CIS581: Computer Vision and Computational Photography

Project 3A – Extra Credit: Image Stitching

Group 9

Ang Li
Da Kuang
Jiaxiao Cai

This report aims to show the extra credit parts our projects.

- **Cylinder Projection**

The frame moisaicing results are projected to a cylinder as shown in Figure 1 and Figure 2. The code for generating cylinder projection is included in cylindricalWarp.py.

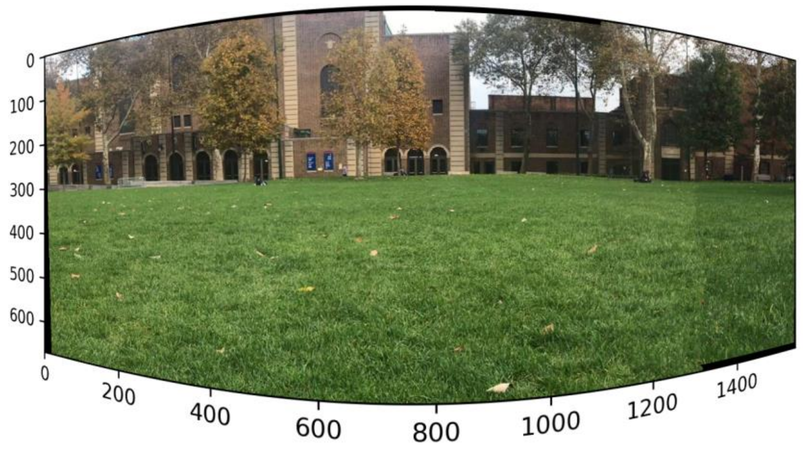


Figure 1 Cylinder Projection of Franklin Field

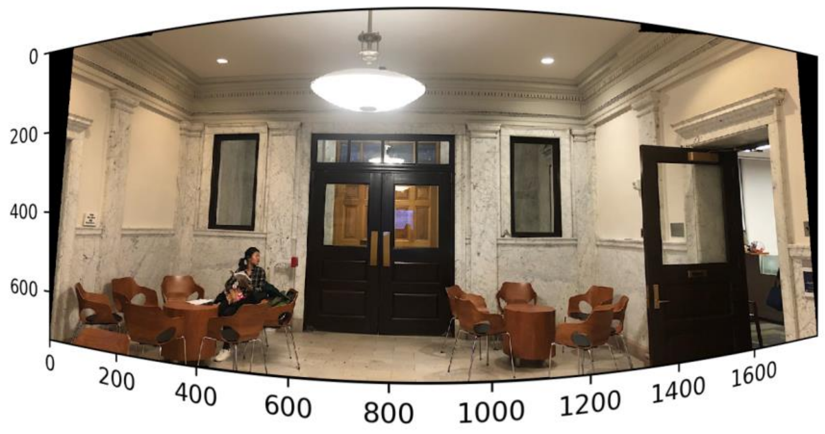


Figure 2 Cylinder Projection of Towne Hall

- ## Another Feature Match Methodology – Cross Check

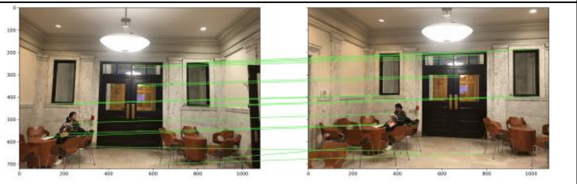
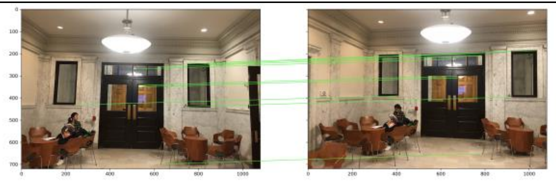
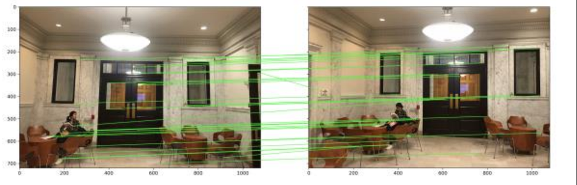
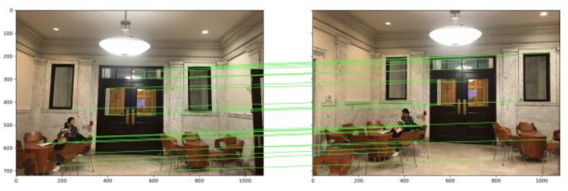
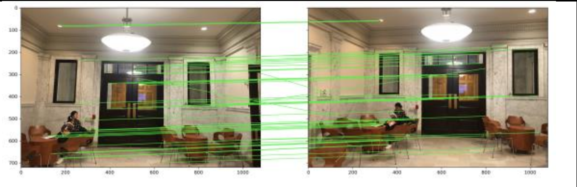
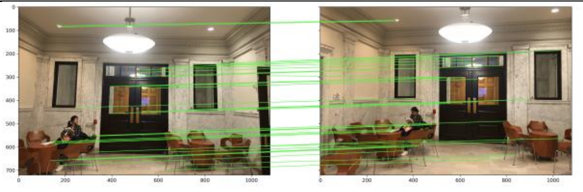
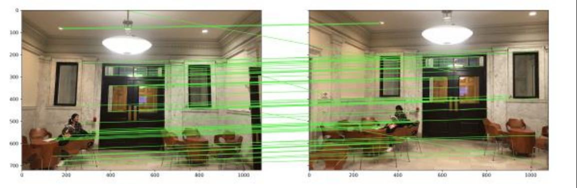
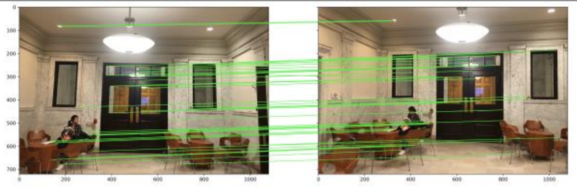
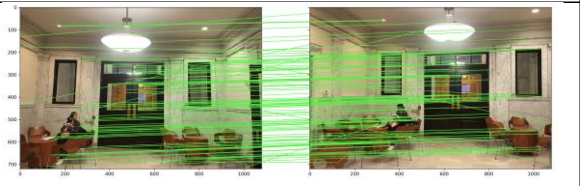
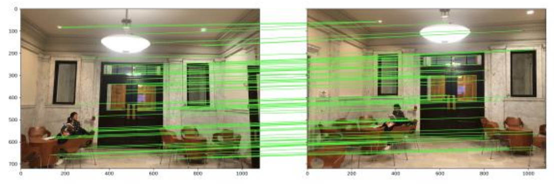
Besides of Ratio Test, we came up with another feature match methodology, which is **Cross Match**. The code is included in *crossMatch.py*. The key idea of Cross Match included the following steps:

I. For each key point in feature descriptor 1, calculate its corresponding nearest neighborhood in feature descriptor 2. And then store the index of these nearest neighborhoods. So, f_b is the best match for f_a.

II. For each key point in feature descriptor 2, calculate its corresponding nearest neighborhood in feature descriptor 1. And then store the index of these nearest neighborhoods. So, f_a is the best match for f_b.

III. Cross check when f_b is the best match for f_a, f_a should also be the best match for f_b.

The results of cross check, compared with ratio test, are shown as following figures. The test is done with different adaptive non-maximum suppression points, which includes 25 points, 50 points, 75 points, 100 points, 200 points and 300 points respectively. The ratio of ratio test is set to 0.7 in the experiment.

