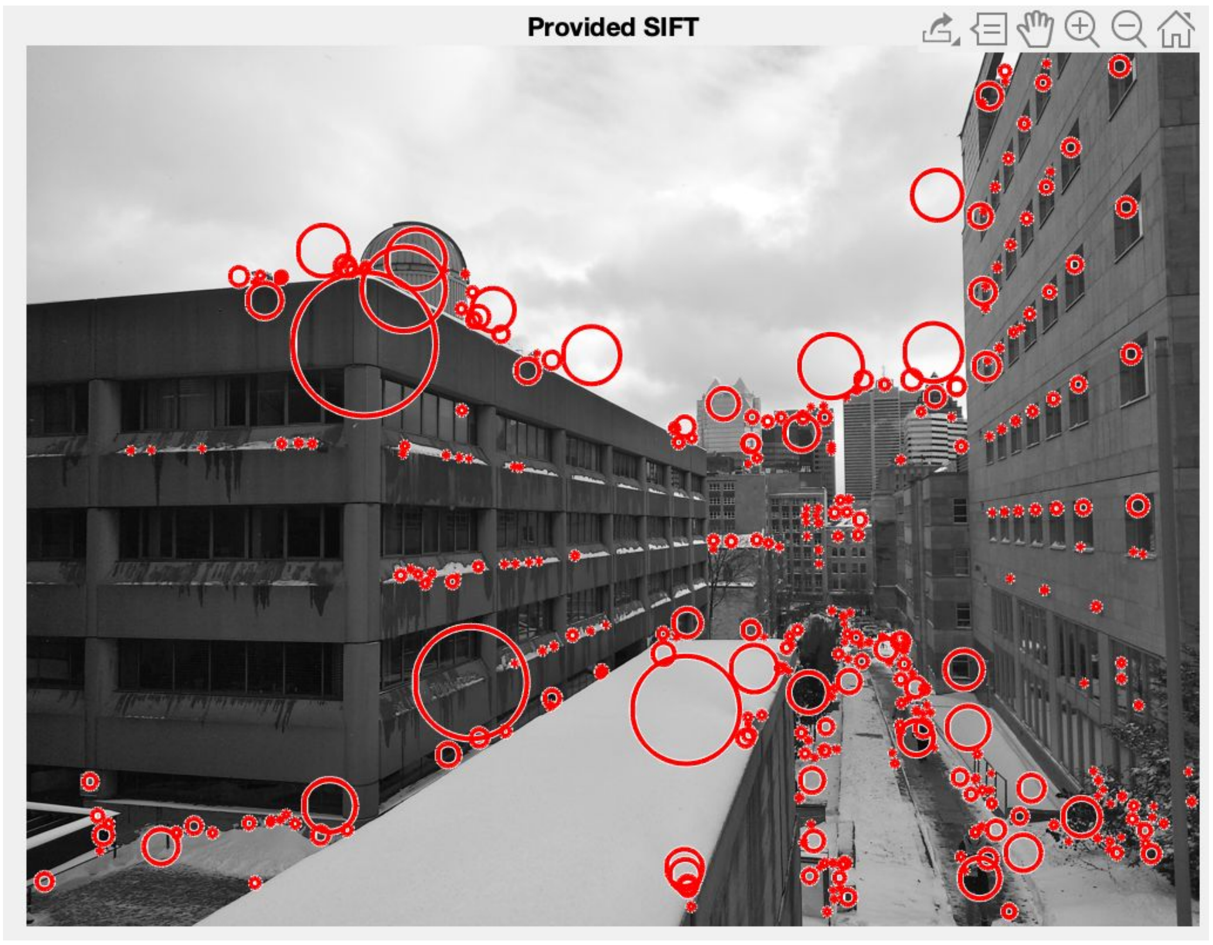
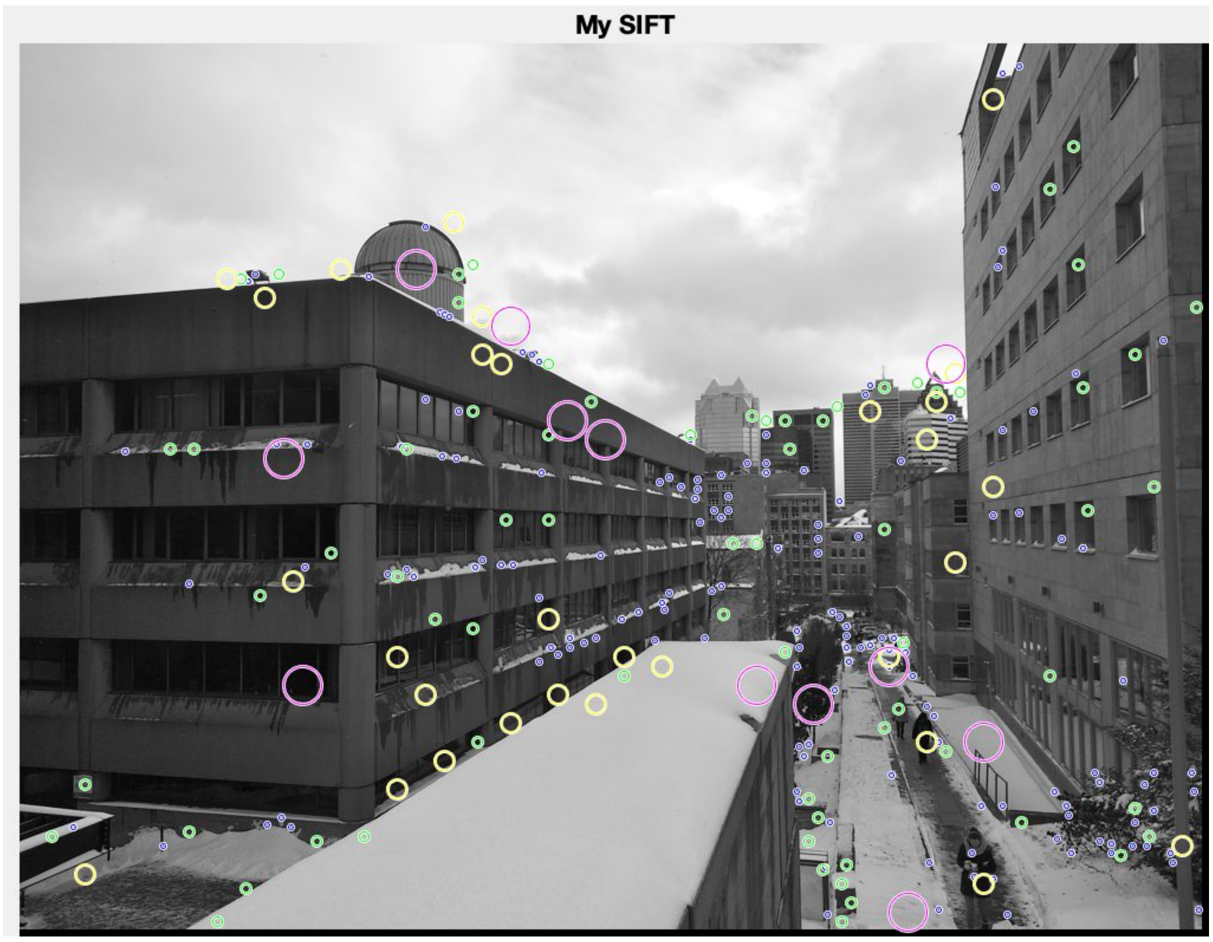
