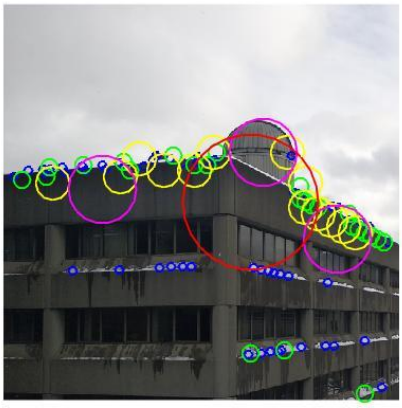
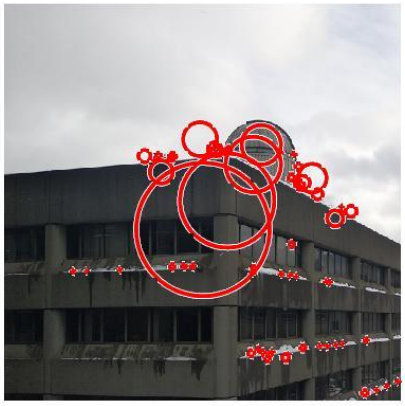
COMP558 A3
Yiran Wang
260825557
In this Assignment I will use trottier data set(the horizontal one).
1.





.

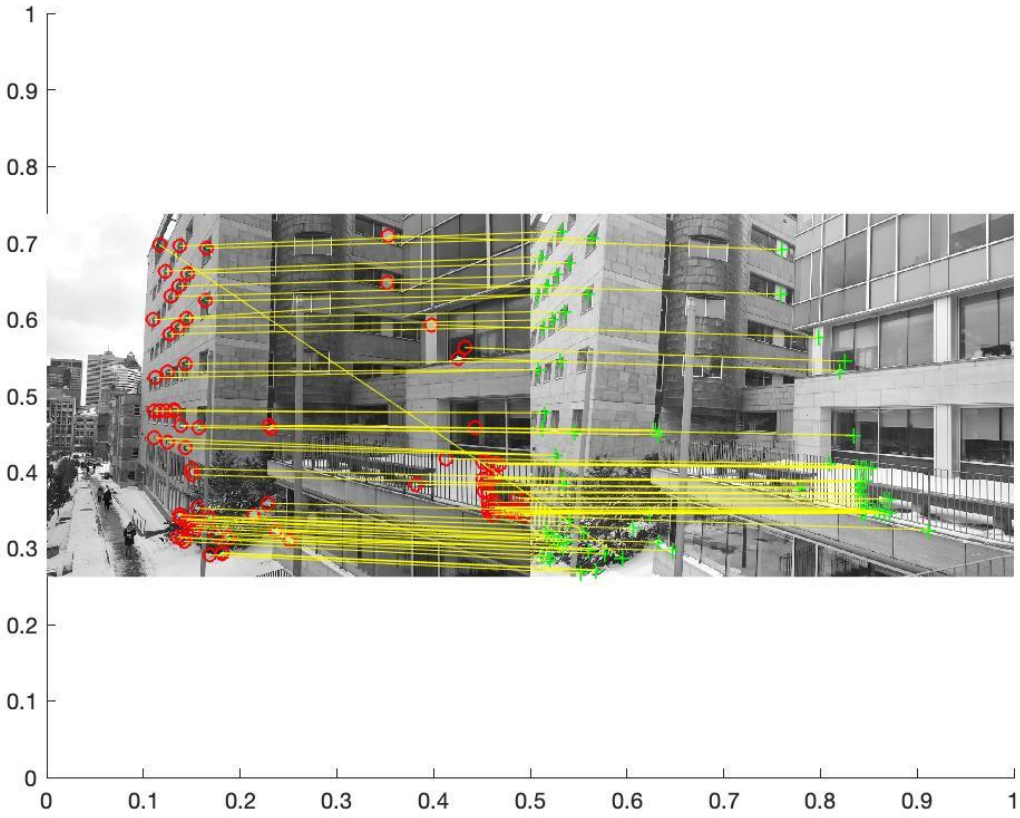
left is from my implementation, right is from sample implementation

Obviously there are much more keypoints detected in my implementation. But they are noisy because most of them lay on edges. According to Lowe's paper, we are supposed to eliminate edge response, but we did not implement that part in A2. Also, by using sample code, we can have multi-level in each octave, which is not implemented in A2 either.
To sum up, the difference between these two results is possibly from:

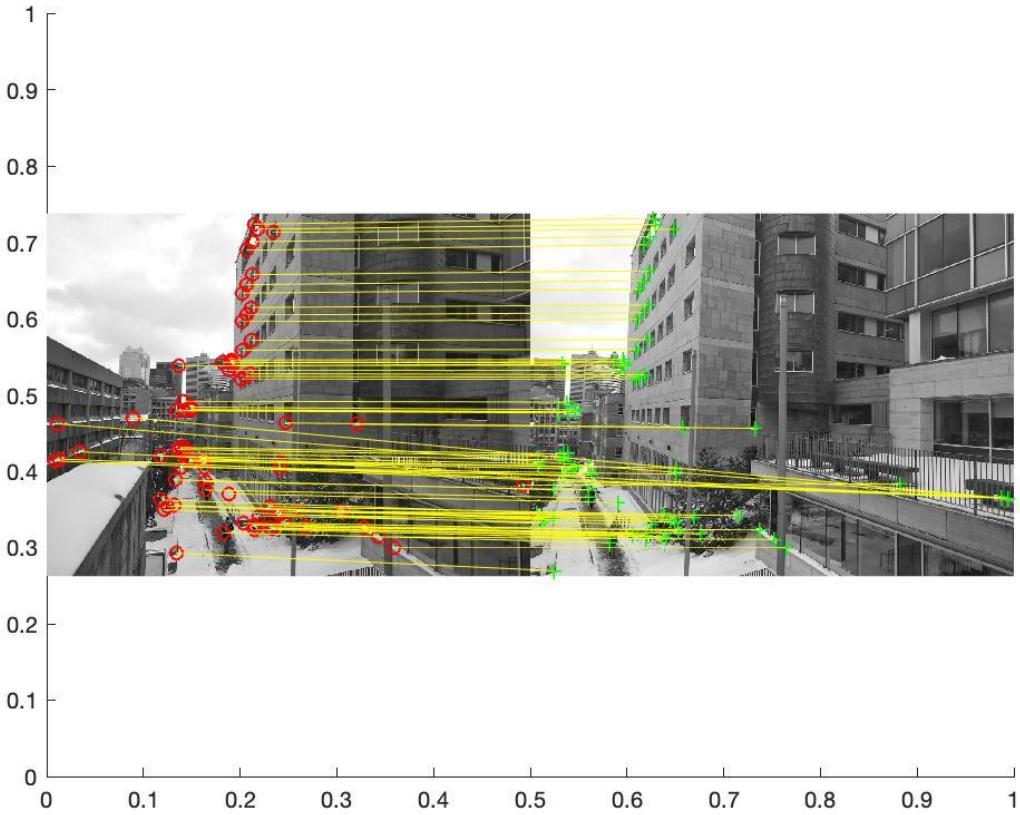
1. We did not eliminate edge response
2. We only have one level in each octave(I used 4 levels per octave in the sample implementation)

To reproduce this result, please run question1.m. No helper function is needed.

