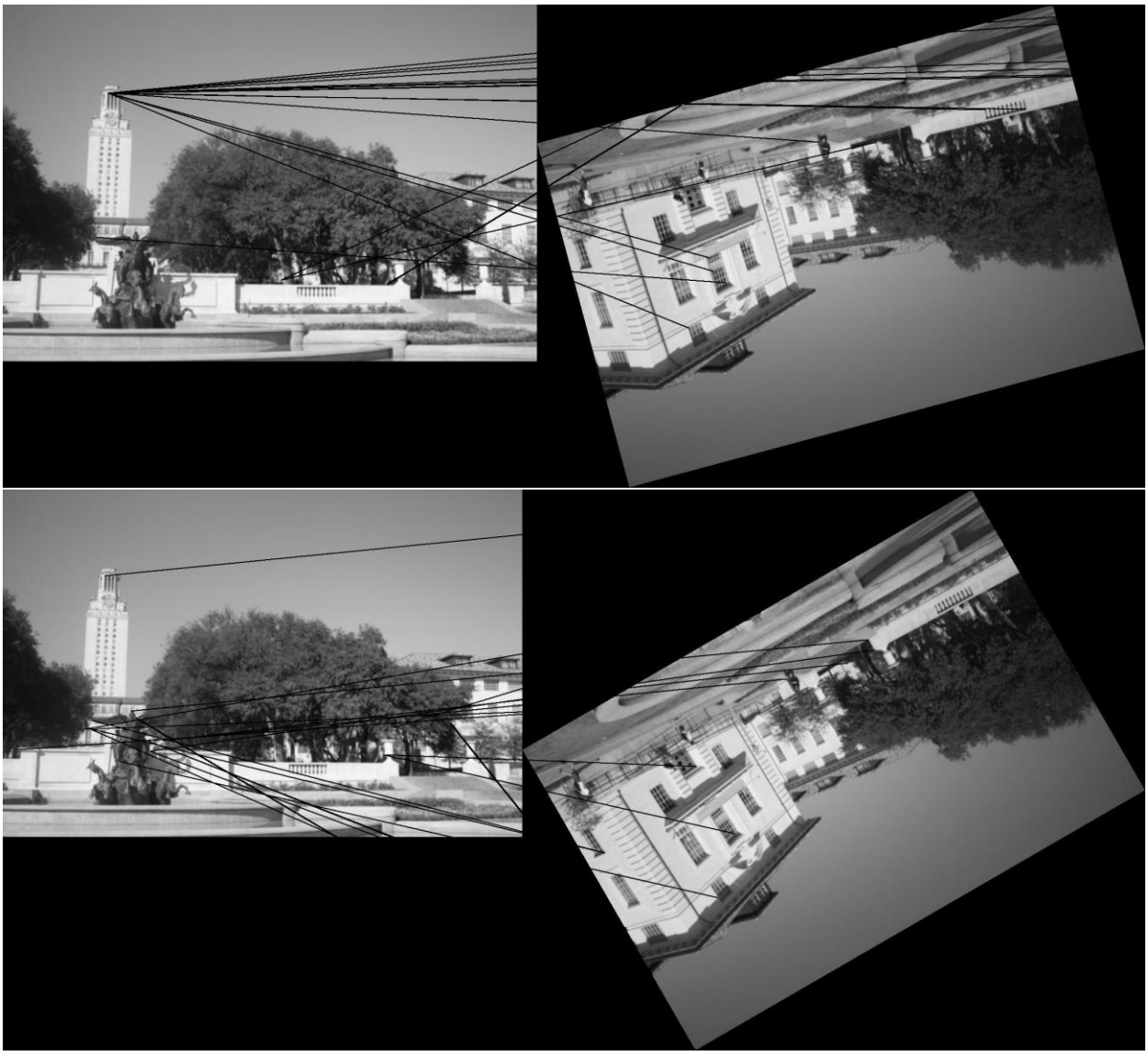
