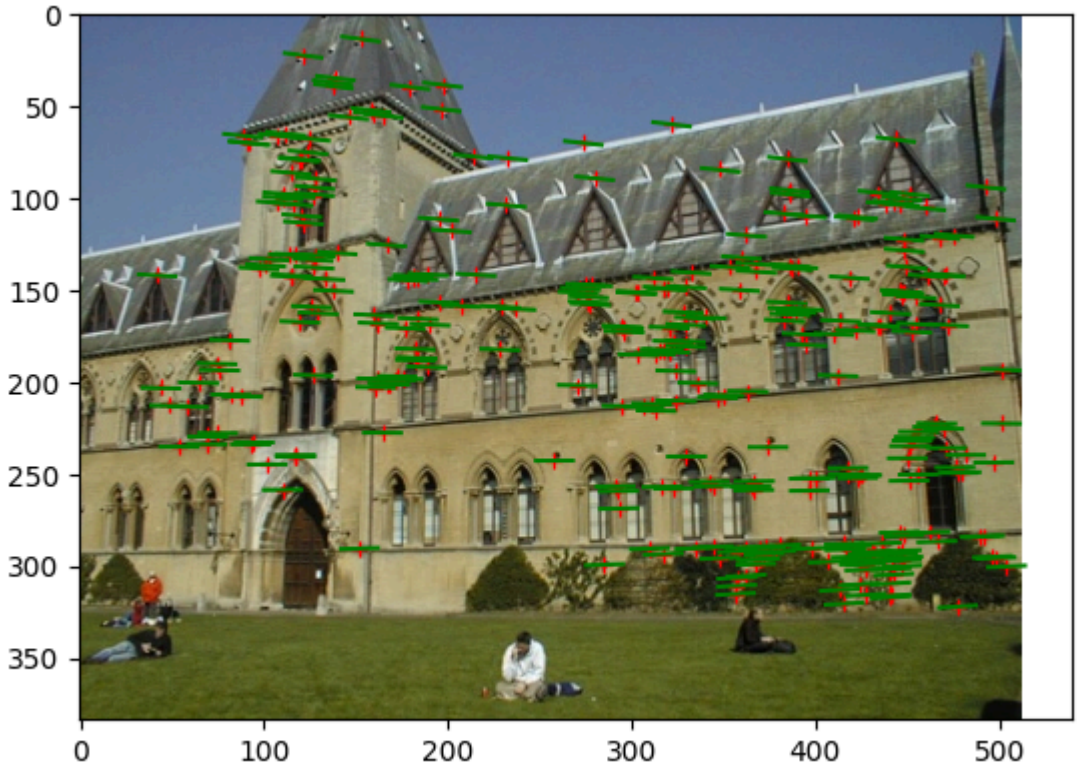
non normalized - frame 1



normalized - frame 2



normalized - frame 1



2-2

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]]

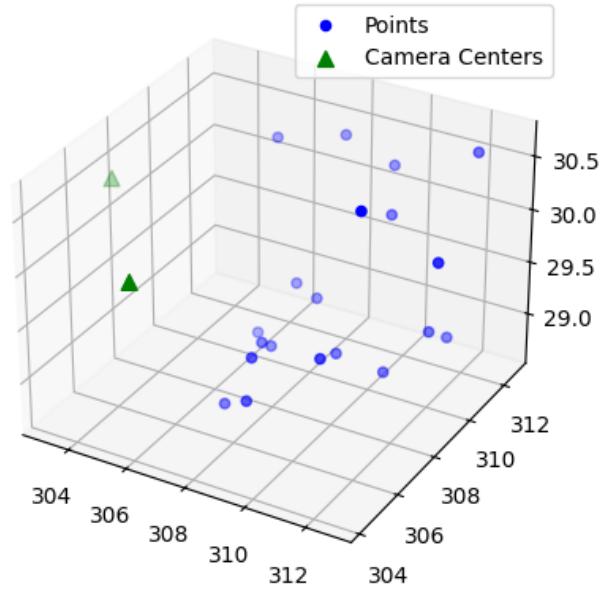
residuals between the observed 2D points and the projected 3D points:
residual in lab1: 13.545832895366306
residual in lab2: 15.544953453719087

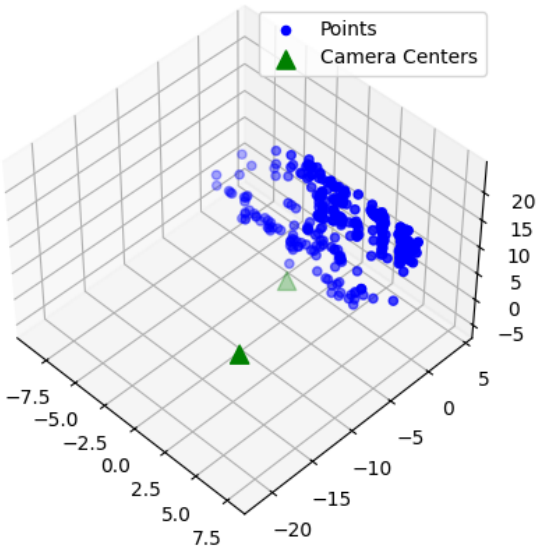
library1 camera projection
[[-4.5250208e+01 4.8215478e+02 4.0948922e+02 3.4440464e+03]
 [4.8858466e+02 2.7346374e+02 -1.3977268e+02 4.8030231e+03]
 [-1.9787463e-01 8.8042214e-01 -4.3093212e-01 2.8032556e+01]]
library2 camera projection
[[-5.9593834e+01 5.5643970e+02 2.3093716e+02 3.5683545e+03]
 [4.6419679e+02 2.2628430e+02 -1.9605278e+02 4.8734171e+03]
 [-1.9116708e-01 7.2057697e-01 -6.6650130e-01 2.8015392e+01]]