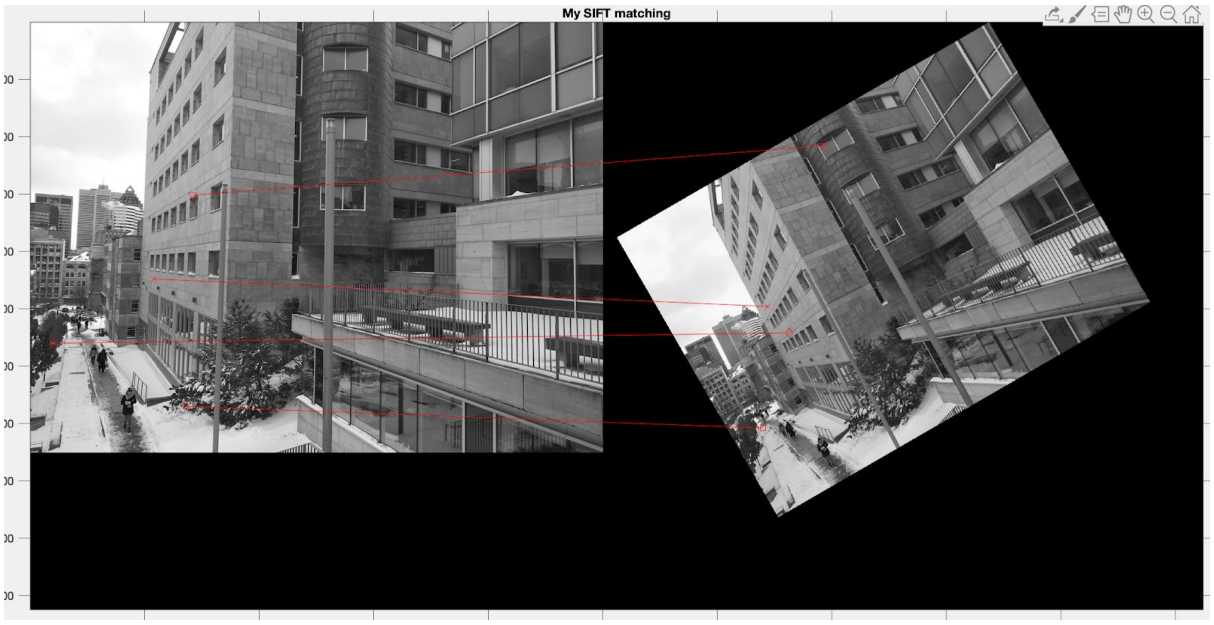
q1.

