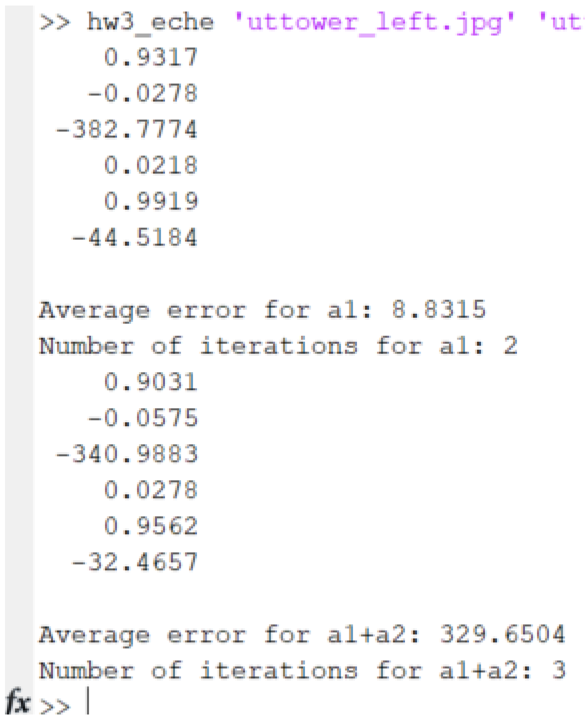
Ethan Che
CS558-A
Homework 3
11/04/2021
I pledge my honor that I have abided by the Stevens Honor System.

NOTE: Requires Computer Vision Toolbox to be installed.

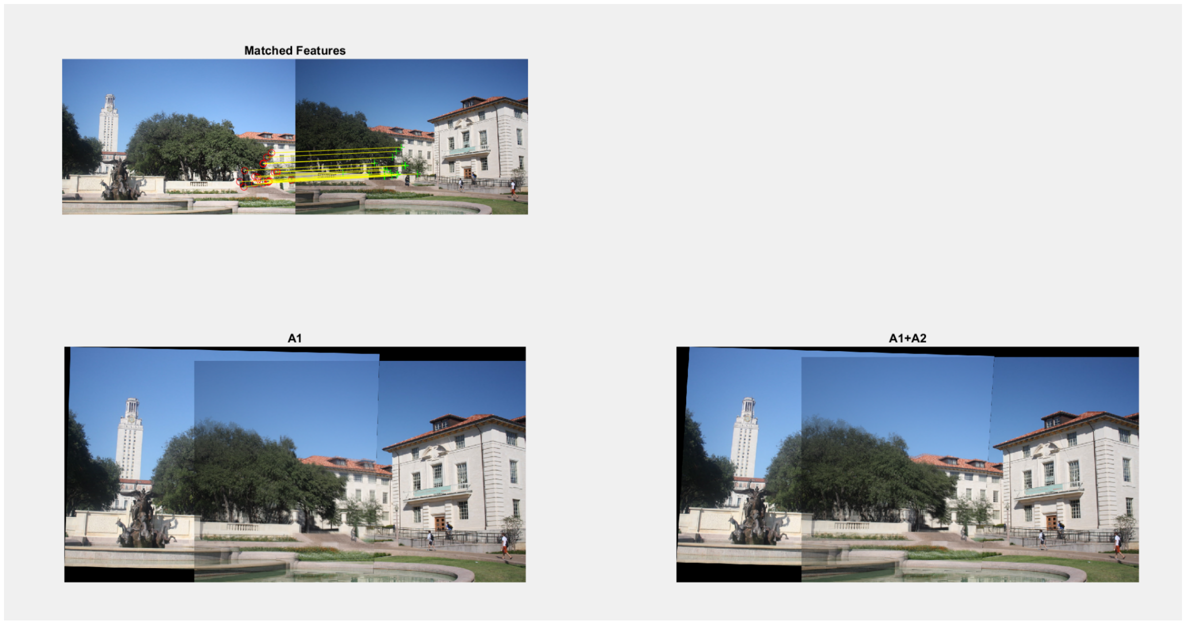
Instructions

- The script will stitch the left and right images together into one panorama. The two images should be in the same directory as the script.
- The script should be ran as follows: hw3_eche 'uttower_left.jpg' 'uttower_right.jpg' stdev
    - stdev is the standard deviation used for the gaussian filter. A standard deviation of 1 produces good results.
    - The script will show the matched features and the two panoramic images. The script also outputs the values of a-f of the affine transform determined by RANSAC, the average error, and number of iterations for both panoramas.
    - 

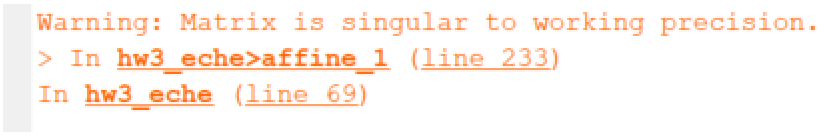
```
>> hw3_eche 'uttower_left.jpg' 'uttower_right.jpg' 1
    0.9317
   -0.0278
 -382.7774
    0.0218
    0.9919
  -44.5184

Average error for a1: 8.8315
Number of iterations for a1: 2
    0.9031
   -0.0575
 -340.9883
    0.0278
    0.9562
  -32.4657

Average error for a1+a2: 329.6504
Number of iterations for a1+a2: 3
fx >> |
```

- 

- Note: when running the program, the following might display:



```
Warning: Matrix is singular to working precision.
> In hw3_eche>affine_1 (line 233)
In hw3_eche (line 69)
```

  - This does not occur frequently, but if it does, you can either let the program finish or halt execution and start over. The results are still good, but it might slow down execution of the program.

