EECS 442 HW3

#1 - RANSAC

1.1

1) We need 2 points to compute a putative model. The formula for deriving the slope explains this - (y2 - y1) / (x2 - x1).

2)
   $(1 - r)^S =$
   $(1 - 0.1)^2 =$
   $(0.99)^2 = 0.98$

3)
   $P_{no\text{-}outlier} = (1 - (1 - r)^S)^N$
   $.95 = (1 - (1 - r)^S)^N$
   $.95 = (1 - (0.99)^2)^N$
   $.95 = (1 - 0.9801)^N$
   $\ln(.95/0.199) = N$
   $N = 1.56 \rightarrow$ **2 trials**

1.2

1) M has 4 degrees of freedom & 2 samples are required to find M

2) Missing

To fit $y := M\bar{x}_i$ for $\bar{x}_i, \bar{y}_i \in \{(\bar{x}_i, \bar{y}_i)\}_{i=1}^{N}$

we must find an M such that : $M = \operatorname{argmin} \sum_{i=1}^{N} \|M\bar{x}_i - \bar{y}_i\|$

$$M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

$$M\bar{x}_i - \bar{y}_i = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix} - \begin{bmatrix} y_{i1} \\ y_{i2} \end{bmatrix}$$

$$= \begin{bmatrix} m_{11}x_{i1} + m_{12}x_{i2} \\ m_{21}x_{i1} + m_{22}x_{i2} \end{bmatrix} - \begin{bmatrix} y_{i1} \\ y_{i2} \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} x_{i1} & x_{i2} & 0 & 0 \\ 0 & 0 & x_{i1} & x_{i2} \end{bmatrix}}_{a_i} \underbrace{\begin{bmatrix} m_{11} \\ m_{12} \\ m_{21} \\ m_{22} \end{bmatrix}}_{m} - \begin{bmatrix} y_{i1} \\ y_{i2} \end{bmatrix}$$

$$M\bar{x}_i - \bar{y}_i = a_i m - y_i$$

$$\sum_{i=1}^{N} \|M\bar{x}_i - \bar{y}_i\|^2 = \|a_i m - y_i\|^2 = \left\| \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix} m - \begin{bmatrix} y_{11} \\ y_{12} \\ \vdots \\ y_{N1} \\ y_{N2} \end{bmatrix} \right\|^2$$

$$A = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & 0 & 0 \\ 0 & 0 & x_{11} & x_{12} \\ & \vdots & & \\ x_{N1} & x_{N2} & 0 & 0 \\ 0 & 0 & x_{N1} & x_{N2} \end{bmatrix}$$

3) S & t matrices

T Matrix

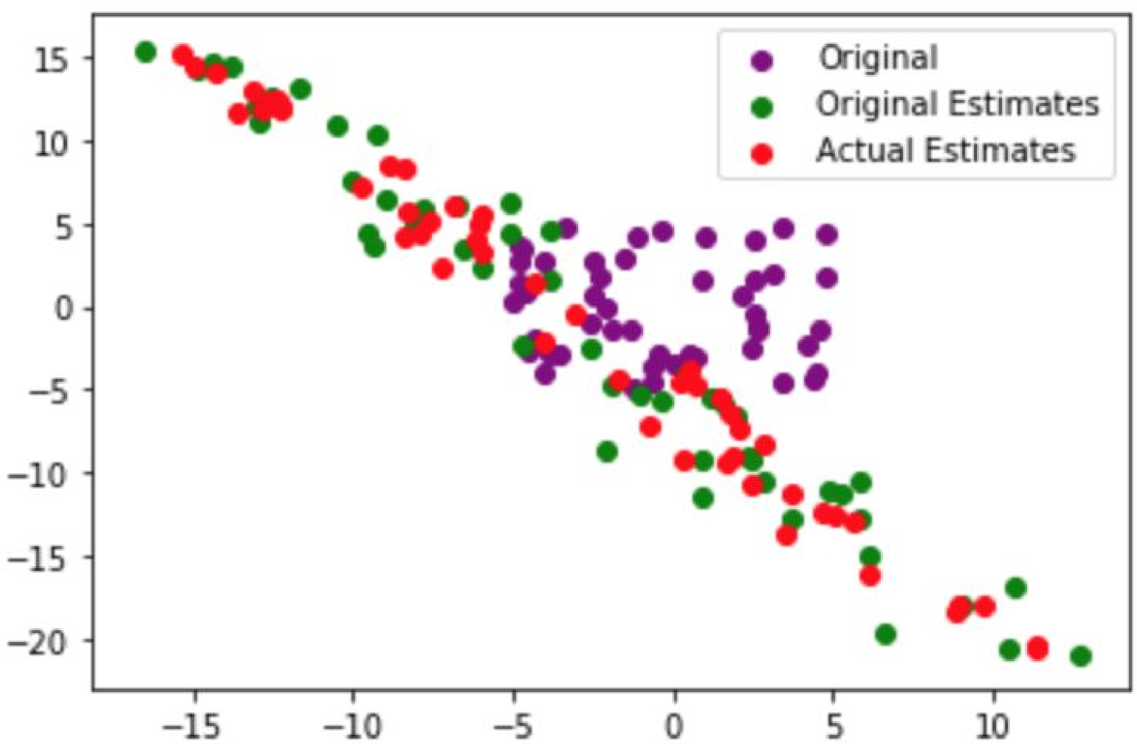
```
array([[-1.87154926],
       [-3.05145812]])
```

