Question 1: H matrix computed using DLT (20pts)

$$\begin{bmatrix} 0 & 0 & 0 & x_1 & y_1 & 1 & y_1'x_1 & y_1'y_1 & y_1' \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & x_m & y_m & 1 & y_m'x_m & y_m'y_m & y_m' \\ x_m & y_m & 1 & 0 & 0 & 0 & -x_m'x_m & -x_m'y_m & -x_m' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

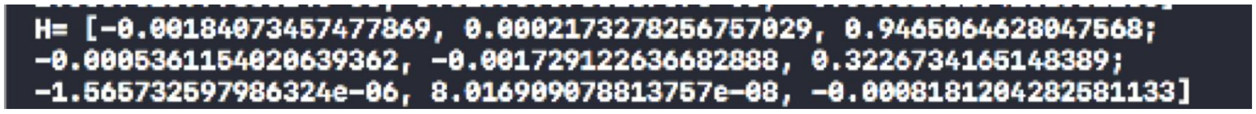
The above equation lead to the expression:

$$Ah = 0$$

,which is equivalent to

$$minimize\ ||Ah||\ subject\ to\ ||h|| = 1$$

Solving the above equation with SVD and find h correspond to the last column of v



```
H= [-0.00184073457477869, 0.0002173278256757029, 0.9465064628047568;
-0.0005361154020639362, -0.001729122636682888, 0.322673416514838 9;
-1.565732597986324e-06, 8.016909078813757e-08, -0.0008181204282581133]
```

Source code: see attached document

Question 2: H matrix computed using normalized DLT. In your report, please tell me how do you normalize the images. (20pts)

For the numerical calculation issues in the DLT algorithm, a normalization process should be applied. Given n ≥ 4 point correspondences X_i and X_i' , a similarity transformation T_1

$$T_1 = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

which consists of a translation and scaling will take points X_i to a new set of points $X_i' = T_1 X_i$ such that the centroid of the new points has the coordinate (0, 0)T , and their average distance from the origin is $\sqrt{2}$. Suppose $X_i = (x_i,\ y_i,\ 1)^T$, we have

$$\widehat{X}_i = T_1\vec{X} = \begin{bmatrix} sx_i + t_x \\ sy_i + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix}$$

And the centroid we are seeking is (0, 0, 1), thus we can obtain the centroid below:

$$\begin{bmatrix} s\overline{x_i} + t_x \\ s\overline{y_i} + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Thus, we have $t_x = -s\overline{x}_i,\ t_y = -s\overline{y}_i$

The average distance $\widehat{X}_i = \sqrt{2}$ is calculated below:

$$\begin{aligned} &\frac{1}{n}\sum_i^m \sqrt{\hat{x}_i^2 + \hat{y}_i^2} \\ &= \frac{1}{n}\sum_i^m \sqrt{(sx_i - s\overline{x})^2 + (sy_i - s\overline{y})^2} \\ &= \frac{s}{n}\sum_i^m \sqrt{(x_i - \overline{x})^2 + (y_i - \overline{y})^2} \end{aligned}$$

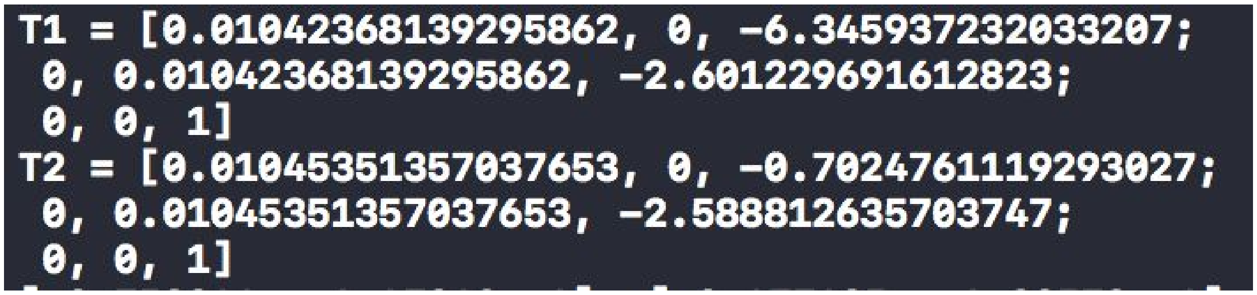
$$s = \frac{\sqrt{2}}{\frac{1}{m}\sum_i^m \sqrt{(x_i - \overline{x})^2 + (y_i - \overline{y})^2}}$$