Cross Match	Ratio Test Match
	
	
	
	
	

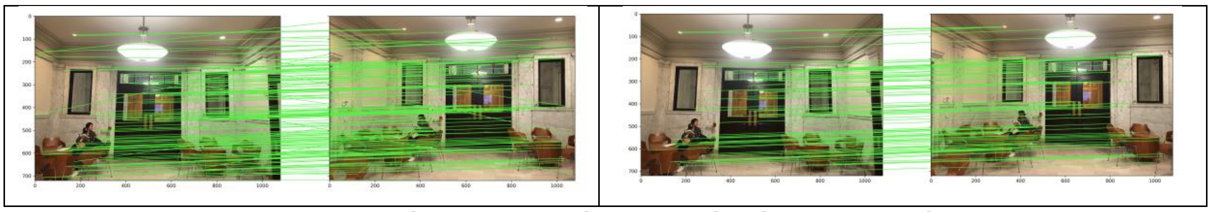


Figure 3 Result Comparison with Cross Match and Ratio Test Match

It can be concluded from the above figures that when the input corner points are the same and when the ratio in set to 0.7, the cross match tends to match more points than ratio test. However, there are more outlier in cross match than ratio test match. Indeed, the advantage of ratio test is that we could manually set ratio value to meet different purposes. It is also worth noting that we could actually combine these two methods, which are bi-directional ratio test taught in the lecture.

The final frame mosaicking result for cross match method is shown in Figure 4. It is quite satisfactory.