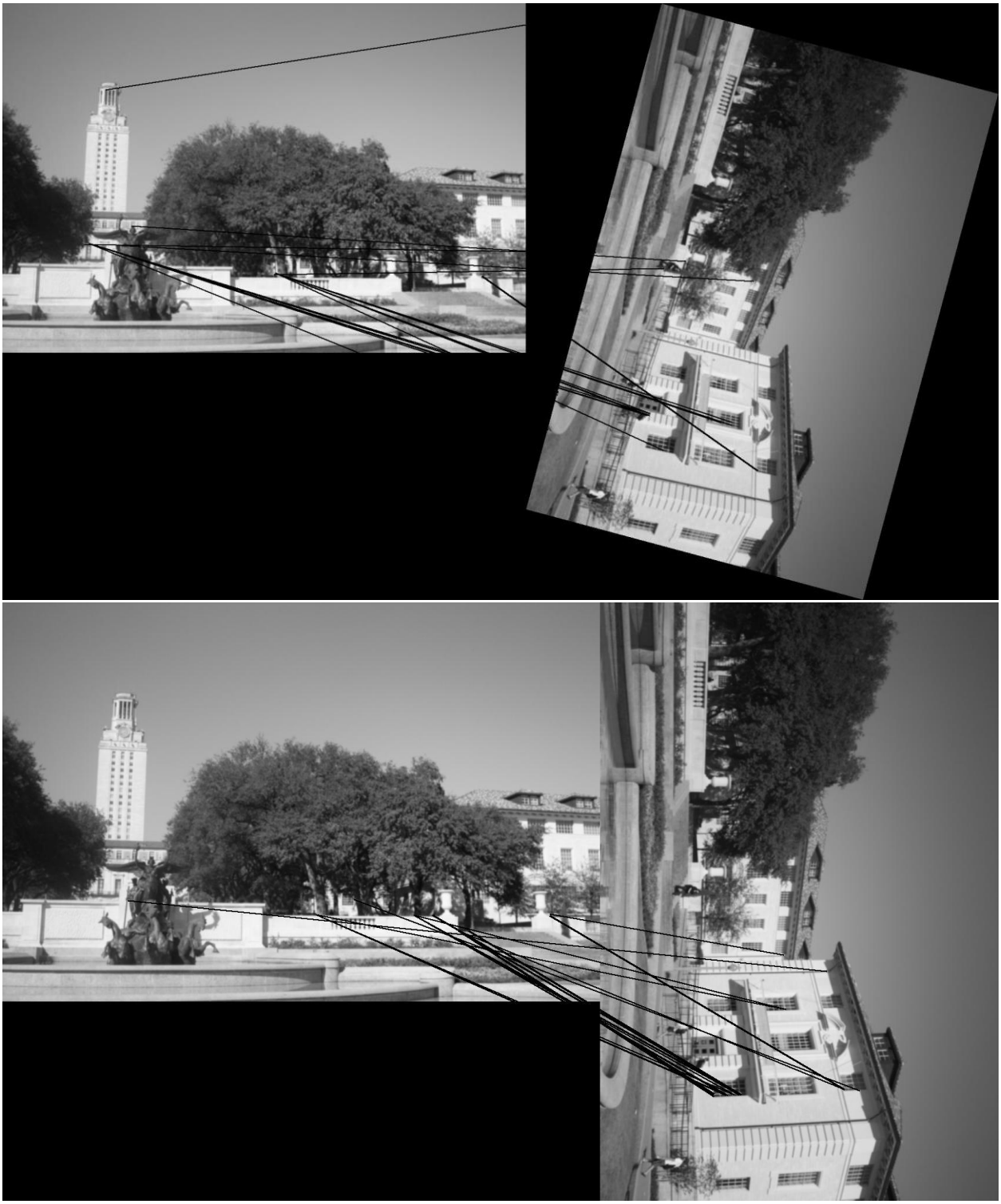
Aidan Fischer 11/4/2021