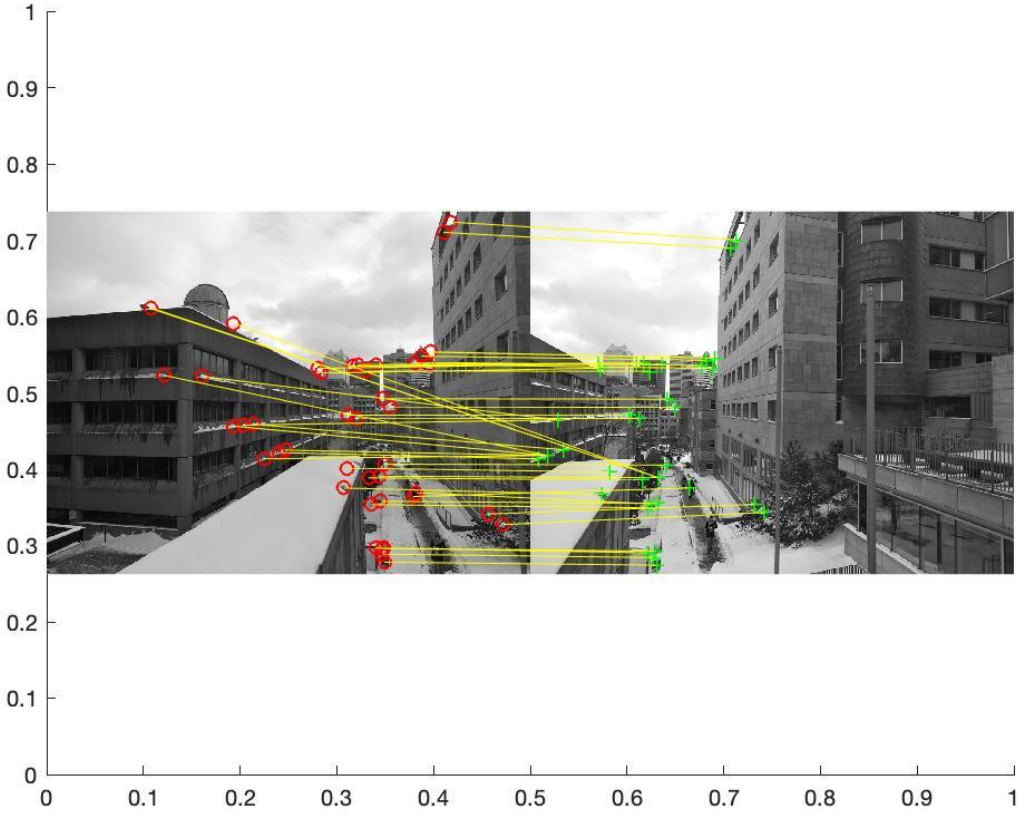
2.



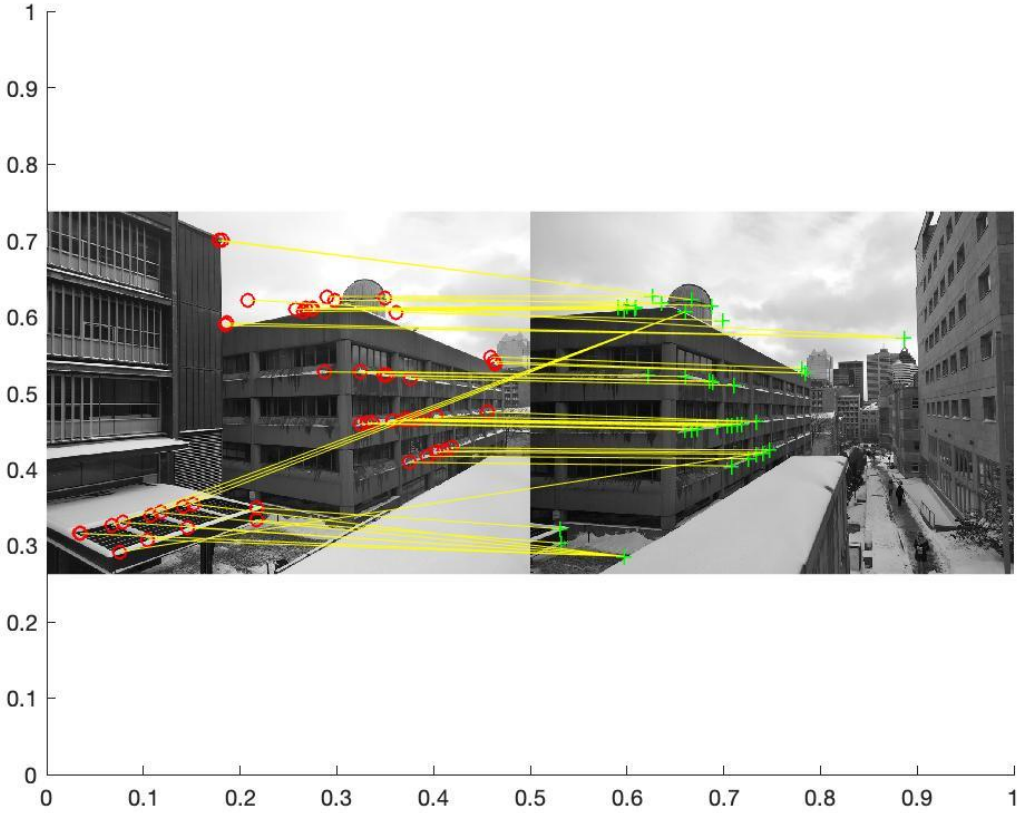
Matched features between trottier1 and 0