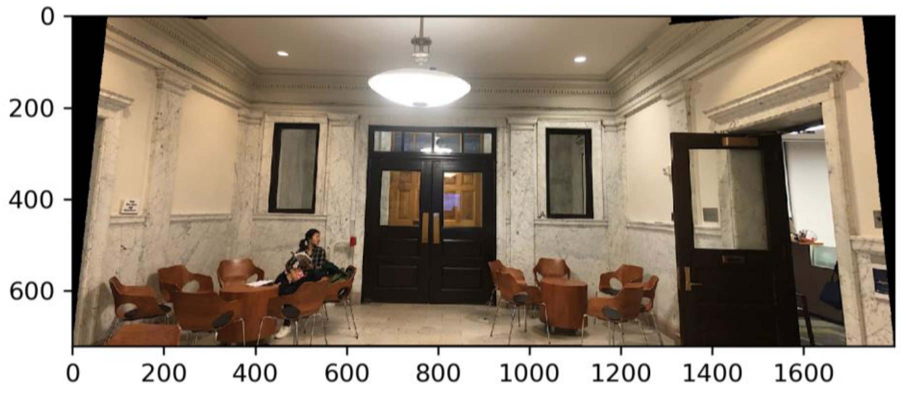


Figure 4 Frame Mosaicking Result for Cross Match

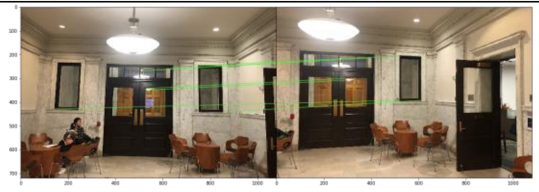
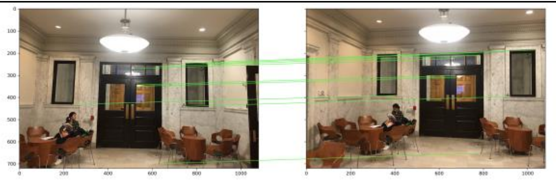
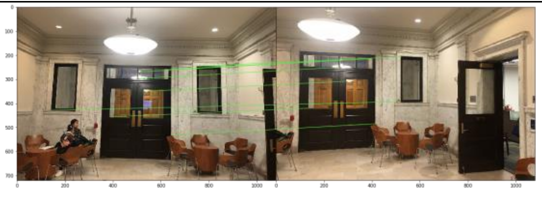
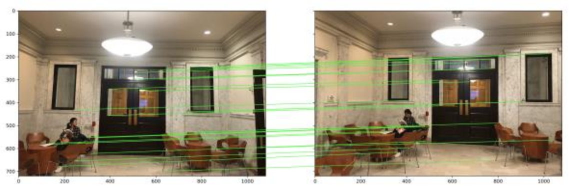
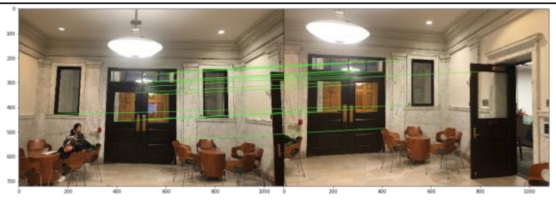
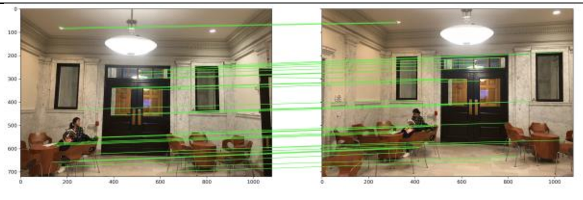
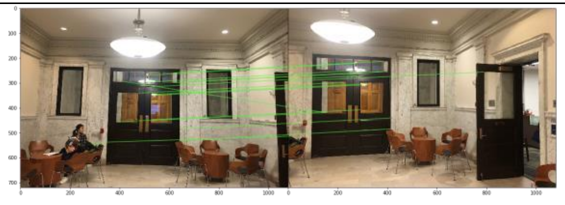
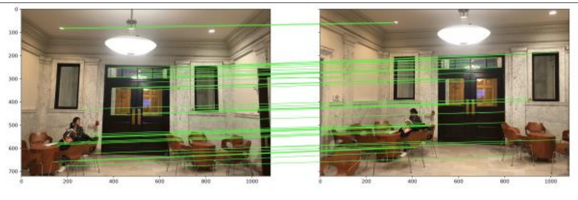
- **Our Own Descriptor**

Besides the standardized descriptor, we write our own feature descriptor and compare the results of our own feature descriptor with the descriptor taught in the class. The code is included in *our_feat_desc.py*.

I. Standardized Descriptor Method:
   a. Get the max gradient value from each 5x5 matrix in the big patch 40x40 - We'll get 64 of these values
   b. Then normalize these max values where the mean and std would be calculated from these 64 max values

II. Our Descriptor Method:
   a. For each corner feature, sample a patch from a larger 40x40 window to blur the image.
   b. Sub-sample these patches with a spacing of 5 pixels.
   c. Then we get a 8x8 patch of pixels, in terms of their pixel values, which then be normalized.

The comparing results for these 2 methods are shown in the following figures. The test is done with different adaptive non-maximum suppression points, which includes 25 points, 50 points, 75 points, 100 points, 200 points and 300 points respectively. The ratio of ratio test is set to 0.7 in the experiment.

Our Descriptor	Standardized Descriptor
	
	
	
	