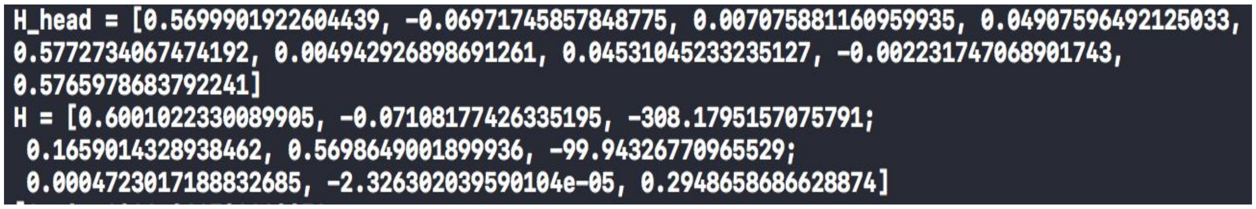
Similarly, a similarity transformation T2 will take points from image2 to the normalized points. Apply DLT to the correspondence $\widehat{X}_i$ and $\widehat{X'_i}$, we obtain a homograph $\widehat{H}$

$$X'_i = HX_i = T_2^{-1}\widehat{X'_i} = T_2^{-1}\widehat{H}\widehat{X}_i = T_2^{-1}\widehat{H}T_1X_i$$

Thus the desired homograph is

$$H = T_2^{-1}\widehat{H}T_1$$


```
T1 = [0.01042368139295862, 0, -6.345937232033207;
 0, 0.01042368139295862, -2.601229691612823;
 0, 0, 1]
T2 = [0.01045351357037653, 0, -0.7024761119293027;
 0, 0.01045351357037653, -2.588812635703747;
 0, 0, 1]
```


```
H_head = [0.5699901922604439, -0.06971745857848775, 0.007075881160959935, 0.04907596492125033,
0.5772734067474192, 0.004942926898691261, 0.04531045233235127, -0.002231747068901743,
0.5765978683792241]
H = [0.6001022330089905, -0.07108177426335195, -308.1795157075791;
 0.1659014328938462, 0.5698649001899936, -99.94326770965529;
 0.0004723017188832685, -2.326302039590104e-05, 0.2948658686628874]
```

Source code: see attachedments

Question 3: H matrix computed using MLE (modifying DLT's objective function to apply MLE,. Note: you can use Sampson Error instead of reprojection error) (40 pts)

The error calculation in question 1 and questions are algebraic error for one image only and assuming the coordinate for other image is accurant. However, this is not true for the real cases. Now, we can consider the error exists in both images and if the true correspondence are { $\overline{X} \leftrightarrow H\overline{X_i}$ }, then the probability density function of the noise-perturbed data is

$$Pr(\{x_i, x'_i\}|H, \{\overline{x_i}\}) = \prod_i (\frac{1}{2\pi\sigma^2}) e^{-(d(x_i - \overline{x_i}) + d(x'_i, H\overline{x_i}))/(2\sigma^2)}$$

The maximum likelihood has to be maximized for the given matching data points for right estimation of homography H and true correspondence. Thus, take the log of this PDF and maximize the expression leads to the below expression:

$$\sum_{i=1}^{m} d(x_i - \overline{x_i})^2 + d(x'_i - \overline{x'_i})^2$$

Then, the ML estimate is identical with minimizing the reprojection error function. However, minimization of the projection error is difficult, so Sampson error can be used as alternative to the projection error. According to the textbook, the Sampson error can be described as