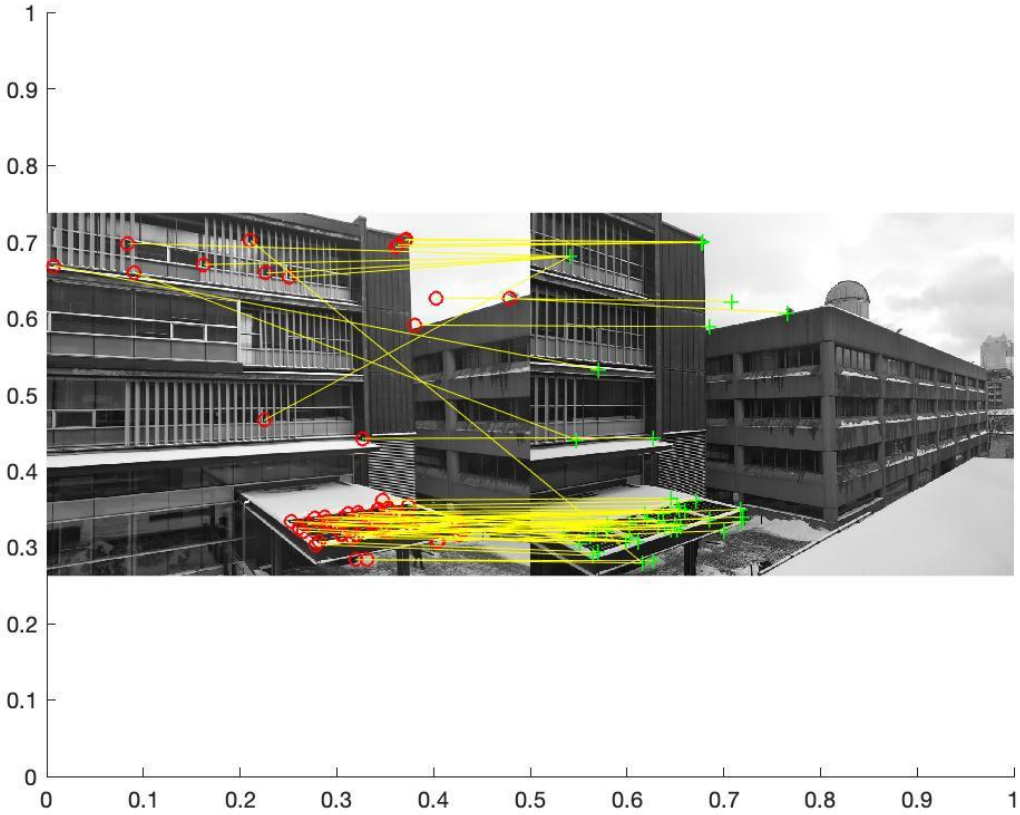
Matched features between trottier2 and 1



Matched features between trottier3 and 2



Matched features between trottier4 and 3



Matched features between trottier5 and 4

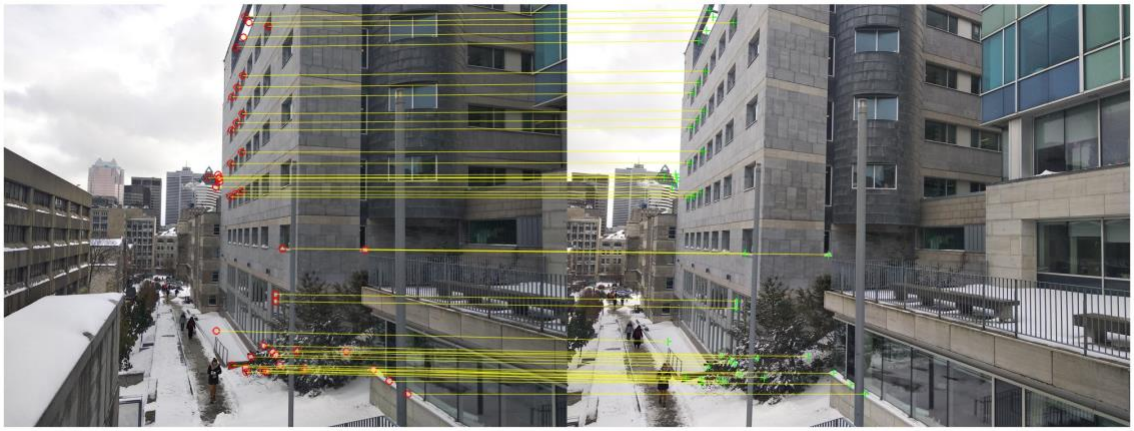
I use the same hyper-parameter as question1 to find key-points in each image. I store each point and its feature(128-dimensional vector), then I use MATLAB built in function matchFeatures with match threshold 10% (from the perfect matching). The matching metric is sum of squared differences between features.