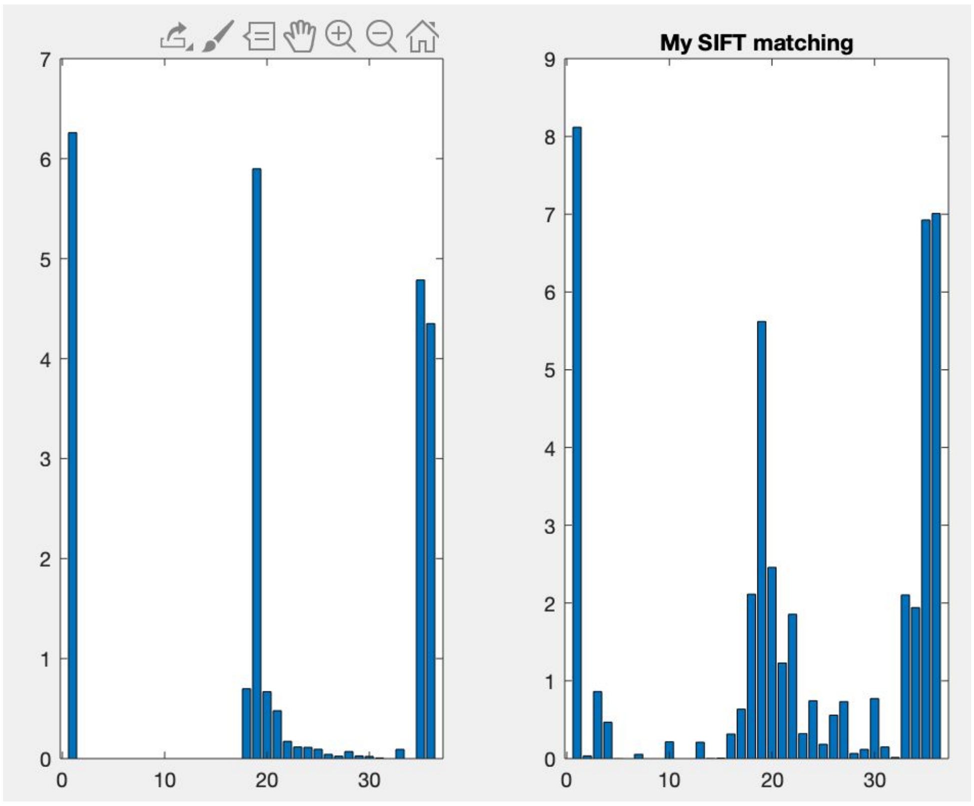




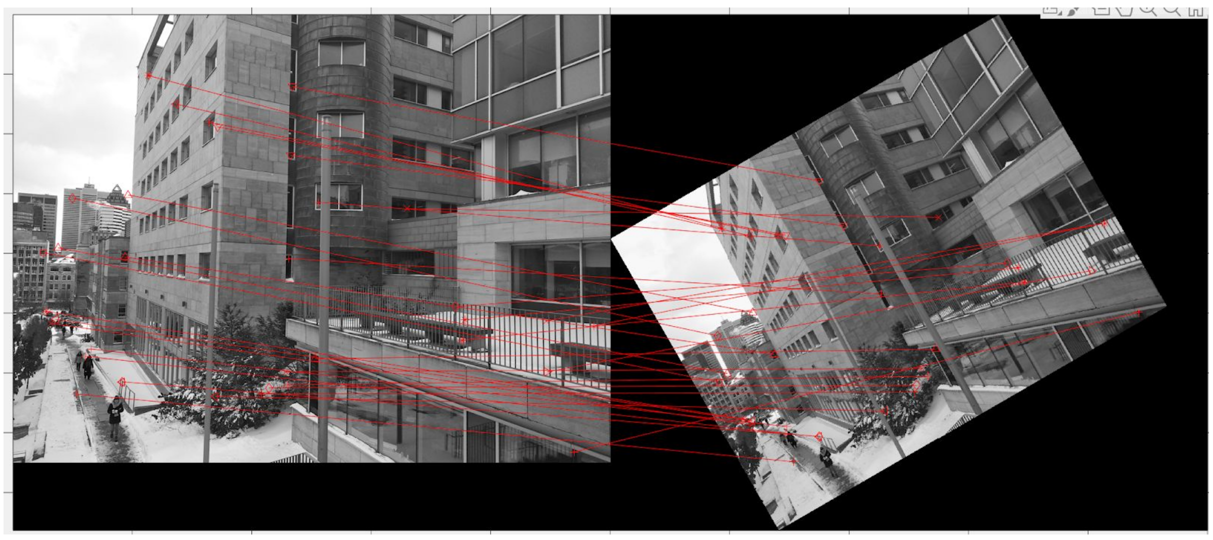
By using my sift to find the match after the rotation and resize (just several keypoints shown):



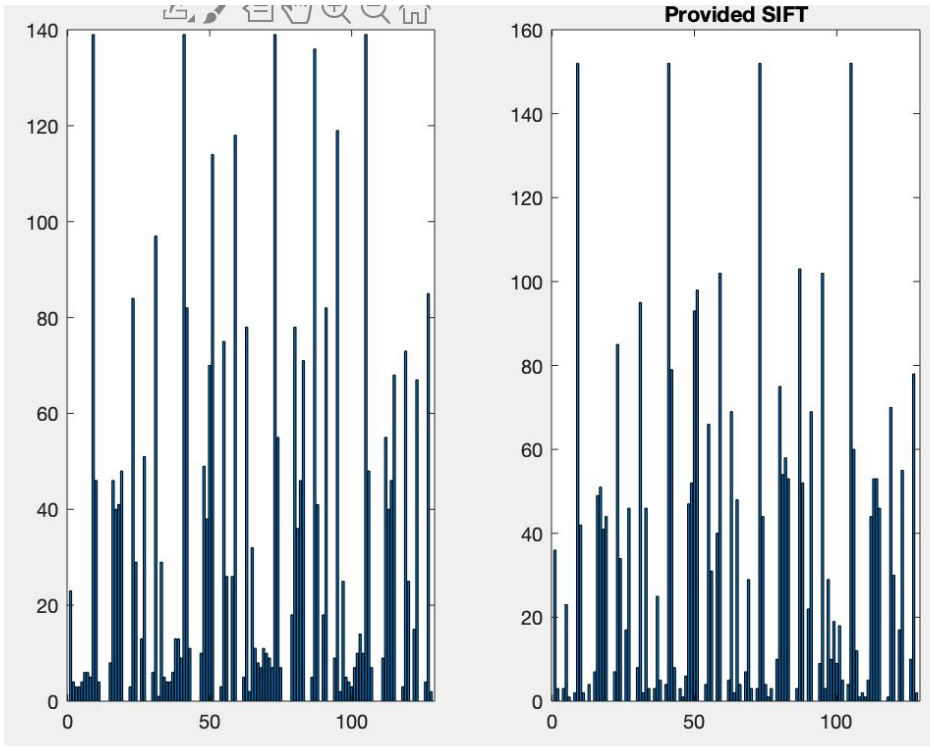
And the histgram of two matched points:



By using **provided sift** to find the match after the rotation and resize:



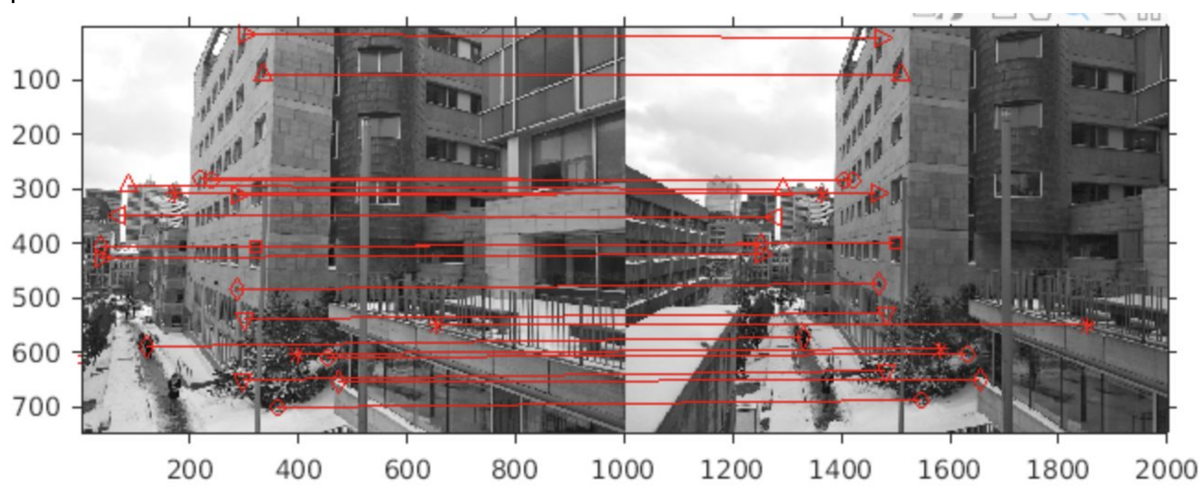
Histogram illustration of a matching pair:



How and why are they different?

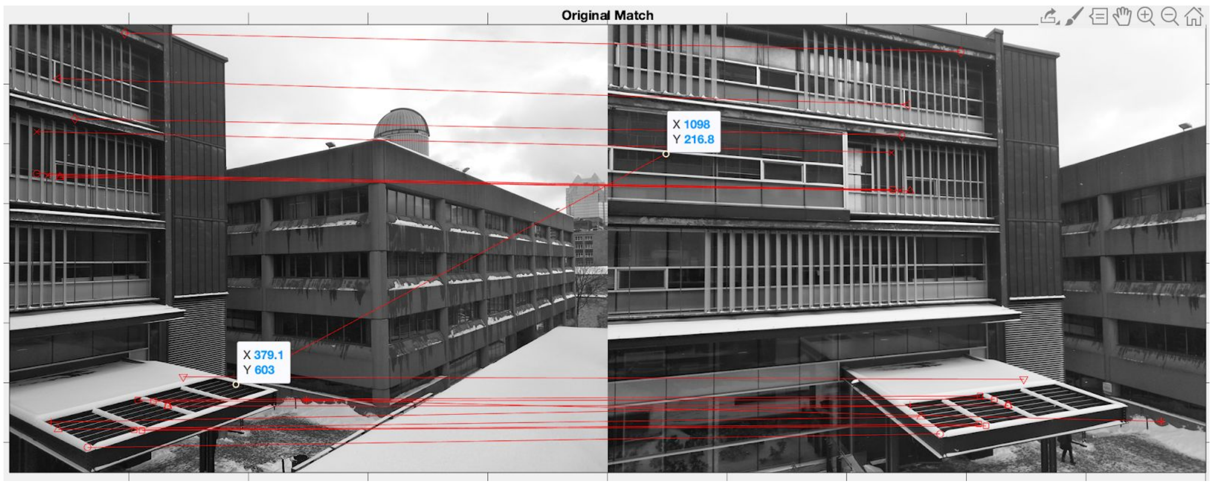
Overall, the provided SIFT feature function is much more robust than the one I implemented as you can see that, by using the provided sift features, almost every matching is precise. The reason I think should be illustrated as follow. First, in the provided SIFT calculation, we can have multiple octaves, while we don't have multiple octaves in the SIFT features finding in assignment1. Second, for each sift features in the provided SIFT, we have 128 elements, while in my own sift feature, we have only 36 elements. So the provided sift features can be finer than my sift features.