S Matrix

```
array([[ 2.04098127, -1.01644528],
       [-3.0696843 ,  0.94335775]])
```

4)

```
<matplotlib.legend.Legend at 0x13f3e7a90>
```

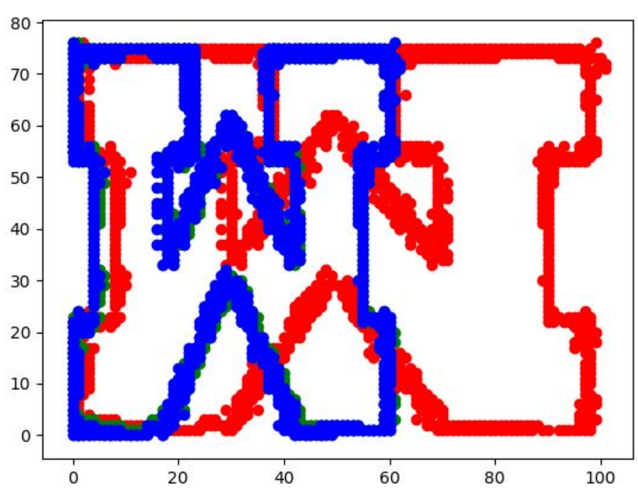


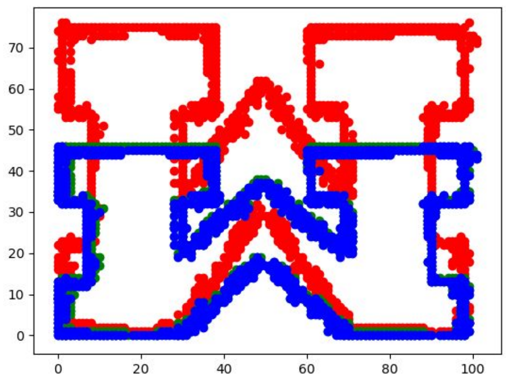
5) In order to transform the points, I take the dot product of the original X & Y points with my S_matrix. For my new x value, I keep the first element from this dot product and for my new y value, I keep the second element from this dot product.

6)