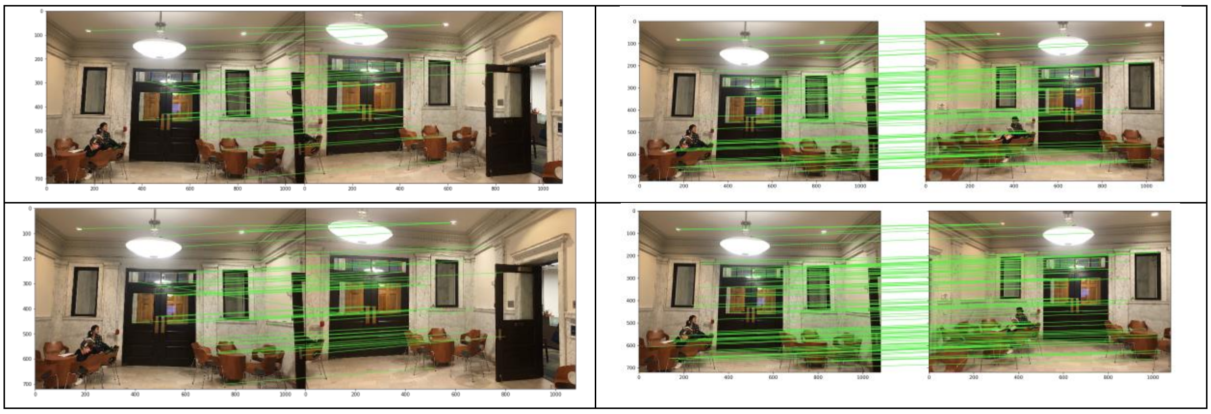


Figure 5 Result Comparison with Two Descriptors

It is clear from the above figures that our descriptor is not satisfactory. Because in each experimental setting, standardized descriptor is able to match up more points successful.

The final frame mosaicking result for our descriptor is shown in Figure 6. Although our descriptor is not as precise as the standardized one, the frame mosaicking result from our descriptor is quite satisfactory.

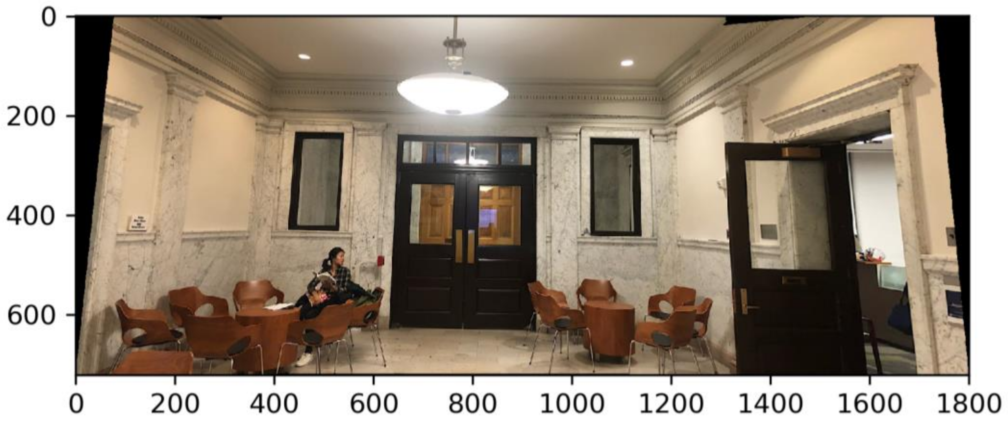


Figure 6 Frame Mosaicking Result for Our Descriptor

- **SIFT Descriptor**

Besides the standardized descriptor and our own descriptor, we also implement SIFT descriptor to extract feature. The code is included in *feat_desc_SIFT.py*, *helper.py and FLANN_Matcher.py*.

The result of SIFT descriptor of different inputs corner numbers are shown in Figure 7 - 11.