$$\sum_{i}^{m} ||\delta_x||^2 = \sum_i \varepsilon_x^T (JJ^T) \varepsilon_x^T$$

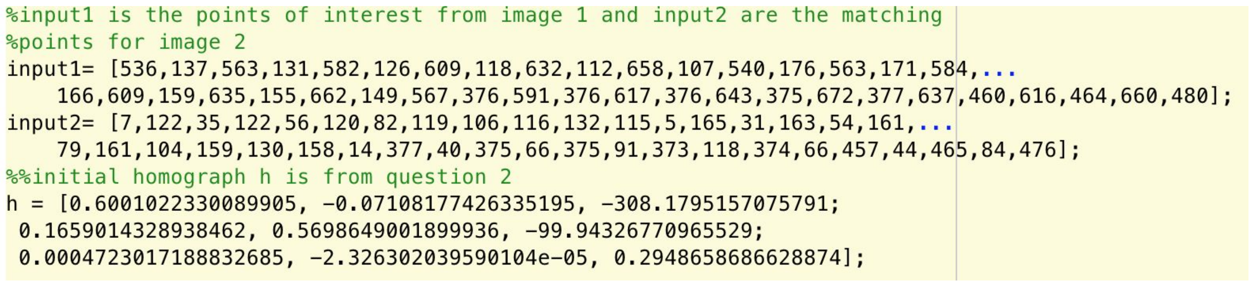
Here, ε is the algebraic error C(H) = |Ah| and J is the Jacobian matrix of ε with respect to x , which is more explicitly written as:

$$C(H) = \begin{bmatrix} x_1 h_4 + y_1 h_5 + h_6 + y'_1 x_1 h_7 + y'_1 y_1 h_8 + y'_1 h_9 \\ x_1 h_1 + y_1 h_2 + h_4 - x'_1 x_1 h_7 - x'_1 y_1 h_8 - x'_1 h_9 \end{bmatrix}$$

$$J(H) = \begin{bmatrix} h_4 + y'_1 h_7 & h_5 + y'_1 h_8 & 0 & x_1 h_7 + y_1 h_8 + h_9 \\ h_1 - x'_1 h_7 & h_2 - x'_1 h_8 & -x_1 h_7 - y_1 h_8 - h_9 & 0 \end{bmatrix}$$

Therefore, both ε and J depends on the homograph vector H, and this is an nonlinear optimization problem. Now, we can solve this nonlinear optimization problem with Levenberg-Marquardt method with initial parameter of H obtained from the question 2 and question 1. Here, the Matlab optimization function *lsqcurveift* can be used to do this job:

```
h1 = reshape(h',9,1);
opt = optimset('Algorithm','levenberg-marquardt','TolFun',1e-8);
input2 = zeros(length(input),1);
h2 = lsqcurvefit(@fun,h1,input,input2,[],[],opt); % Refined homography by L.M
```



```
%input1 is the points of interest from image 1 and input2 are the matching
%points for image 2
input1= [536,137,563,131,582,126,609,118,632,112,658,107,540,176,563,171,584,...
    166,609,159,635,155,662,149,567,376,591,376,617,376,643,375,672,377,637,460,616,464,660,480];
input2= [7,122,35,122,56,120,82,119,106,116,132,115,5,165,31,163,54,161,...
    79,161,104,159,130,158,14,377,40,375,66,375,91,373,118,374,66,457,44,465,84,476];
%%initial homograph h is from question 2
h = [0.6001022330089905, -0.07108177426335195, -308.1795157075791;
 0.1659014328938462, 0.5698649001899936, -99.94326770965529;
 0.0004723017188832685, -2.326302039590104e-05, 0.2948658686628874];
```

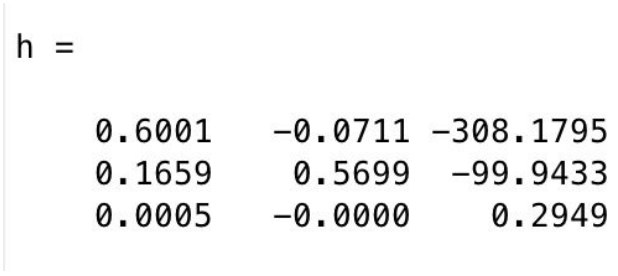
The other parameters are initialized as above for reference, and the initial Sampson error calculated is:

$$\sum_{i}^{m} \|\delta_x\|^2 = \sum_{i} \varepsilon_x^T (JJ^T) \varepsilon_x^T = 118.5779$$

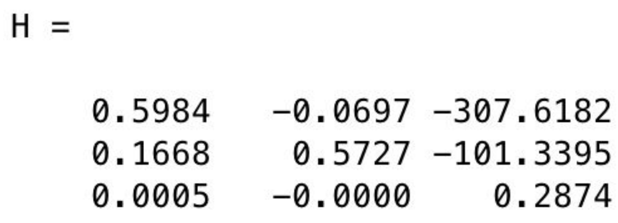
After refinement, the final Sampson error reduced to

$$\sum_{i}^{m} \|\delta_x\|^2 = \sum_{i} \varepsilon_x^T (JJ^T) \varepsilon_x^T = 117.6820$$