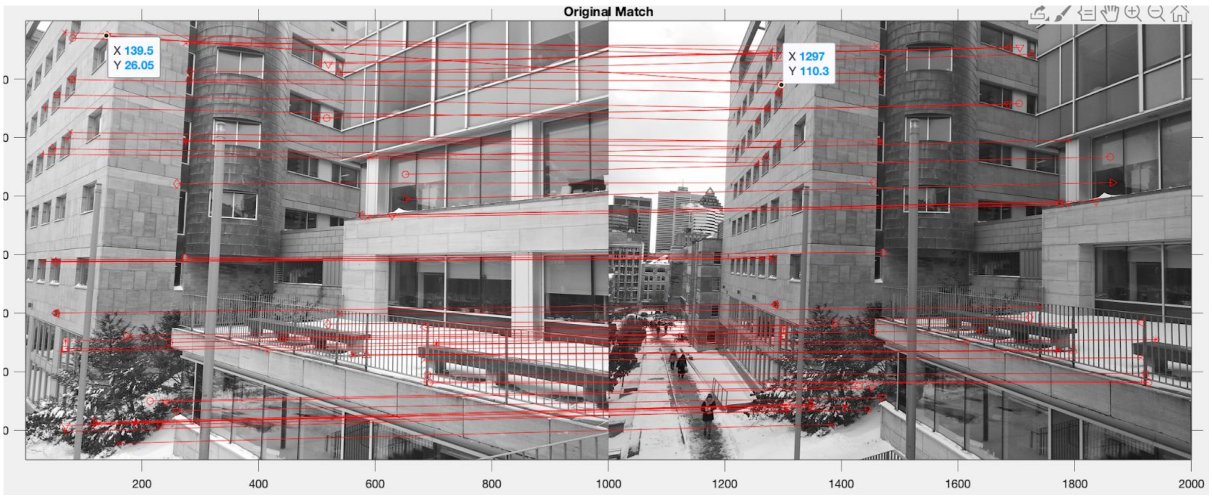
q2.



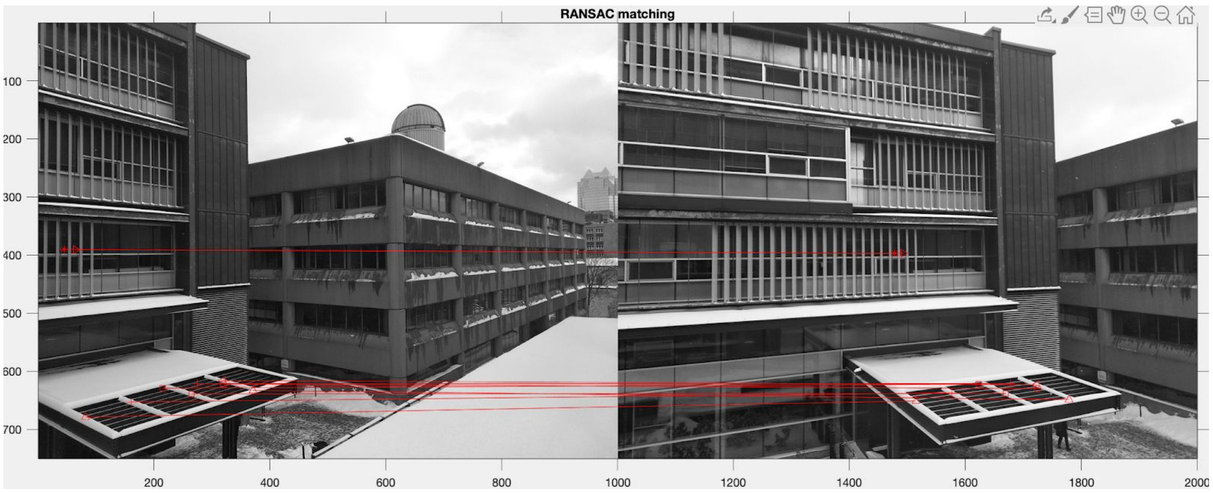
The figure above shows the feature matching by using provided SIFT function. For the matching process, I use Matlab's build-in function *matchFeatures*. The '`Exhaustive`' method

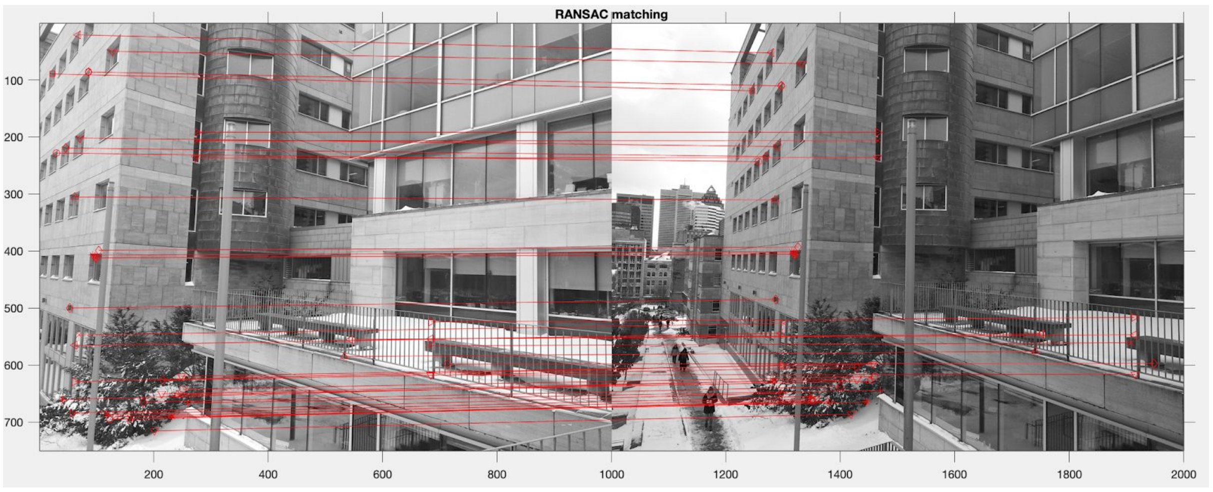
is used, which Compute the pairwise distance between feature vectors in features1 and features2.
q3.
Detail of steps of my RANSAC algorithm:
Based on the point pairs I have found, I will randomly choose 4 pairs to construct the matrix A. Then use *null(A)* to find a current H. Then I will go over each point pairs and match one pixel point to another pixel coordinate. If the euclidian distance between the transformed pixel and the golden-truth pixel is below a threshold, I will add the pair to the consensus set. After loop over all the pairs, if the number of current consensus set is larger than the best one, I will use the current consensus set as the best one.
The above process will be looped for enough times.
After I have found the best consensus set, I will use this set and the least square(LS) method to find the best H for these two consecutive images. For LS algorithm, a matrix K will be built based on the consensus set. Then I computer matrix AAT = K^T*K. Find the eigenvector H of AAT with the smallest eigenvalue. This H is the homography for the two images.