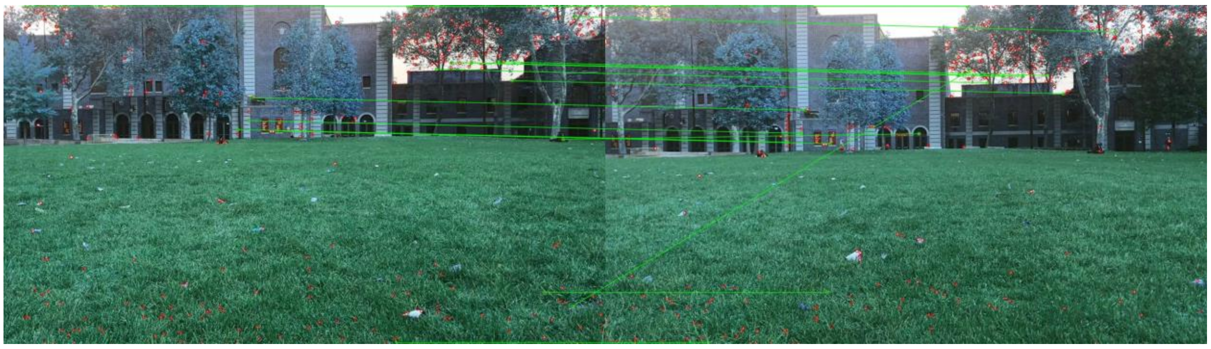


Figure 7 SIFT Descriptor Result of 300 Input Corner Points

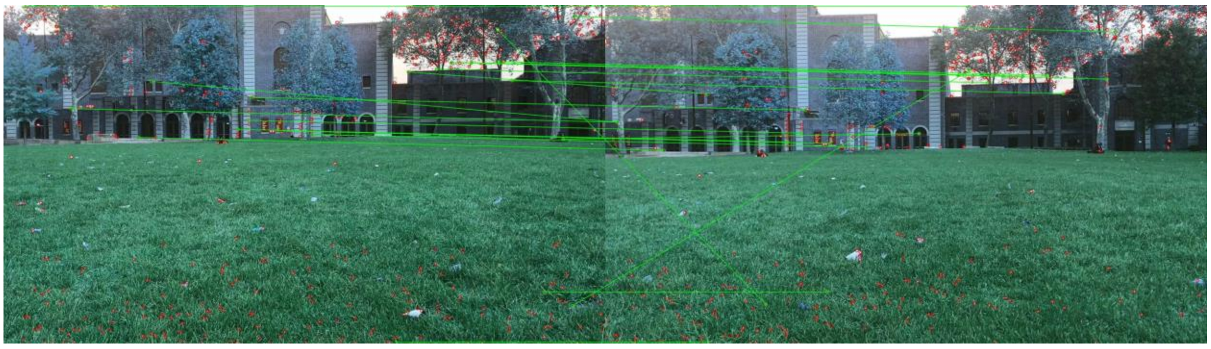


Figure 8 SIFT Descriptor Result of 400 Input Corner Points

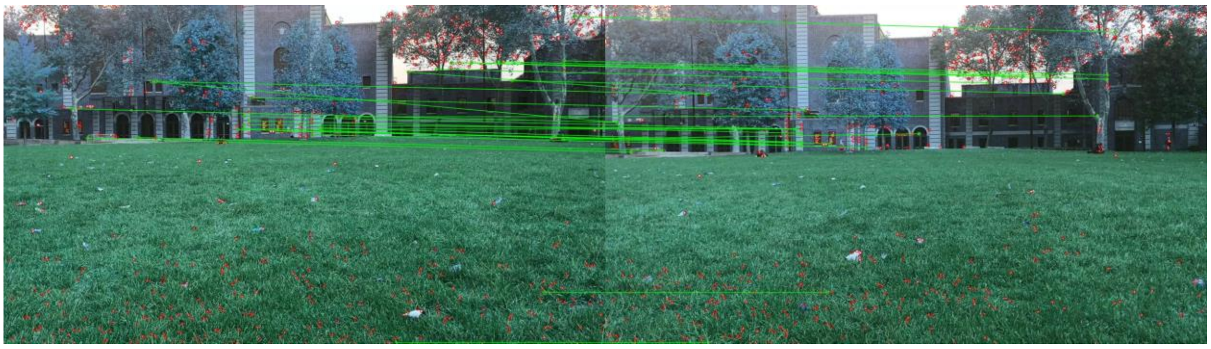


Figure 9 SIFT Descriptor Result of 500 Input Corner Points

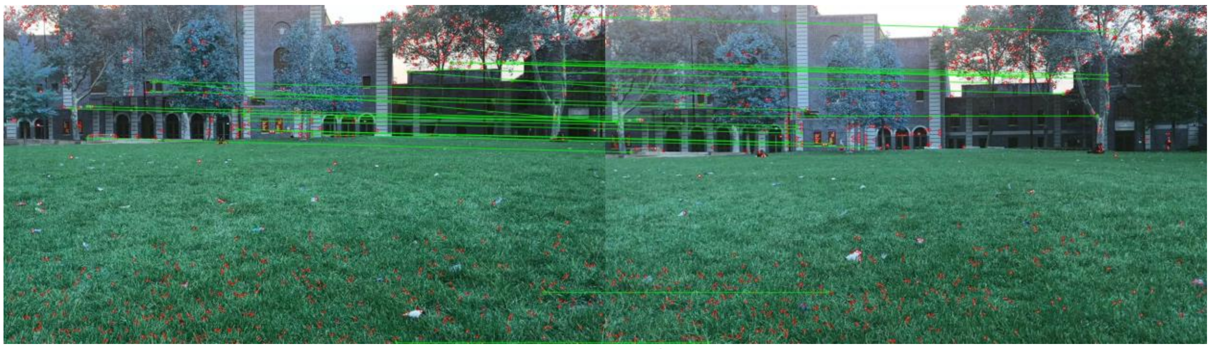