Since the H obtained from question 2 is already very close to the true value, so it make sense that the final Sampson error obtained from nonlinear optimization function is just slightly few than the initial error. The initial h and final H calculated are listed here:

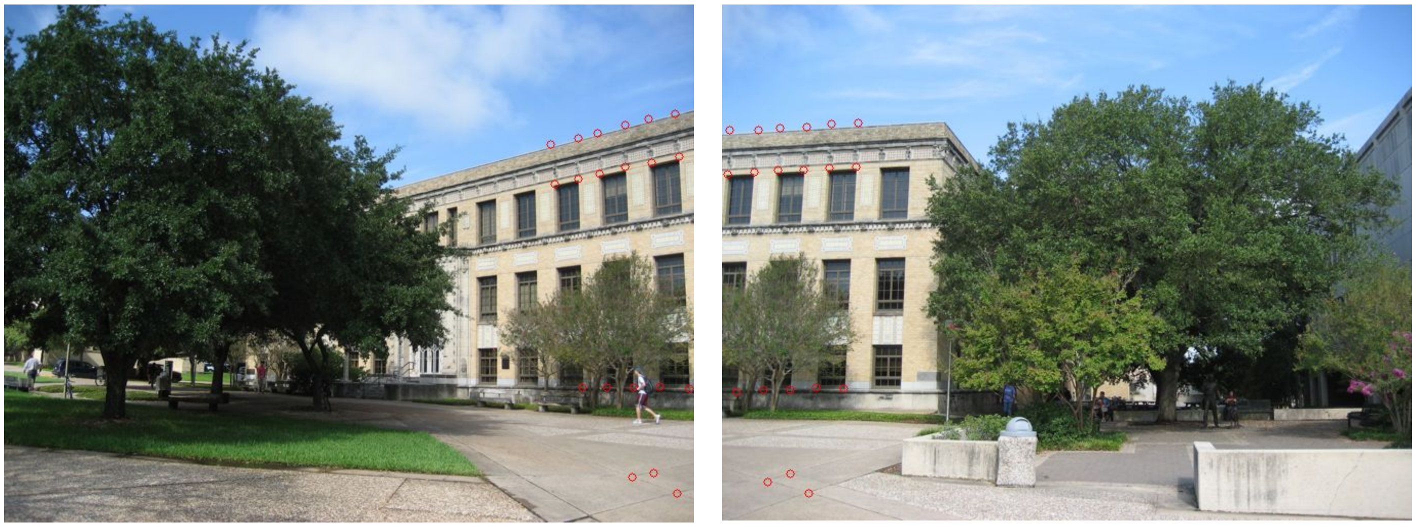


```
h =

    0.6001   -0.0711 -308.1795
    0.1659    0.5699  -99.9433
    0.0005   -0.0000    0.2949
```



```
H =

    0.5984   -0.0697 -307.6182
    0.1668    0.5727 -101.3395
    0.0005   -0.0000    0.2874
```

We can see that there is only slight difference between these two.
Source code: see attachments