The figures shown above is the one using only feature matching function (I just show some of the key points by random selection for clarity). The point pairs highlighted is mismatching ones.



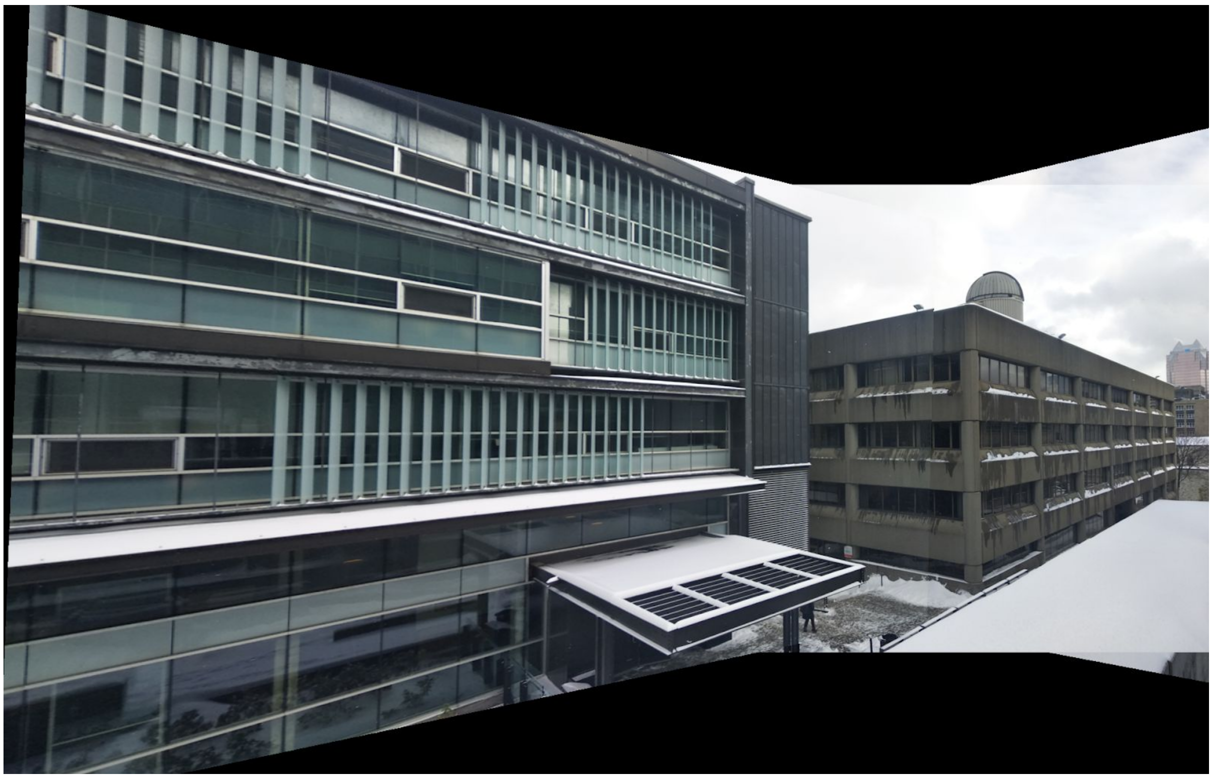


The figures shown above are pairs from the consensus set. You can see that the pairs are less than original matching because we eliminate some outliers when building our consensus set.