2-3

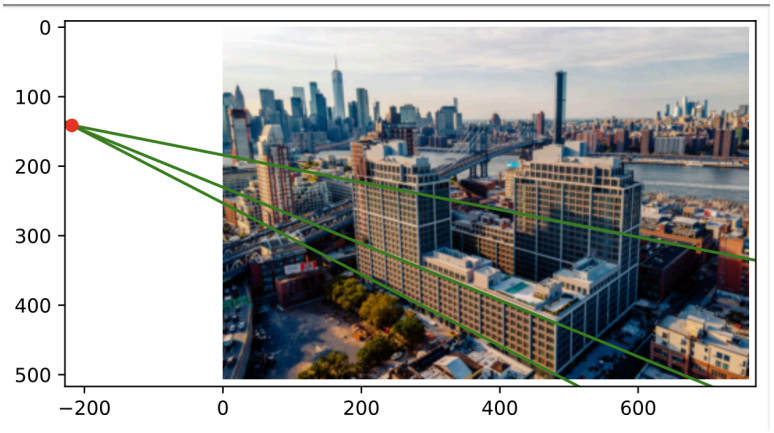
lab1 camera center [305.83276769 304.20103826 30.13699243]
lab2 camera center [303.10003925 307.18428016 30.42166874]
library1 camera center [7.28863053 -21.52118112 17.73503585]
library2 camera center [6.89405488 -15.39232716 23.41498687]

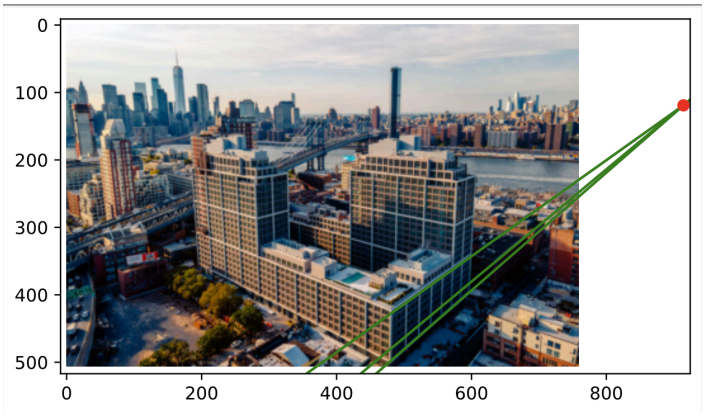
2-4
Mean 3D reconstruction error for the lab data: 0.01332
2D reprojection error for the lab 1 data: 10.899446031539766
2D reprojection error for the lab 2 data: 1.5485148082894804
2D reprojection error for the library 1 data: 24.662071196870507
2D reprojection error for the library 2 data: 28.6495377352594

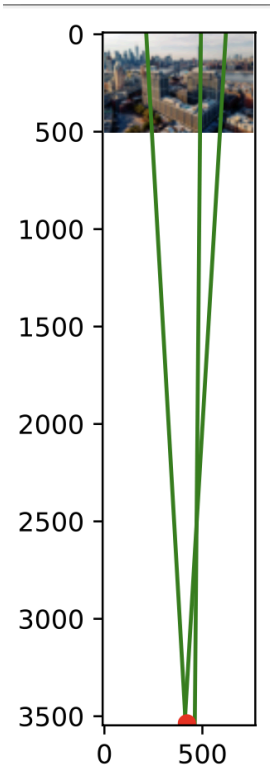




3-1







pixel coordinates
[-218.12925725 141.33875189 1.]
[914.30303401 119.05724645 1.]
[4.21082572e+02 3.53668775e+03 1.00000000e+00]

3-2

horizon_line parameter: y + -130.1980 = 0

3-3
u = 355.906379457695
v = 224.181564060017
f = 558.418109511998
K = [[558.4181 0. 355.90637]
 [0. 558.4181 224.18156]
 [0. 0. 1.]]

My code:

```
def get_camera_parameters(vpts):
    """
    Computes the camera parameters. Hint: The SymPy package is suitable for this.
    """
    vpts = vpts / vpts[-1, :]
    vpt1, vpt2, vpt3 = [sp.Matrix(vpt) for vpt in vpts.T]

    px, py, f = sp.symbols('px py f')
    x1, y1 = vpt1[0], vpt1[1]
    x2, y2 = vpt2[0], vpt2[1]
    x3, y3 = vpt3[0], vpt3[1]
```

```
    # Setup the equations based on the geometric constraints
    eq1 = sp.Eq((x1 - px) * (x2 - px) + (y1 - py) * (y2 - py) +f**2, 0)
    eq2 = sp.Eq((x1 - px) * (x3 - px) + (y1 - py) * (y3 - py) + f**2, 0)
    eq3 = sp.Eq((x3 - px) * (x2 - px) + (y3 - py) * (y2 - py) + f**2, 0)

    # Solve the system of equations
    solution = sp.solve((eq1, eq2, eq3), (px, py, f), dict=True)[1]

    K = sp.Matrix([[solution[f], 0, solution[px]], [0, solution[f], solution[py]],
[0, 0, 1]])

    return solution[f], solution[px], solution[py], np.array(K).astype(np.float32)
```

3-4
R = [[-0.71298563 0.70091021 0.01939838]
 [-0.10289558 -0.13195407 0.98590088]
 [0.69358772 0.70093715 0.16620192]]