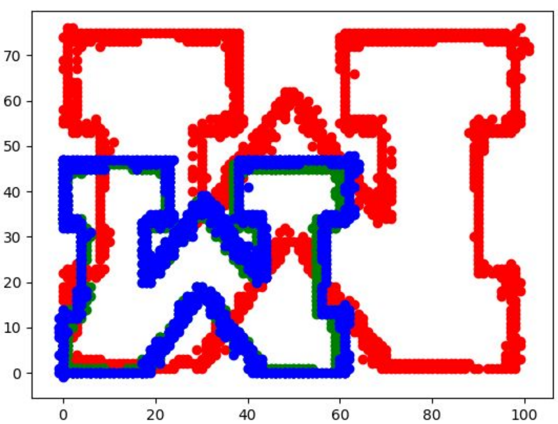
```
case  0
[[ 1.00555949e+00  1.61370672e-03 -1.35143989e-01]
 [ 2.56045861e-03  6.22536404e-01 -7.35872070e-01]
 [ 4.51704286e-05  3.59823762e-05  1.00000000e+00]]
case  1
[[ 6.20394776e-01  1.50050345e-03 -6.60977645e-01]
 [ 4.62565656e-05  1.00232229e+00 -4.06369835e-02]
 [ 3.66035139e-07  2.25931463e-05  1.00000000e+00]]
case  2
[[ 6.46216419e-01  1.01850154e-02 -1.45233663e+00]
 [ 1.39156925e-02  6.46332905e-01 -1.60285342e+00]
 [ 3.11050183e-04  2.76954685e-04  1.00000000e+00]]
case  3
[[ 2.27055861e-14  1.00000000e+00  2.85662774e-13]
 [ 1.00000000e+00 -9.26614147e-16  4.78080376e-14]
 [ 9.61481343e-17  0.00000000e+00  1.00000000e+00]]
case  4
[[-1.00000000e+00 -1.45848347e-13  7.60000000e+01]
 [ 6.59624040e-13 -1.00000000e+00  1.01000000e+02]
 [ 9.52540533e-15  6.85280959e-16  1.00000000e+00]]
case  5
[[ 4.01387046e-01  6.36777993e-01 -1.79445757e+00]
 [ 6.40534201e-01  3.96745666e-01 -1.76059696e+00]
 [ 2.21034153e-04  1.39932850e-04  1.00000000e+00]]
case  6
[[ 7.70056604e-01  4.54255201e-02 -6.34808816e-02]
 [-7.87253712e-02  8.63813194e-01  5.49671671e+00]
 [-3.05270237e-04 -6.37121535e-05  1.00000000e+00]]
case  7
[[ 8.41165566e-01 -4.86333708e-02  4.30457199e+00]
 [-4.77384685e-02  9.86999947e-01  3.56055033e+00]
 [-4.55556648e-04  1.08171485e-04  1.00000000e+00]]
```

7) Cases 0 through 8

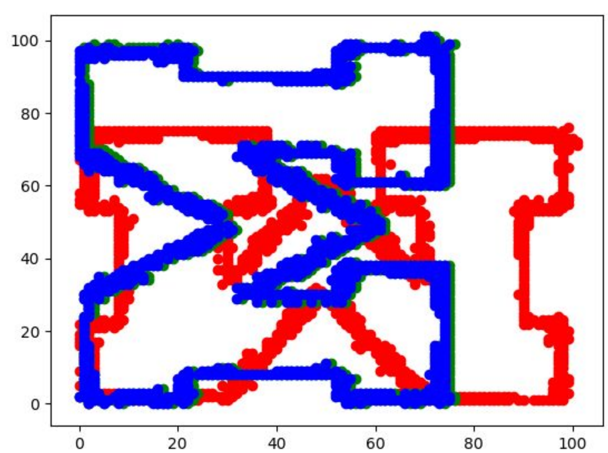




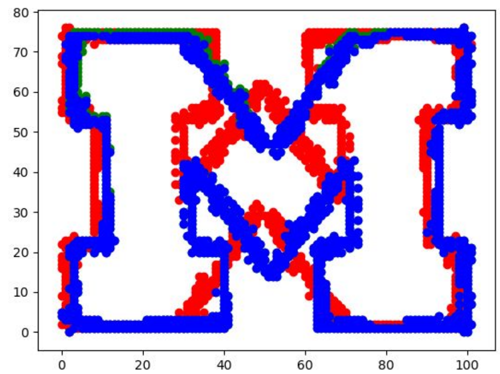
0
10
20
30
40
50
60
70
0
20
40
60
80
100