And obviously the matches are not ideal, on the contrast they are very noisy. Abundance of mis-matches appear. I will improve the out-come in the next part by using RANSAC.

To see these results, please run question2script.m.
The helper function file to implement this is compare.m.

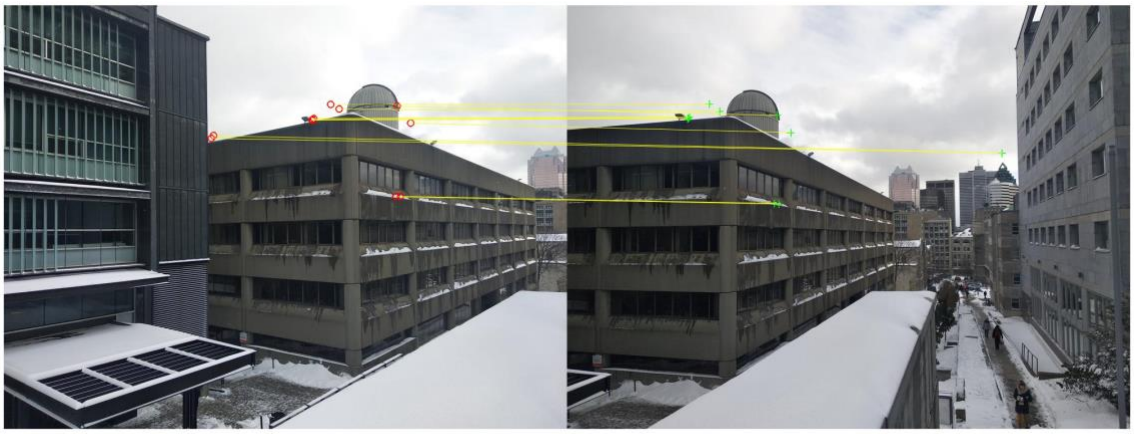
3.

Ransac of trot1 and 0



Ransac of trot2 and 1

Ransac of trot3 and 2



Ransac of trot4 and 3

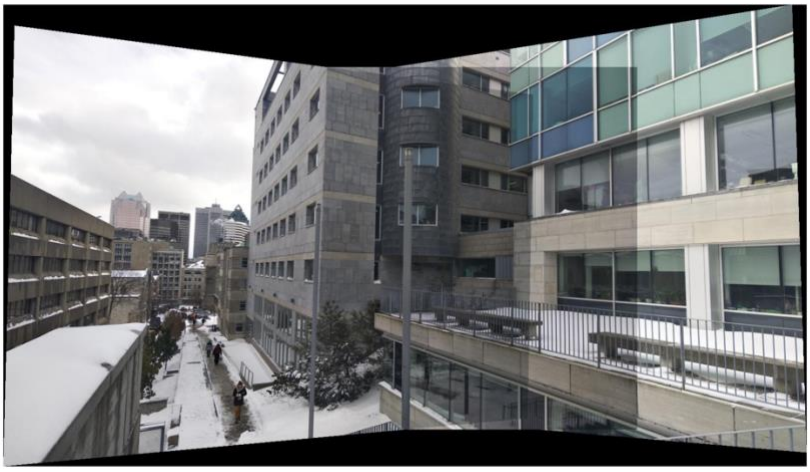
Ransac of trot5 and 4

Approach:(for two images next to each other)
Firstly I randomly choose 4 matches from all matches, which means we get 4 key-points in each image. Then we use these 8 points to calculate homography matrix H.
After iterating 500 times, we choose the best H that has most inliers(matches) and plot them, as showed above.
The criteria of deciding if a match is an inlier is that for a match (p1, p2) and a homography matrix H, we calculate p2' = H * p1, and then we check if p2' is close to p2 enough in terms of Euclidean distance as showed in lecture notes. If p2' and p2 is close enough, then we say that (p1, p2) is an inlier.
Then for the H that has the largest consensus set(i.e. most inliers), I use its inliers to construct a 2N*9 matrix, and use least square method to calculate the final homography matrix H_Best.