Original Image with selected points of interest

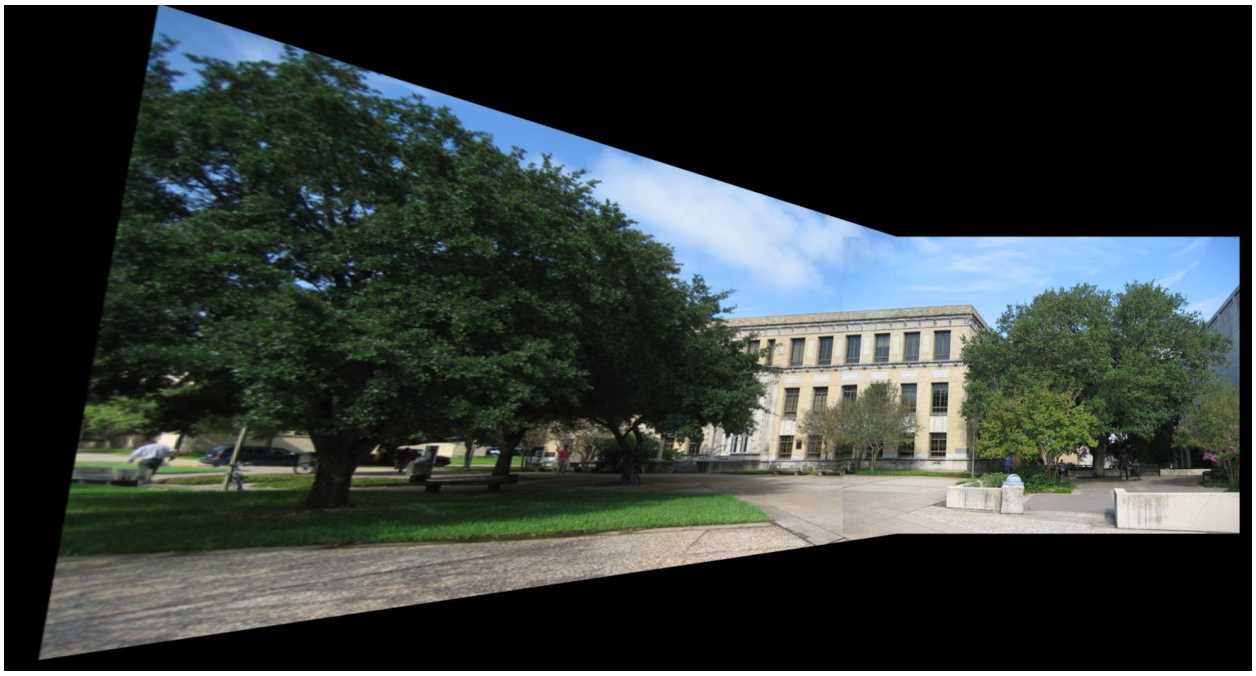


Image with Direct DLT

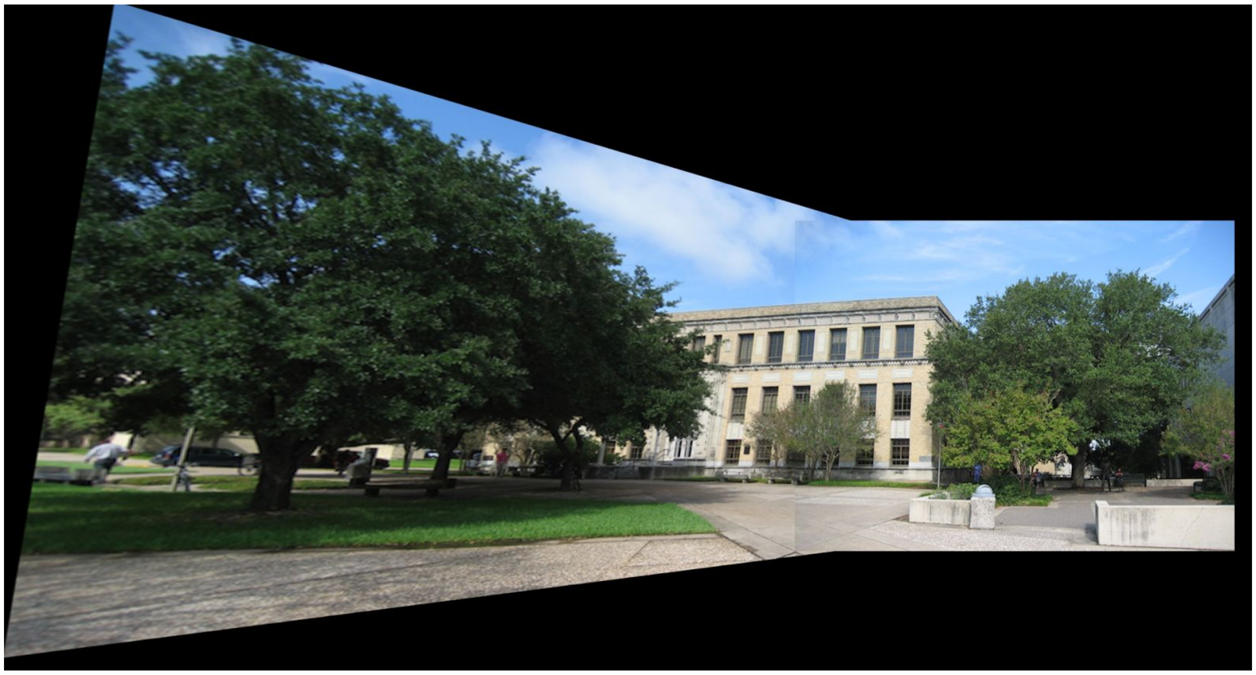


Image out after normalization

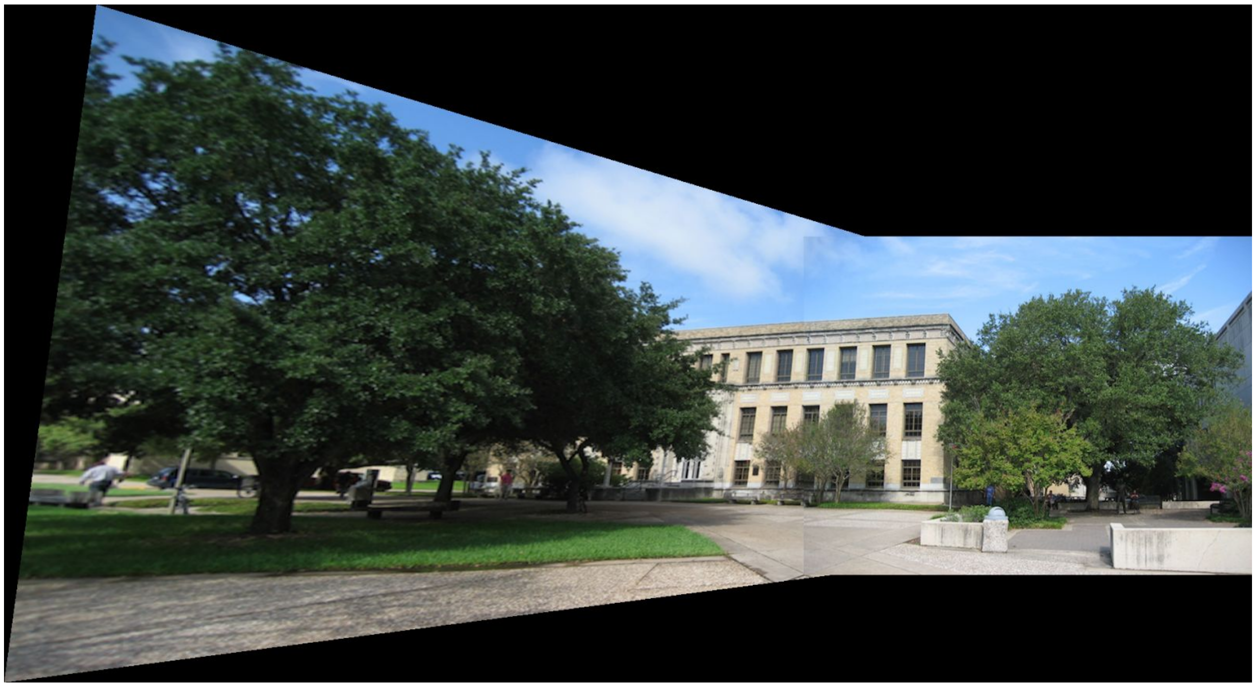


Image out after Sampson error refinement

Challenge:

1. Parameters setting for RANSAC

   - p : The probability that at least one of the N trials will be free of outliers. In this implementation it is set as p = 0.99
   - n : set of correspondences for calculating homography matrix H in each trial. In this implementation it is set as n = 4
   - ε : rough estimation of false correspondences from the total set. In this implementation it is set as = 20/(20+N), N is the total number of false corresponding points
   - δ : decision threshold to construct the inlier set i.e. if the distance between the data point and the estimate is less than δ we place it in the inlier set. In this implementation it is set as $\delta = 3.99 * \sigma^2$ in which σ^2 is the standard deviation obtained from the previous 20 points homography estimation
   - N1: Number of trials, which can be given as $N_1 = \frac{ln(1-p)}{ln(1-(1-\varepsilon)^n)}$, it dynamically change as the number of ε changes
   - M: A minimum value for the size of the inlier set for it to be acceptable, which can be given as $M = n * (1 - \varepsilon)$