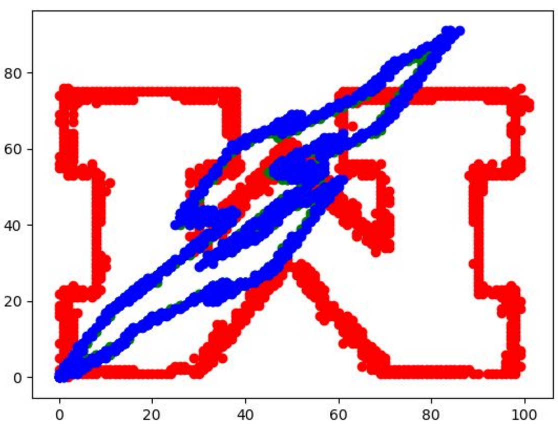


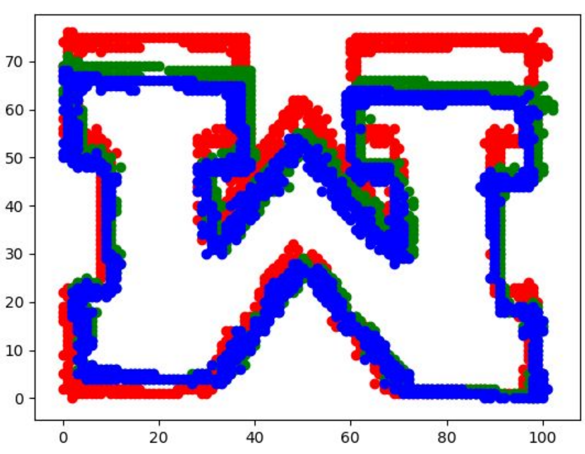
0
10
20
30
40
50
60
70
0
20
40
60
80
100



0
20
40
60
80
100
0
20
40
60
80
100







8) These transformations are unsurprising given the espoused function of a homography matrix. Obviously, this does not result in full coverage given that green is shown in many areas.

#2 - Image Stitching

1) Grayscale images

2) Feature Points

3) Distance was computed using Euclidean distance. Specifically:

```
distance = np.sqrt(np.sum((des1[:, np.newaxis, :] - des2[np.newaxis, :, :]) **
2, axis=-1))
```

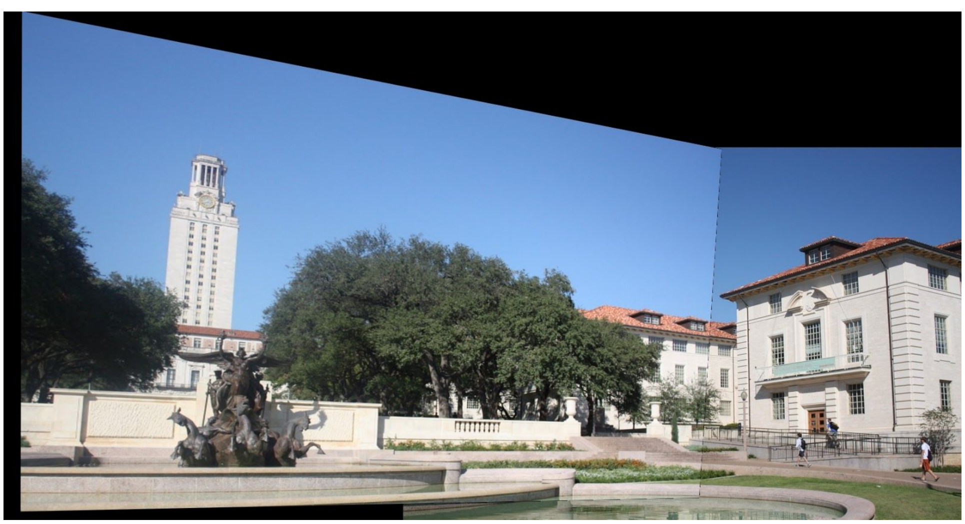
4) The top 250 matches with the least distance were selected. Specifically, this function was used where the 'a' was the distance matrix and N was 250.

```
def smallestN_indices(a, N):
    idx = a.ravel().argsort()[:N]
    return np.stack(np.unravel_index(idx, a.shape)).T
```

