Computer Vision - Assignment 1

Gianzina Alexandra i629435

1 Implementation Details

In order to read the images I used cv2.imread("imageName"). The colorful images have 3 dimensions so I had to reshape the size of the images and change them to 2 dimensions. Turning images to gray scale immediately changes the dimension of the images to 2 dimensions. After that, I normalized the gray scale pictures to avoid different contrast in the images that need to be stitched.

For the feature detection I used corner Harris. I noticed that the values that work the best for the left image were with k=0.1, threshold_abs=0.00001 and threshold_rel=0.00001. For the right image the best results were with k=0.1, threshold_abs=0.0001 and threshold_rel=0.0001. In Figure 1 you can see the results.

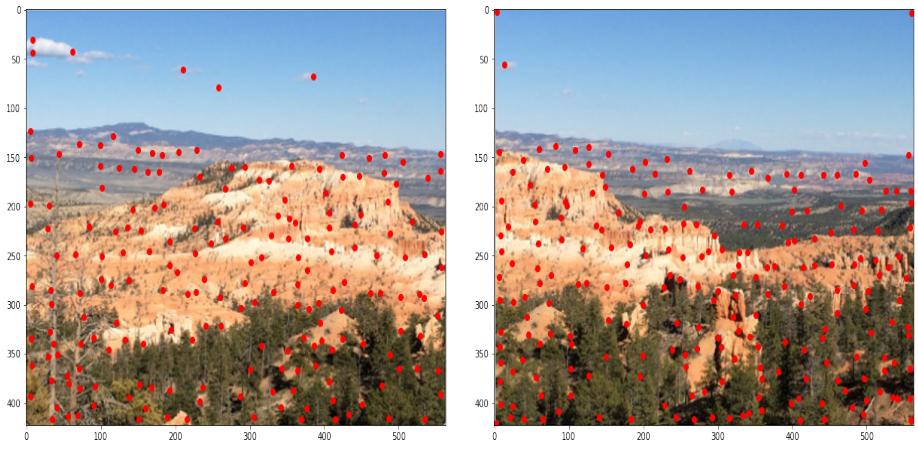

Figure 1: Feature detection with corner harris

I computed descriptors with SIFT using the function sift.compute(image, keypoints). In Figure 2 you can see the results of the descriptors. The results for the descriptors look less dense because they were customised, in comparison to Figure 3, where I immediately computed the keypoints and descriptors with the help of the function sift.detectAndCompute(image). The advantage of SIFT is that even if the image is scaled, the stitching will be good. So I implemented RANSAC on the keypoints produced by sift.detectAndCompute(image).

