Rongzhong Li

Project 3

To run the program, type **stitch('l.jpg', 'm.jpg')**, or alternatively **stitch('Reynolda_leftMid.jpg', 'Reynolda_mid.jpg')**.

My function requires two input images taken from different perspectives. For efficiency, I down sampled the images by a factor of 2. I used the red channel of the colored image for analyzing so it can be extended to other channels easily. (Though the transform can be acquired using only one channel and is enough for stitching the whole image.)

First step is to get the feature of the two images. I use the corner detection function to find the corners in image f and g. Then pass their coordinates to the function find_sift. The function will create an N×128 matrix as the descriptor of each image, while N corresponds to the number of feature points.

Then I make an M×N table of the distance between descriptors of each image. I'm simply using the Euclidean distance between the two vectors. Among the distances, I pick out those distances smaller than a threshold and record their indexes. The index helps to find the paired coordinates in image f and g.

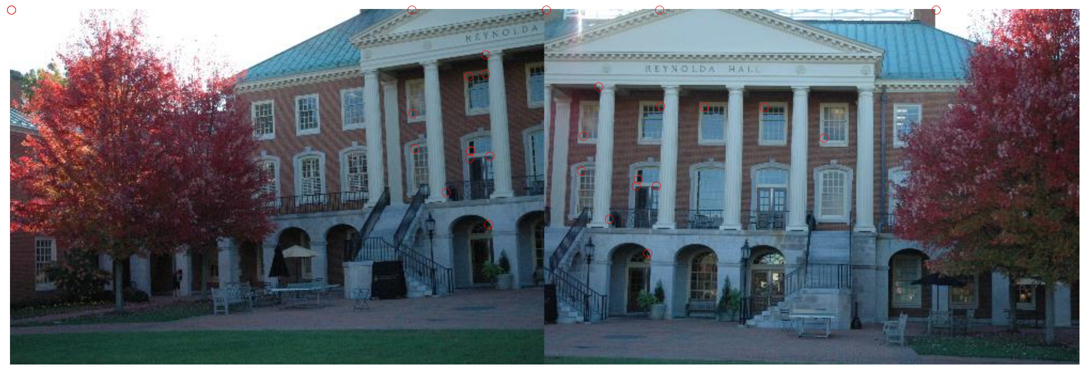
After getting the possible corresponding coordinates in two images, I need to find their transformation matrix. This is done by RANSAC fitting. I randomly pick 4 pairs of f and g coordinates. Write the entries in the formula Ah= 0. Use SVD decomposition of A to get the solution of h. With the h, I can predict all the corresponding points of f in g. By calculating the distance between predicted points and real points, I get an error value. The errors smaller than a threshold are counted as inliers. If the ratio of inliers over the total points is smaller than a certain threshold, the transform h is considered correct.

With the correctly paired coordinates, we know the transform between the two images so we can stitch them together.

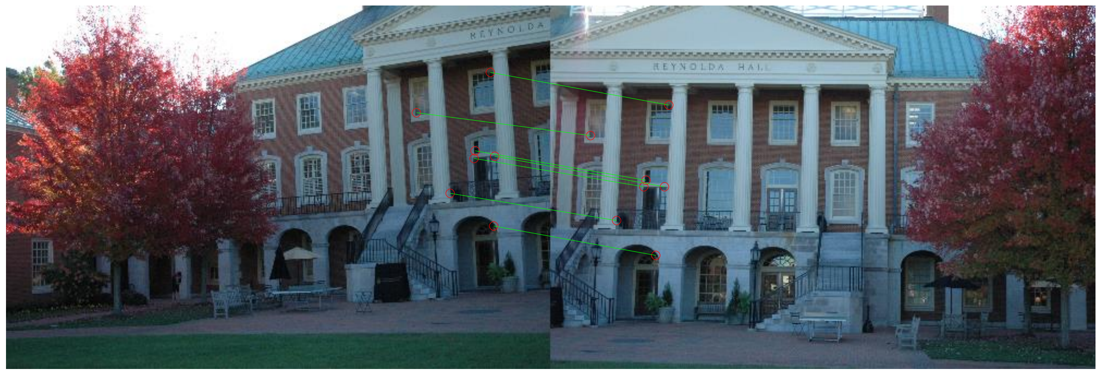
There are five major parameters to consider.
corner_threshold tunes the detected corners in each image. Increased numbers of corners improves the accuracy of the algorithm, but also increase the amount of calculation. I set it to be 0.005. It returns about 600 corners in each image.
enlarge_factor tunes the region of the feature area. Larger factor will consider a larger neighbor area. It makes the feature more specific. I chose the factor to be 2.
dist_threshold tunes the number of pairs that can be considered matching. A large threshold may produce false matching. I plot the histogram of the distance distribution and set it to be 0.3. So I get about 15 paired coordinates in f and g. You may think it to be small, but I'll explain later.
err_threshold tunes the number of inliers in RANSAC algorithm. If the transform is correct, we should be able to find lots of errors close to 0. So I'm using 1;
inlier_threshold tells the RANSAC when to stop. Though the standard RANSAC should define certain converging values like 0.9 or so, I find the situation is different in this problem. Because when stitching two images, we don't know what portion of the images are overlapping. If the two images share a lot in common, the threshold should be high. If the two images only overlap

at their right or left borders, the threshold should be low. Thus the threshold of inliers is hard to tell. On the other hand, 4 points can determine one transform. And the transform should apply to the other coordinates if it's correct. This is a quite strong constraint. If a transform works for several points other than the 4 points used for derivation, it should be a good one. So instead of using a threshold ratio, which is N_inliers/N_total, I use the condition N_inliers>=7. To exclude the false matching situations, I make the dist_threshold very strict so the pairs are few but separate and accurate. The result shows the philosophy "*Calculate by 4 and validate by 7*" works.