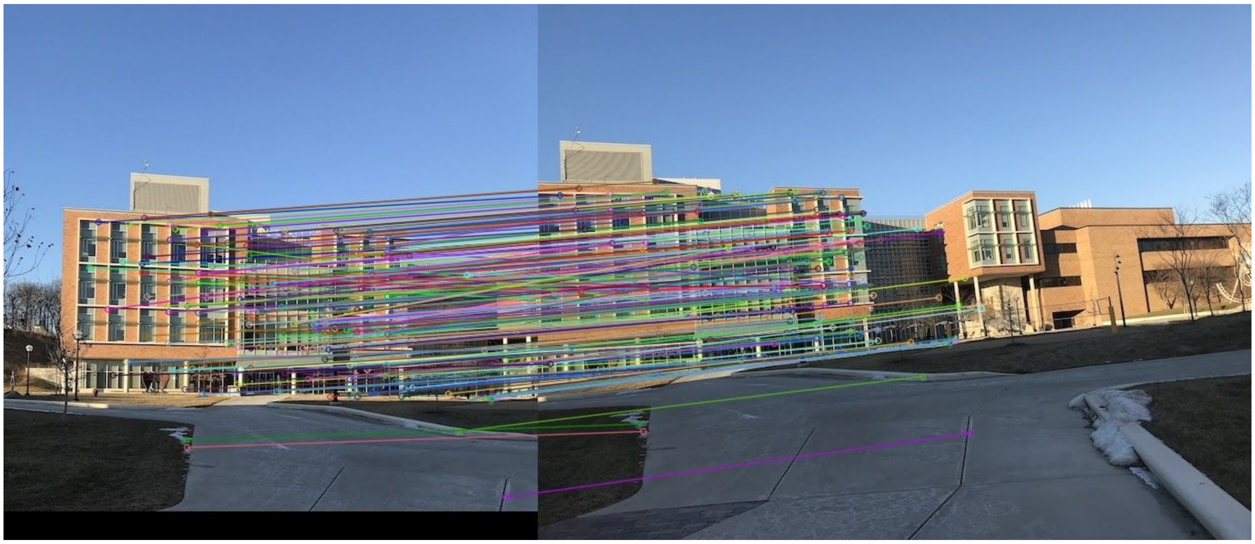
5) Matches → 199 inliers were found where the average distance of the residual was 97.0306. However a distance of less than 5 was required to qualify as an inlier.