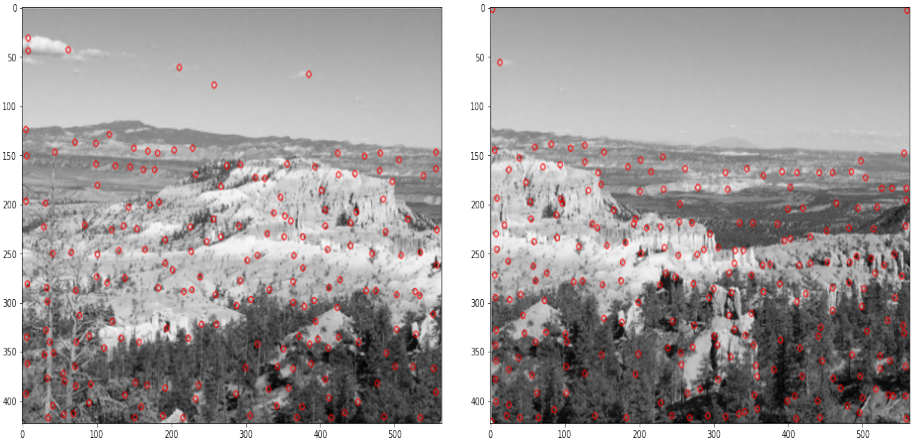

Figure 2: Calculate keypoints and descriptors with the help of SIFT

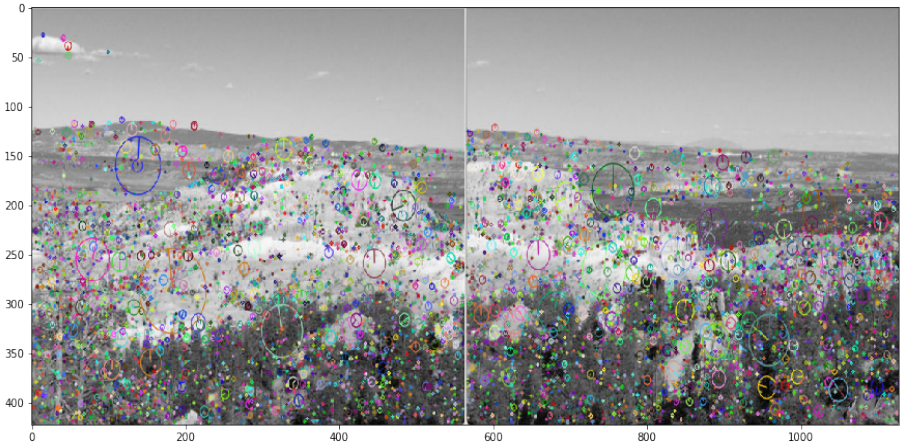

Figure 3: Descriptors using SIFT

I calculated all the possible matches and plotted them, as seen in Figure 4.

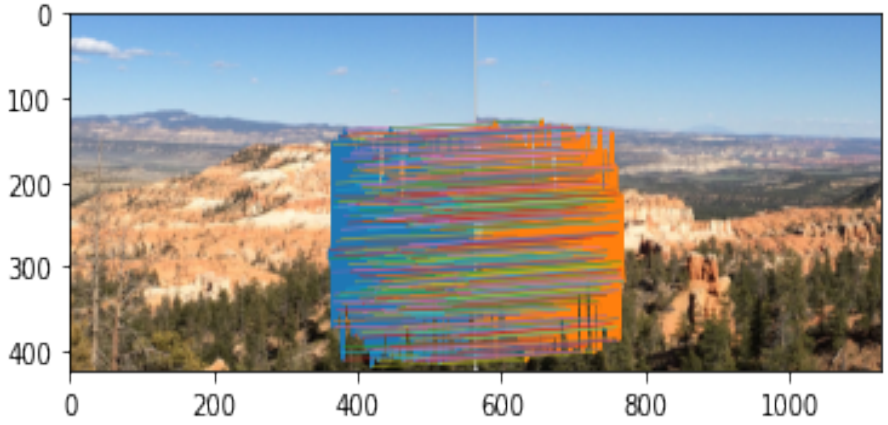

Figure 4: Matches without finding the inliers

I iterated for a fixed number to compute the inliers. Then I was able to find the transformation and stitch the pictures. In Figure 5 you can see the first test case for image stitching.

Figure 5: Image stitched

2 Experiments

In the first Experiment I allowed only the 10% of the matches in order to calculate the inliers. Figure 6 shows the matches and Figure 7 show the results.