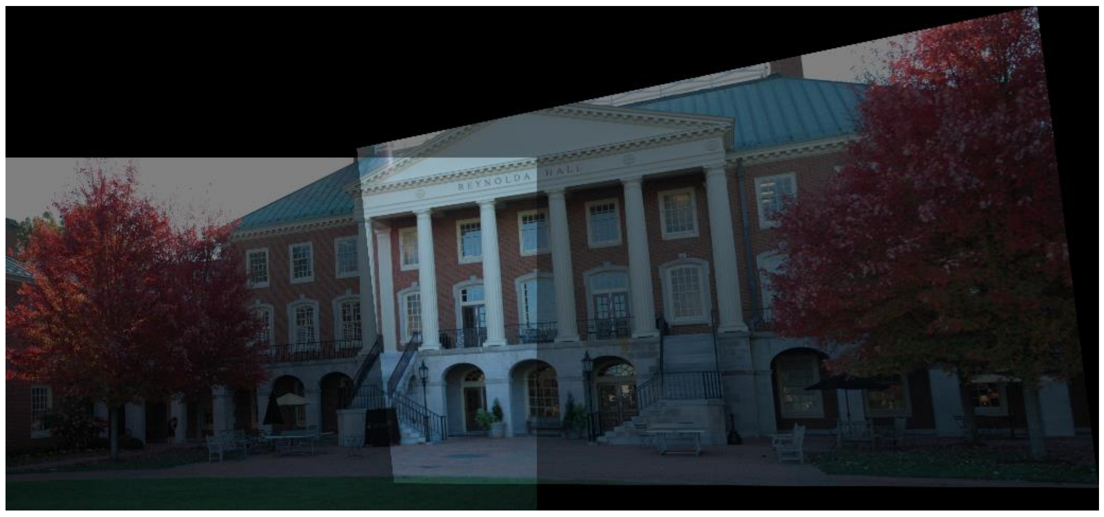
The red circles shows the feature points after filtered by distances.



Below shows the matching points in two images.

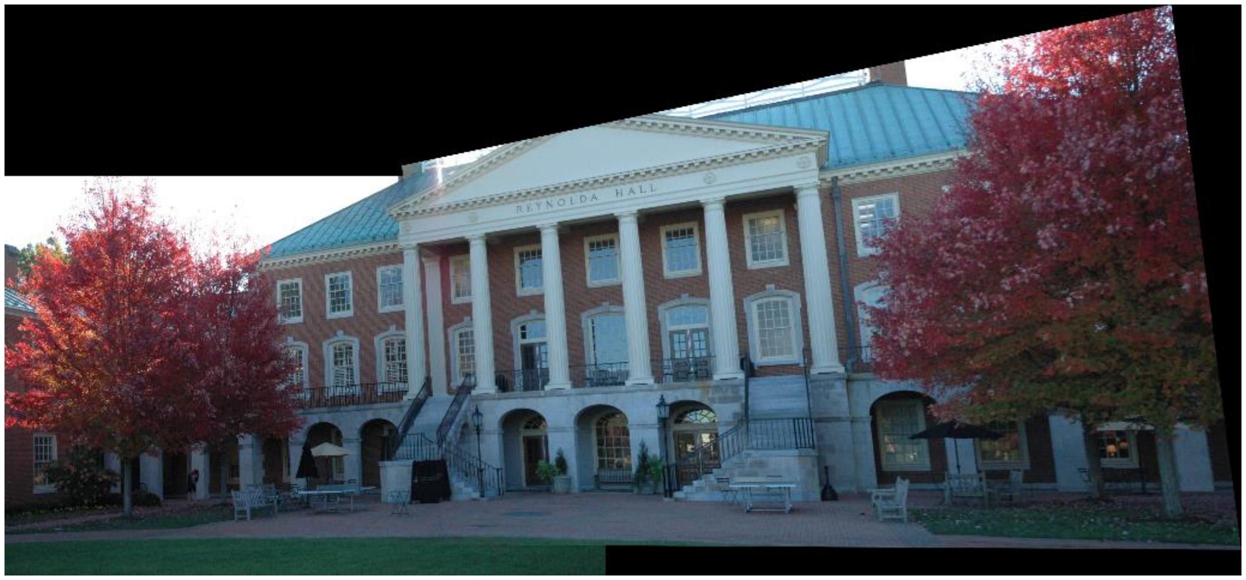


And here's the merged image from my stitching function.



Rongzhong Li

A better maximum merging method gives a decent result as below:



Below is another example of the stitch output. Note parameters have to be adjusted if we want better results.