2. Program Framework

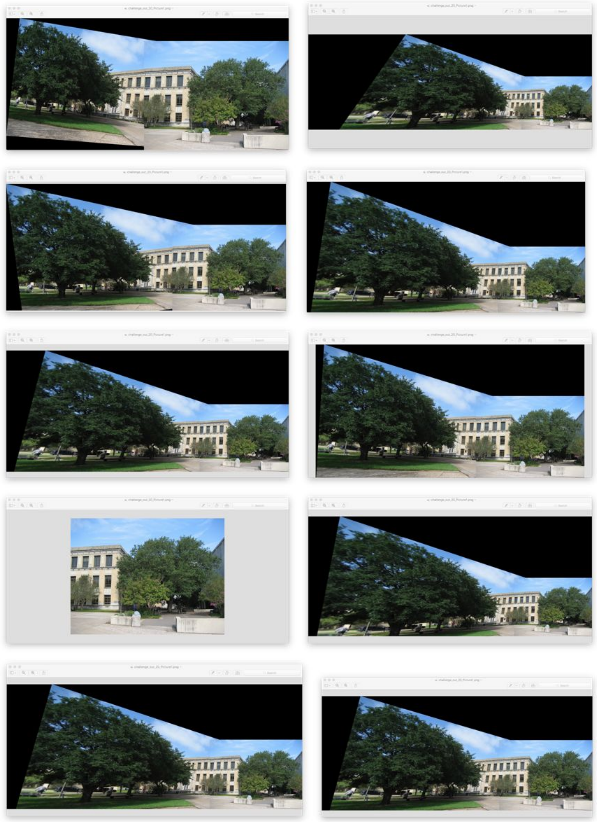
1) Generate random N 2-d points from image a with C++ function *uniform_int_distribution* along the x and y direction respectively and do the same for the second image
2) Determine the total number of running trials according to parameter setting
   $N_1 = \frac{ln(1-p)}{ln(1-(1-e)^s)}$
3) For each of the N trials
   a) Select four random correspondences from the set of input points
   b) Compute the homography between the two images using the algorithms already implemented in question 2. The data of both images is first normalized using a transformation T for the image in the world plane and T for the image in the image plane. The final homography is then given by H = $T_2^{-1} H T_1$.
   c) Compute the distances between the true locations of the correspondences and their measured locations, and add them to a list of inliers if that distance is less than a given threshold δ , distance $d = (x' - Hx)^2 + (x - H^{-1}x')^2$
   d) After N trials, keep the homography that gives the most number of inliers.

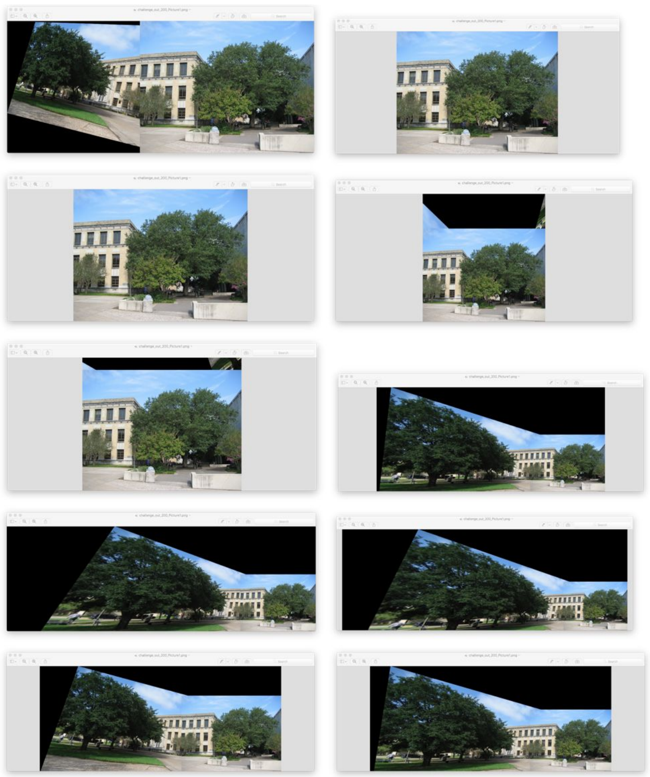
3 Findings:

N	20	200	1000	2000	5000
Trials	72	67422	67422*	67422*	67422*
Time/s	0.055379	44.7604	974.756	408.551	1115.4
Inliners Found	20	20	21	35	60
Accuracy	99%	60%	False**	False**	False**