Discussions

- Patch similarity measure
  - For this assignment, I used the SSD method for patch similarity instead of the NCC method. I chose to use SSD because it is faster to run. In addition, since the two images are very similar in intensity and contrast, I knew that SSD would give me good results. NCC is slower, but is also invariant to local average intensity and contrast. However, I figured that NCC was not needed for this assignment.
- RANSAC
  - For this assignment, we ran two “experiments” using RANSAC: one with only the top 20 feature correspondences, and one with an additional random 30 correspondences (for a total of 50). For the first method, the expected number of RANSAC iterations is pretty low (around 1), since only the best correspondences are being used to begin with. This is in contrast to the second experiment, which should take more iterations since bad correspondences can be used.
  - These hypotheticals differ in my project. For the first RANSAC experiment, since I knew the feature correspondences were better, I made RANSAC much more

selective: it needed to find parameters that fit 15/20 feature correspondences. The second experiment only needed to find parameters that fit 15/50, since it was more likely that bad lines would be in the calculations. When my code is run, the first RANSAC experiment usually takes a smaller number of iterations than the second. However, since I am being very selective, it is possible for it to take a few more iterations.

Images

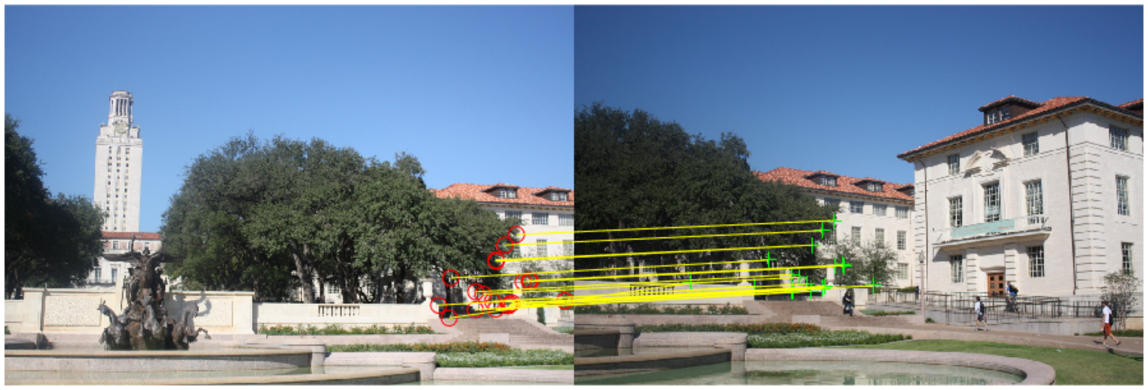
Top1k Left

Top1k Right

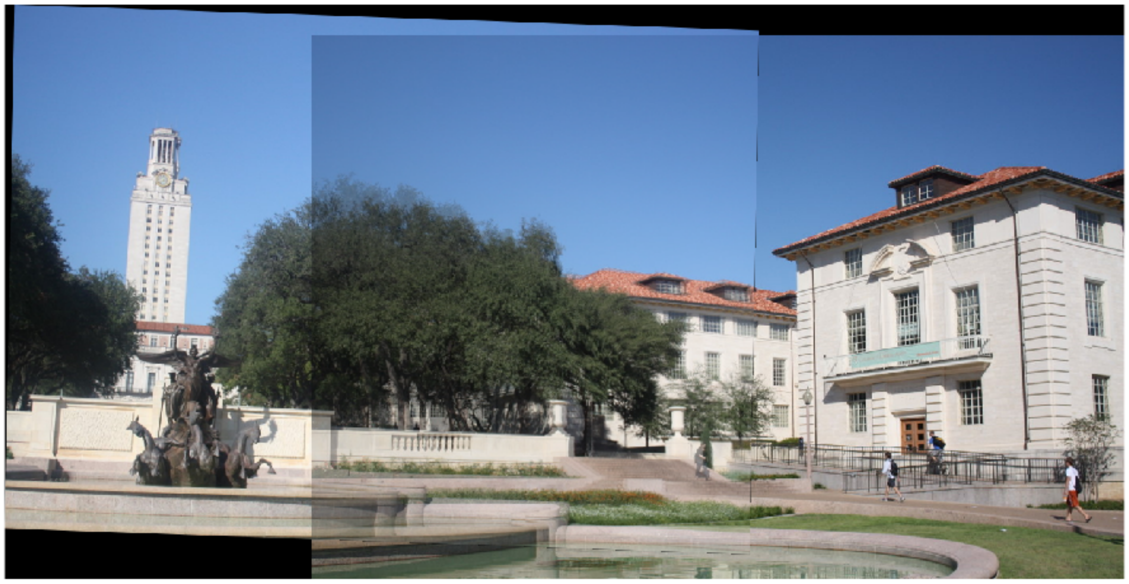
NMS Left

NMS Right

Matched Features



A1



A1+A2