Figure 10 SIFT Descriptor Result of 600 Input Corner Points

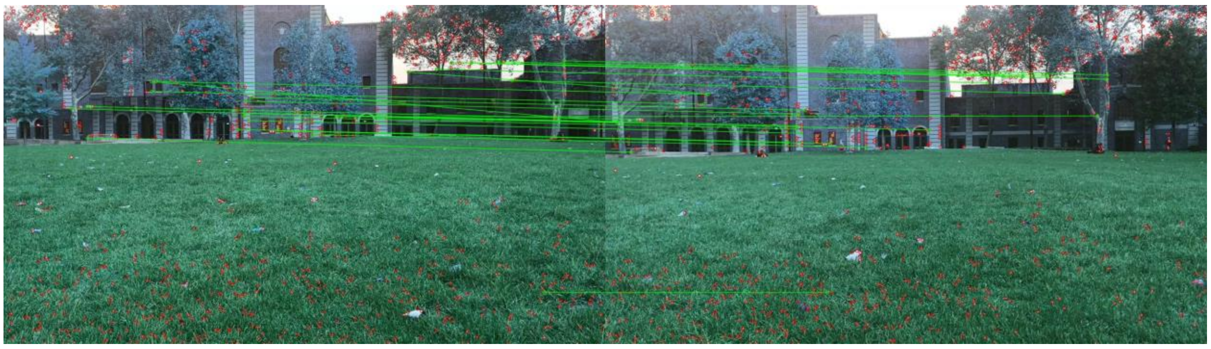


Figure 11 SIFT Descriptor Result of 700 Input Corner Points

he final frame mosaicking result for SIFT descriptor is shown in Figure 12. It is shown from the result that SIFT descriptor is quite satisfactory.

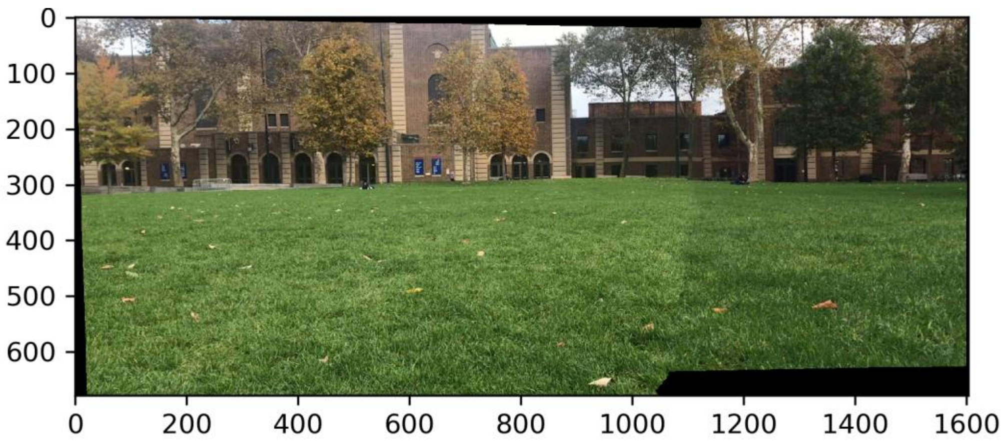


Figure 12 Frame Mosaicking Result for SIFT Descriptor