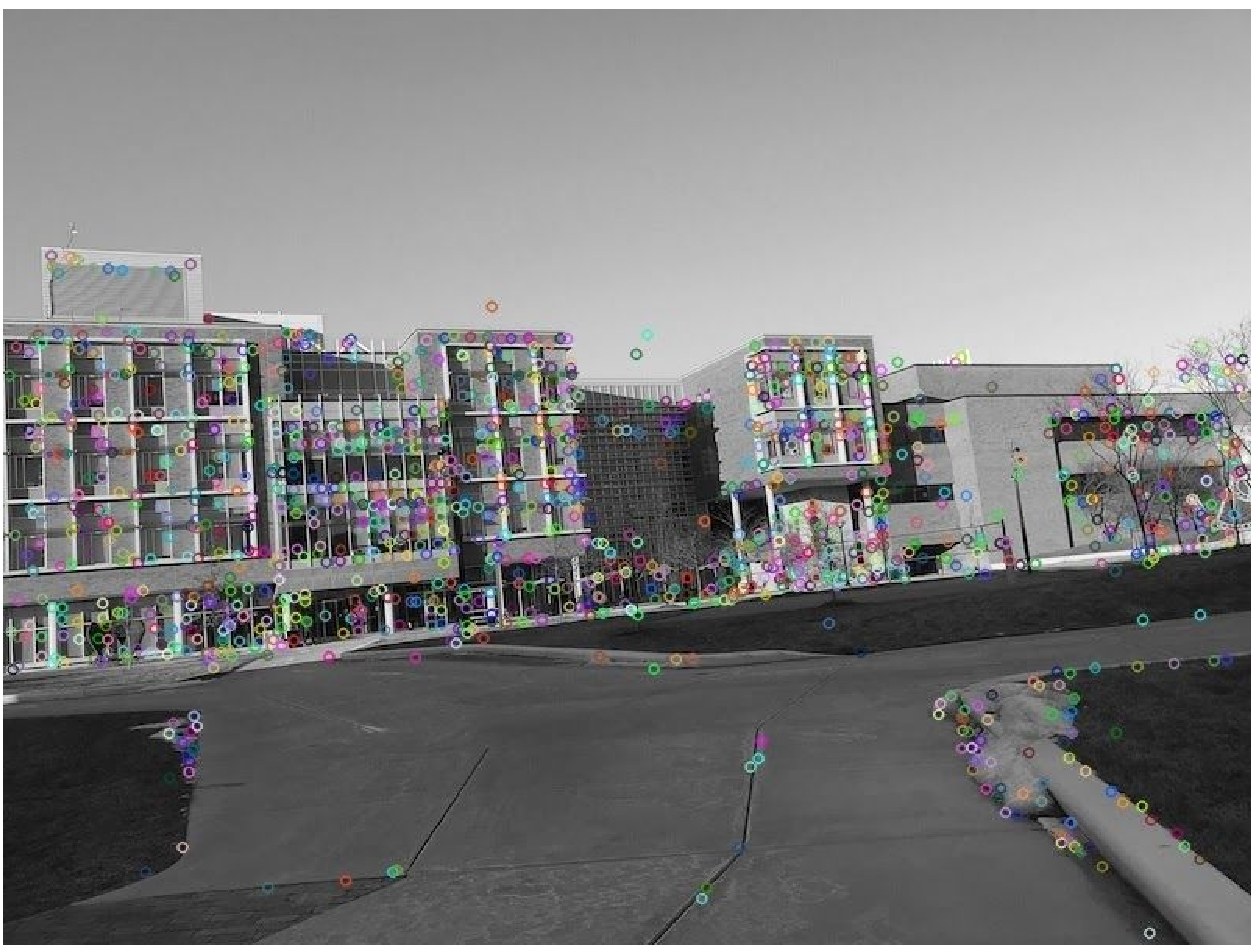
6) N/A → see code

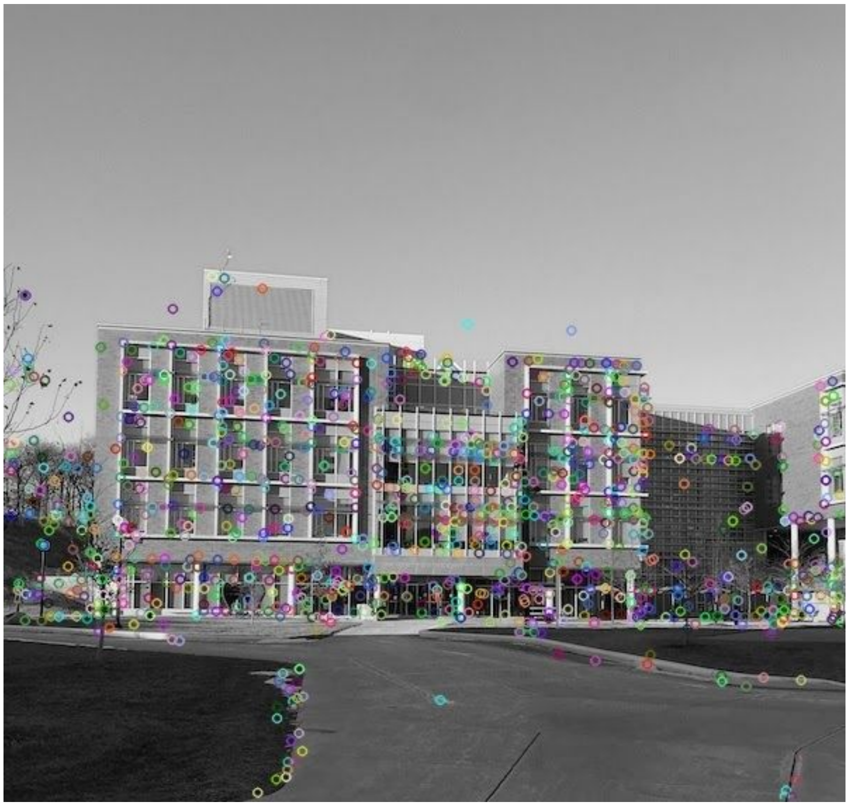
7) Stitched image.



8) BBB image.







159 inliers with an average residual of 10.906617376795804.