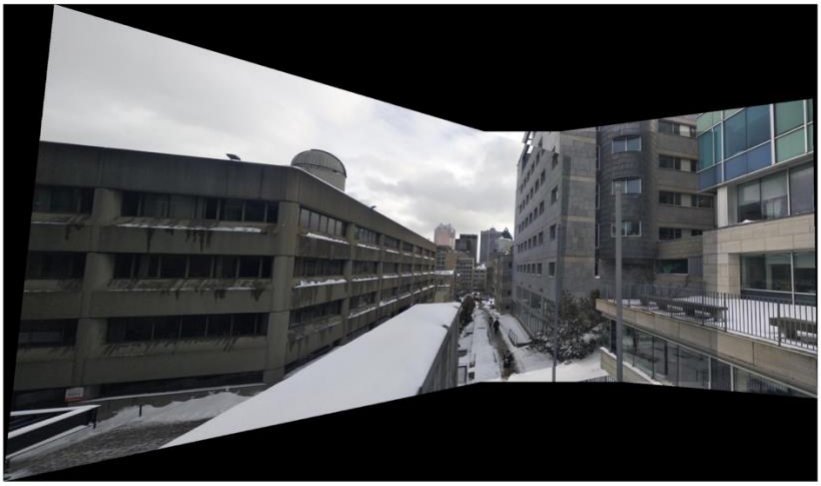
It is obvious to see that after using RANSAC, there are much less mismatches. However, we also get much less matches. But in order to get the accurate homography matrix, it is acceptable.

To see these results, please run question3script.m
The helper function for this part is showRANSAC.m

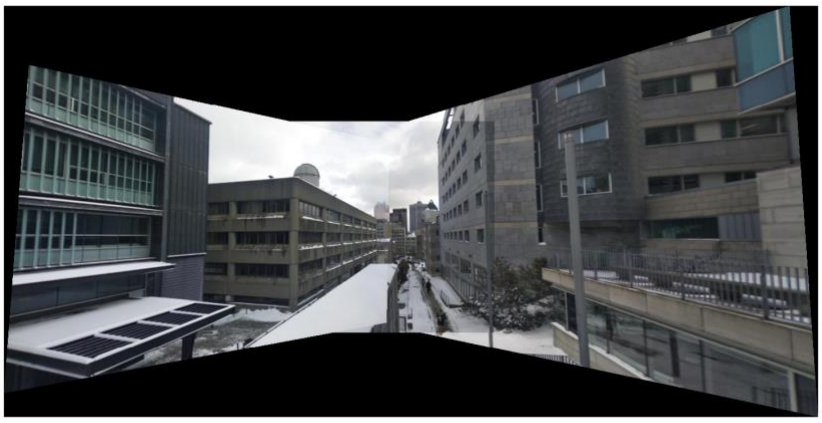
4.