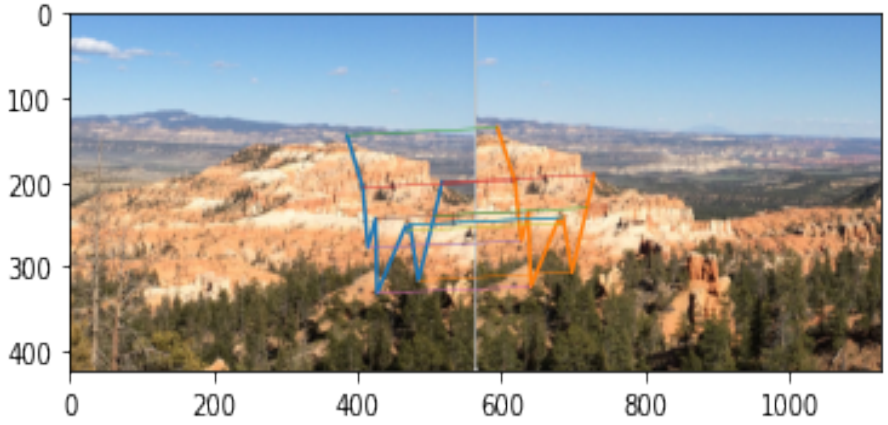

Figure 6: All the allowed matches

Figure 7: Image stitched

In the second Experiment I reduced the threshold for the RANSAC. Figure 8 shows the matches and Figure 9 show the results. We notice that this Experiment has the best stitching results.

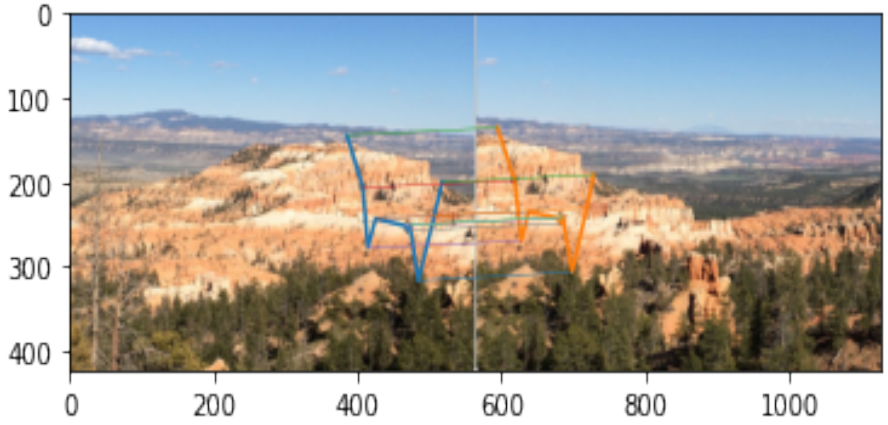

Figure 8: All the allowed matches

Figure 9: Image stitched

In the third Experiment I reduced the number of iteration in order to compute inliers. Figure 10 shows the matches and Figure 11 show the results.

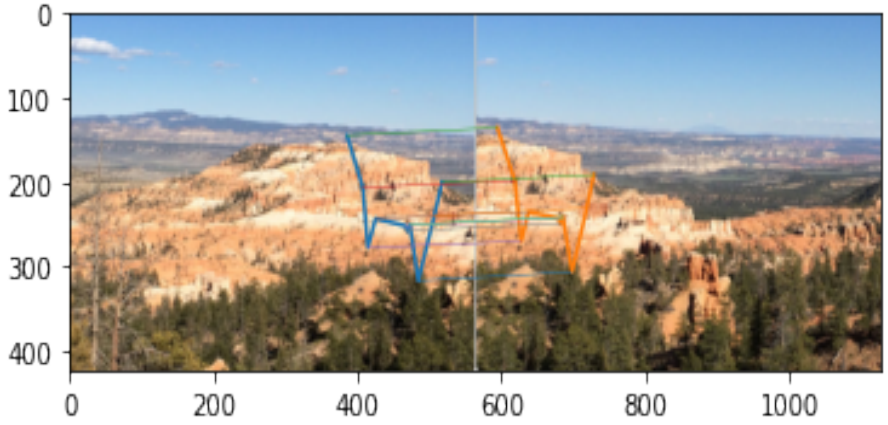

Figure 10: All the allowed matches

Figure 11: Image stitched

3 Difficult test case

I iterated for a fixed number to compute the inliers for the difficult case, I used the same numbers that I used in the first test case as I noticed that these values had very good performance and I wanted to check whether they will perform the same. Then I was able to find the transformation and stitch the pictures. In Figure 12 you can see the difficult test case before stitching and in Figure 13 you can see after stitching. We can see that the results are very good.

Figure 12: Difficult test case before stitching

Figure 13: Image stitched