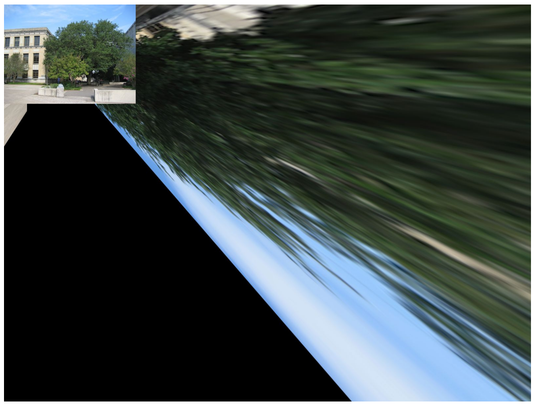
*31154900 is the actual number for N = 1000 but it is too big to run in my machine, so i used 67422*5, 479215857 is the actual number for N = 2000 but it is too big to run in my machine, so i used 67422*5, 18278496896 is the actual number for N = 5000 but it is too big to run in my machine, so i used 67422*5
** False means, it find the inliner, but after checking the actual panoramic image formed by the homography H, it does not give the correct answer, thus the finding is incorrect.

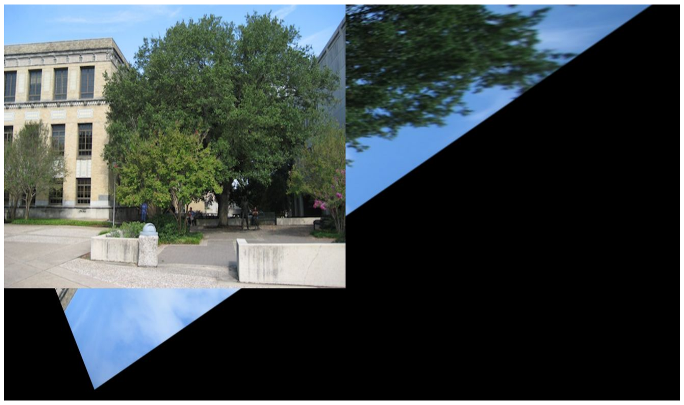
I run the the case N = 20 for 10 times, and manually check the panoramic images to see if the good match has found. In the below pictures, it is clearly found that 9 pictures has good match, and 1 has bad match, so accuracy is 90%. To have more accuracy, more experiments needed.

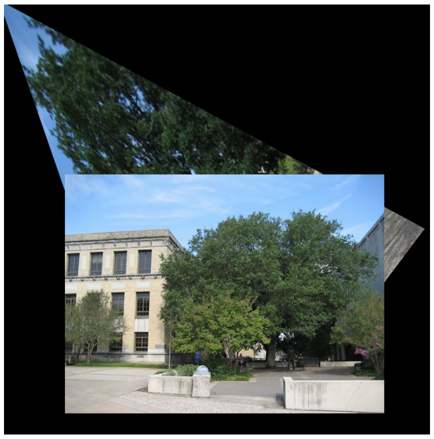

Run N = 20 for 10 times


Run N = 200 for 10 times

The other N = 1000, 2000, 5000 gives more false results,


Panoramic Image with N = 1000


Panoramic Image with N = 1000


Panoramic Image with N = 5000

Discussion:

The panoramic images created with homograph H obtained from the ransac algorithms are listed above and we can see that the images are mismatched when N goes up, which means ransac fails to return the correct matching points pair for large N points. If we look it further, the total number of trials is calculated with $N_1 = \frac{ln(1-p)}{ln(1-(1-e)^s)}$, when the outline ratio e goes up to a large value, the number of trials will also increase significantly, and make computationally expensively to solve, and also tends to give more error. Also, the random generated points might also create some certain of matching or duplicates if we take into consideration of the image dimension is actually less than 1000,2000,5000.

If the number of mismatched points increases, the outlier rate increase also. To maintain the same success probability, more trials are required. According to the formula $(1-(1-e)^s)^{N_1} = 1-p$, it means that the probability of not found correct matching points is 1-p if we choose the right parameter s, and N1. In my code, I choose p = 0.99, so there are still 1% probability which I can not find the right matching points.

In addition to that, the random number generation process can’t generate real random number and only pseudo random number generated, it will also affect the success rate.