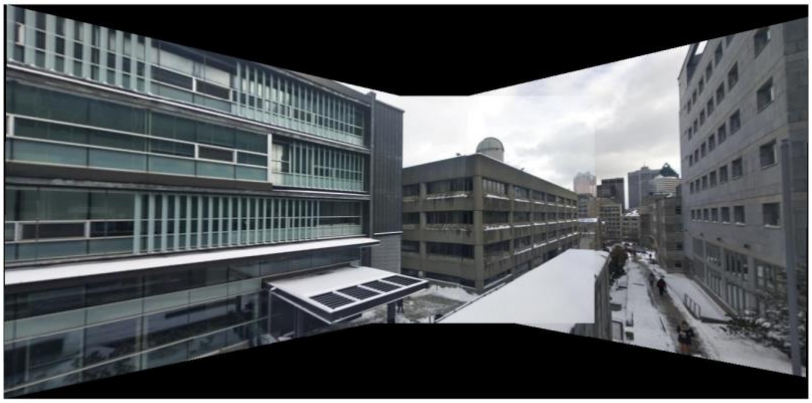
Panorama of trot0, 1, 2



Panorama of trot1, 2, 3



Panorama of trot2, 3, 4



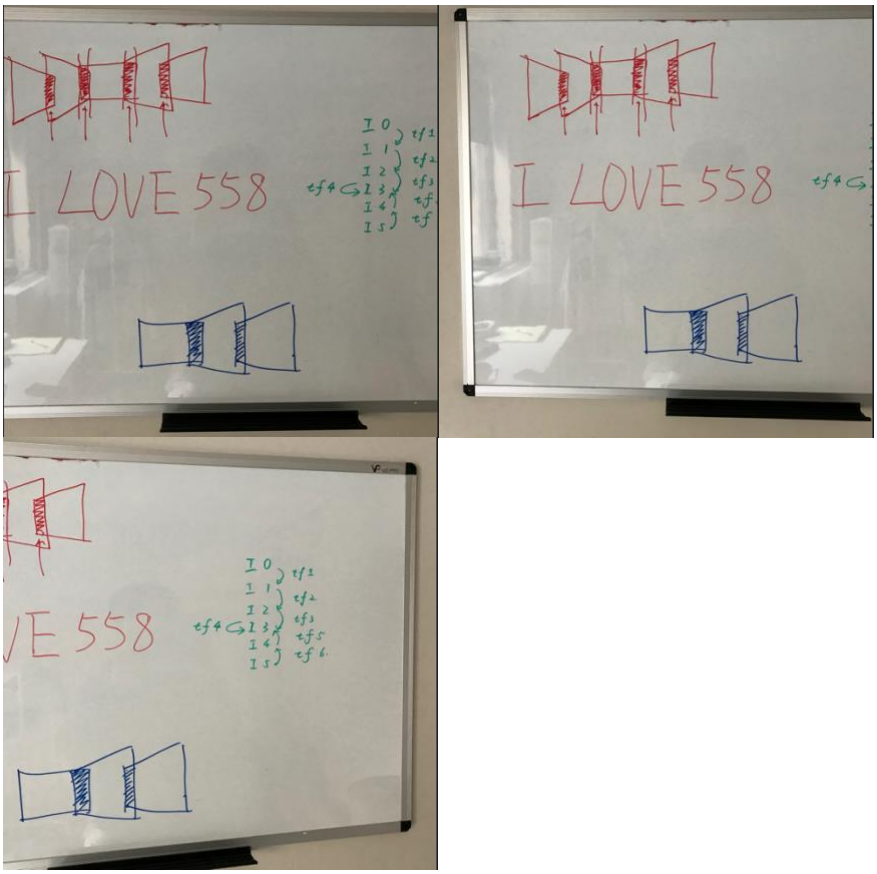
Panorama of trot3, 4, 5

When using more than 3 images, they will be twisted terribly, so I only show these fine panoramas here.
Approach:

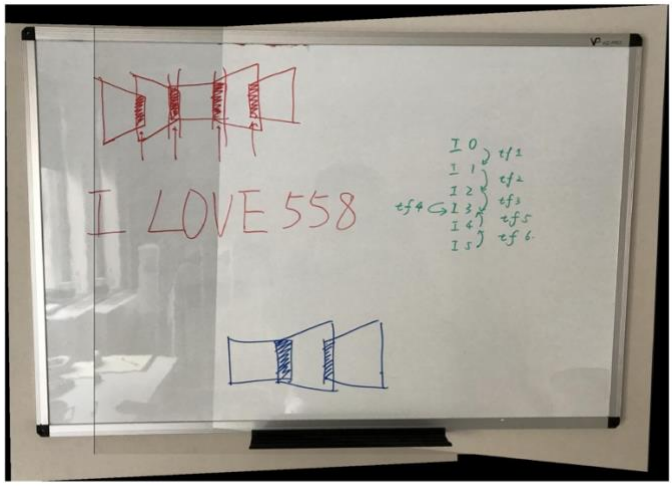
1. Calculate homographies between each two images. For example we have I1, I2, I3, then we need to calculate 2 homographies: H(1->2), H(2->3), and then use H's to transform corresponding images.
2. Then we predict what is the possible panorama size, and create an empty panorama.
3. Then we put transformed images into the empty panorama with alpha blender.
4. Then we get the result shown above.

To see the results, please run question4and5script.m
The helper function for this part is RANSAC.m, showStitching.m

5.



Stitching result:



6.

Reference: Automatic Panoramic Image Stitching using Invariant Features. By Matthew Brown and David G. Lowe

In this paper, an approach is introduced to reduce the ghost edge effect.

The method is called gain compensation. The main idea is to eliminate intensity differences.

To implement this, we need to first find the overlap region from two aligned images and calculate the average intensity of the overlapped region of the two aligned images respectively. Then we calculate the difference of the average intensity of the overlapped region.
After that we add the difference to one of the image(on every pixel), then we are done.