CS558 Homework 3 I pledge my honor that I have abided by the Stevens Honor System.

Results of NCC matching of features, normal.

Results of NCC matching of features when 2nd image is rotated in 15 degree increments from 15 to 345 degrees of rotation. Results discussed at end.





First image: Image stitching with RANSACed warping with a1 group, second is a2 group. Results discussed at end, however something obviously is not working properly here.

Results of Harris detector, first image of each pair is normal results of top 1000 features, 2nd is after non-maximum suppression.

1) Difference between NCC and SSD
    a. SSD is faster, but variance in local intensity levels will affect the final results
    b. NCC is slower, but invariant to local contrast and intensity levels.
2) Discussion of results of NCC matching
    a. I'm not entirely sure if I implemented something wrong or if the results were just not supposed to be good, but my high quality matches were not matching the correct points, but generally were matches between points that seemed close to matching but non perfect. I attempted to remedy this by increasing the patch size, but this just ended up with worse quality matches after a certain point. The best of the best matches did match the correct points, but the bad matches likely resulted in the bad affine results later in the project.
3) Discussion of results of rotated matching
    a. While the match quality of the normal matching already left something to be desired, rotations of the second image very quickly degraded the top matches even further. I believe this is due to how the matching is being done. I am unsure if NCC is supposed to be rotation invariant, but even if it was in terms of patches, the way this happens does not let that work. The patches are still taken in the same area, so pixels outside the patches of the first image are taken from the second image patch, due to the nature of the rotation, and thus the matches degrade. It is impossible to know beforehand if an

image is rotated in such a way unless that knowledge is given. A way to remedy this situation would be to match against a set of rotated patches, but this would be even slower than normal NCC, since it would require multiple matches per attempt, and rotation math as well as interpolation.

4) Discussion of RANSAC
    a. If I implemented RANSAC the proper method with removing of inliers, then I would expect somewhere closer to 200 iterations. However, the bad matches combined with my general confusion with regards to implementing RANSAC led to me just running 50000 iterations and hoping it gets a good affine matrix.
5) Discussion of results of warping
    a. Obviously this did not go right, but I really am not sure where it messes up. It is definitely related to the overall bad quality of matches. The a1 group match at least ends up on the first image, but a2, due to taking 30 random matches from a set of horrible matches, just does not work at all.
6) Discussion of implementation
    a. For the first step, I implemented the harris detector as explained in the slides and parameterized as given in the homework instructions (top 1000, NMS). I then implemented NCC to every pair of features between image 1 and 2, and created a list of tuples containing the signature of each feature in the match, and the result of NCC on patches centered on those features from each image. I then sorted that list by the NCC result and took a1 and a2 from that list. From that I ran RANSAC on a1 and a1+a2. RANSAC used a random sample of 3 matches, running a matrix equation solver on a solution equation given in the slides that defines the values for an affine. Since I couldn't be entirely sure there would be a solution, I used the least squares solver from numpy's linear algebra module, which takes the solution with the least squared error. Inliers were determined by the distance from an expected feature location from the original feature in image 2 from the calculated location when the estimated affine matrix is applied to the location of the feature in image 1. Average error is the sum of the distances for all matches in the considered group divided by the size of the group. Better matches are determined by having average error be the lowest encountered so far. Using the returned affine warps, image 2 is warped then the images are then stitched together. To stitch, I create an image array large enough to hold the stitched image, then I create two more arrays of the same size. I place image 1 in one, and the warped image 2 in the other, positioned such that they are where they will be in the final image. I then loop over the pixels, and determine the proper value at each pixel in the final image. In cases where both the positioned image 1 and 2 have a pixel value, I average the pixel colors by taking the square root of the average of the color elements squared for each color element in the pixel (e.x. r' = sqrt((r1^2 + r2^2)/2). Throughout, I output interim images to produce the requested result images. The rotation results were a one time run, and thus the code producing them is not present in the final code.