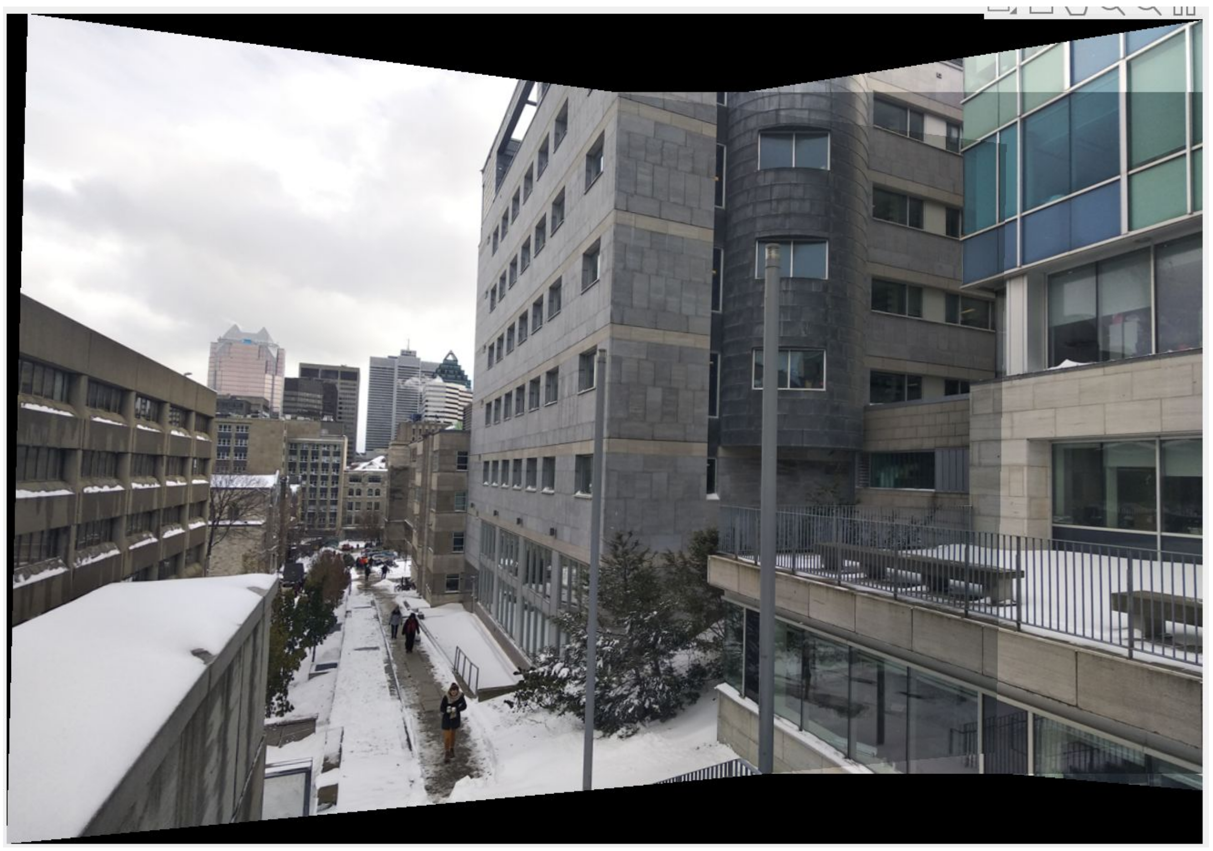
q4.

For this question, because of the memory limitation of my computer, I will only use 3 images to create a panorama. Based on the result obtained from q3, we can compute the homography between any two consecutive images. Then I use the second image (the middle one) as a fixed image and the other two images are mapped to the middle image's pixel coordinate. Finally, the two transformed images are concatenated to the middle image to create a panorama.

by using horizontal images 3, 4, 5:

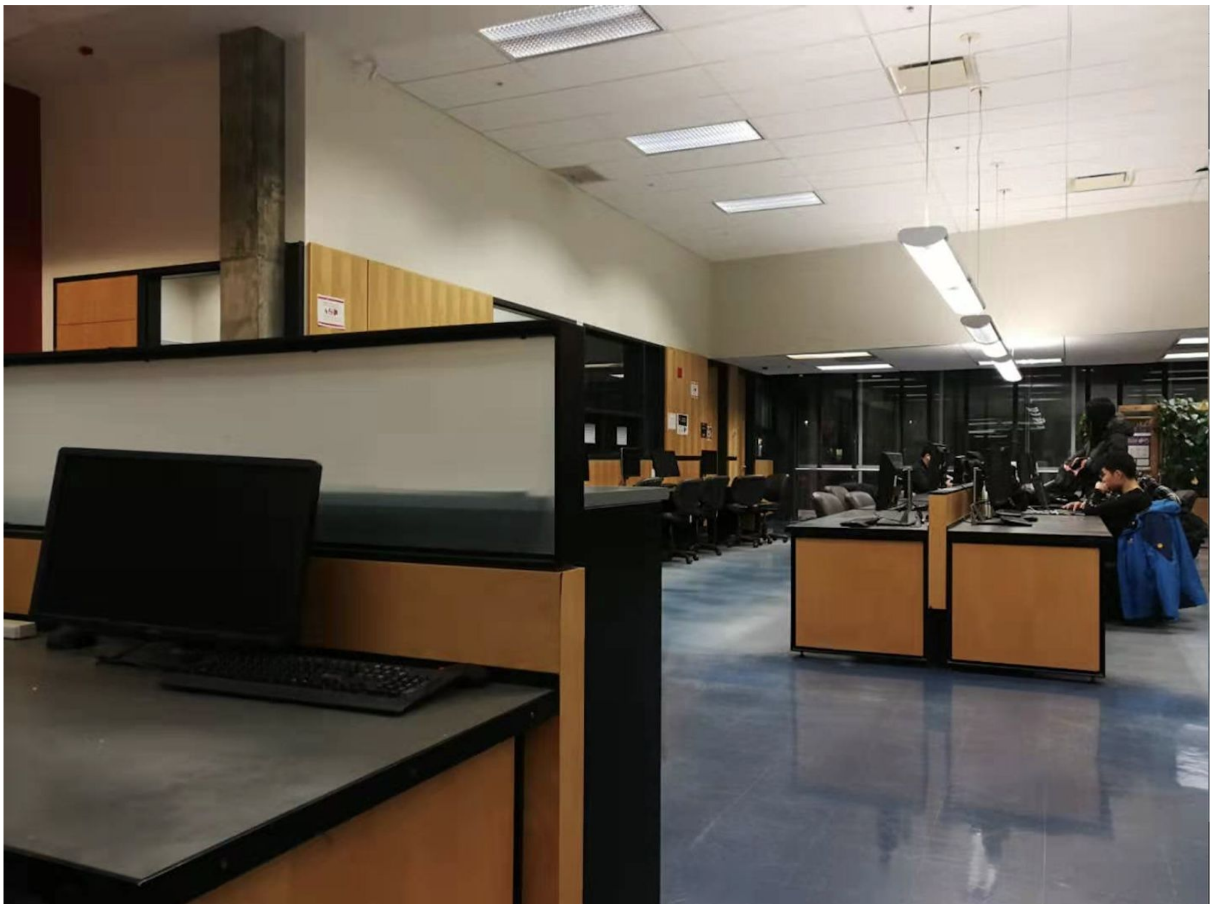


By using horizontal images 0, 1, 2:

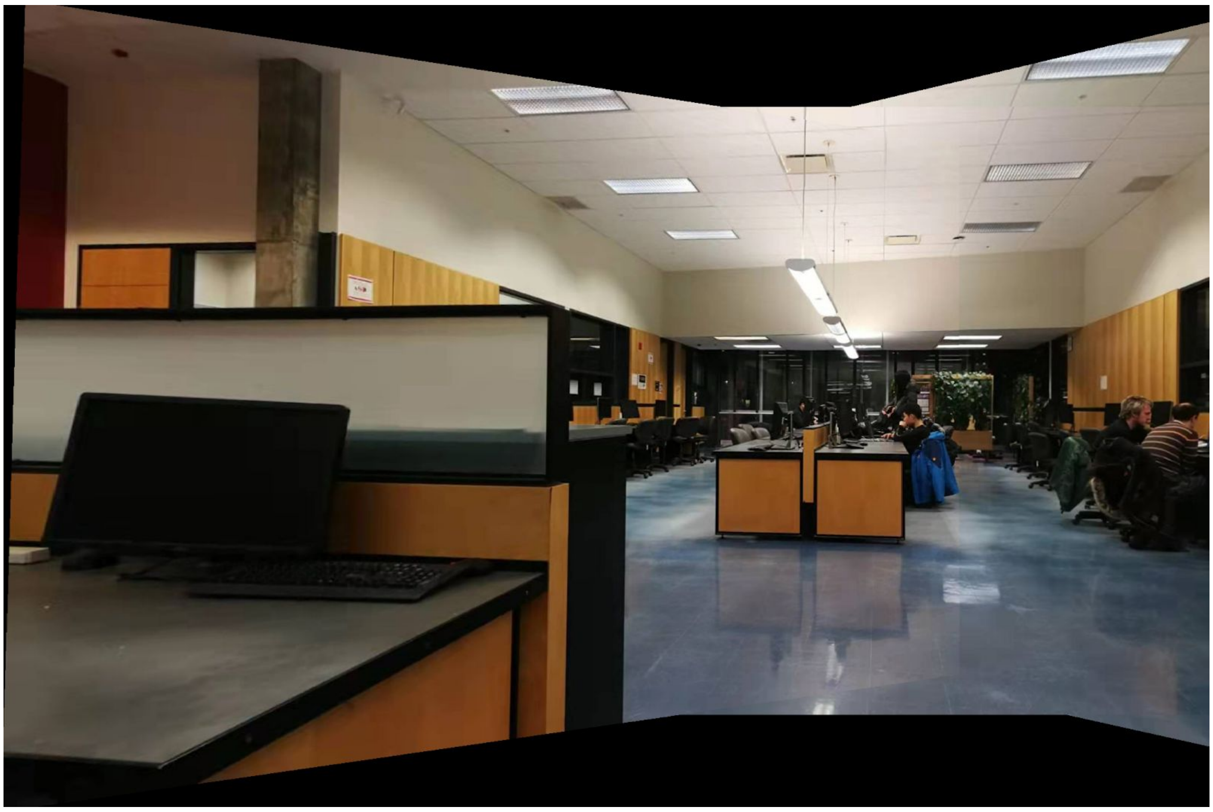


q5.

original image:



Panorama:



6.

I think the ghost edges are mainly caused by the intensity differences between images. For my images, you can see there are not very clear ghost edges.
So the basic idea here I think we should make the intensity distribution of two images similar. We can first compute the mean intensity values *mean1* and *mean2* of both images. Then, without loss of generality, we use image1 as the reference image. For each pixel value in image2, we just multiply it with *mean1/mean2*. Then the intensity distribution can be similar. Also, I find a paper [1] talking about how to alleviate this problem. In the paper, they first use a block-based transfer function to make it look more like it neighbours. Then they use two techniques to smooth variation in the transfer function distributions. The first one is to average functions in each patch with those of their neighbours. The second one is to blend the results of applying the transfer function from neighbouring patches.

[1] Uyttendaele, Matthew, Ashley Eden, and Richard Skeliski. "Eliminating ghosting and exposure artifacts in image mosaics." *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 2. IEEE, 2001.