Name: ZHANG Xiaojun StudentID: 1155181655

ASSIGNMENT 3 REPORT

Local Feature Matching

1. Dataset

The provided dataset contains 10 images, which can be considered as 5 pairs of images as following:

1. Notre Dame images
2. Mount Rushmore images
3. Episcopal Gaudi images
4. Test1
5. Test2

2. Approach

2.1 Interest Point Detection

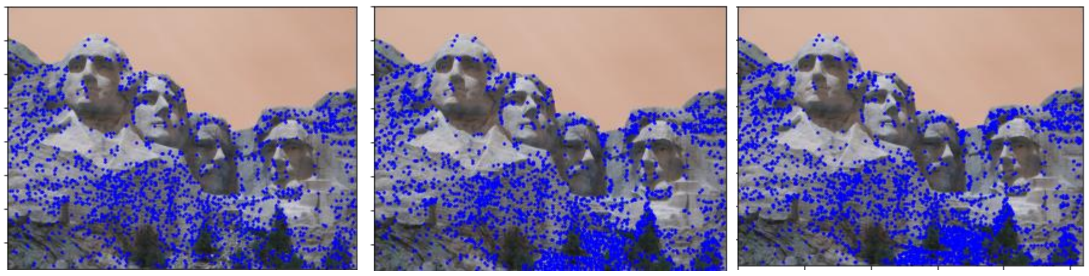
2.1.1 Algorithm

In order to extract the features of the image, first of all, the Harris corner detector will be applied to find corners. The procedure of this algorithm implementation is as following:

1) Spatial derivative calculation: compute the derivatives in x and y directions.
2) Harris response calculation: $R = \det(M) - k * (trace(M))^2$
3) Find corners using R: the point belongs to corner if R>0
4) Group the corners from same corner regions detected from last step into one point.

Based on the principle, I have implemented the algorithm only using numpy module. And to simplify the code, I just use opencv built-in function. The procedure is as following:

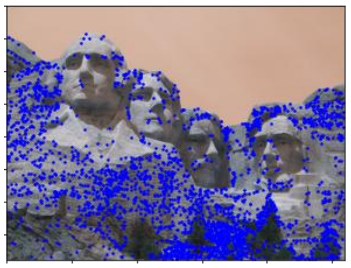
a) Use function cv2.cornerHarris(image, blockSize, ksize, k) to obtain all corners, where blockSize is a key parameter for corners detection. I change the blockSize to different values(blockSize=2,4,8) and the results are as following:

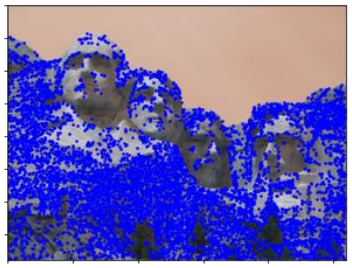


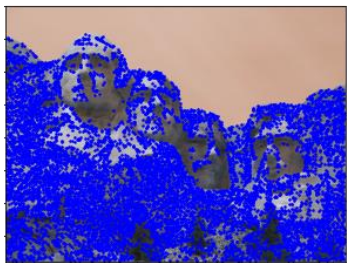
blockSize=8 blockSize=4 blockSize=2

It's obvious that with the blockSize value decreasing, more corners could be detected. In final stage, to get high accuracy, different values are tested.

b) Use function cv2.threshold(dst, m*dst.max(),255,0) to filter the corners detected from last step. At this stage, m is a key parameter for the performance of detecting. I change the m to different values(m = 0.01, m=0.002, m=0.0008)


m = 0.01


m = 0.002


m = 0.0008

c) Use function cv2.connectedComponentsWithStats(dst) and cv2.cornerSubPix(gray,np.float32(centroids),(5,5),(-1,-1),criteria) to find the corner point among the corners from the same group. At this stage, I use default setting and delete the background component.

d) Save x and y values of all corners

2.1.2 Discussion

It's obvious that if decreasing the blockSize and m value, the algorithm can capture more interest points. Since if the threshold is large, less points can satisfy the condition. Although getting more interest points means the accuracy of features matching can be higher, it's will take longer time to compute.

2.2 Get features

2.2.1 Algorithm

1) Convolve the gaussian kernel with the whole image using function cv2.GaussianBlur
2) Calculate the image gradient of the Gaussian blurred image using function dx = cv2.Sobel(blur,cv2.CV_32F,1,0,ksize=5) dy = cv2.Sobel(blur,cv2.CV_32F,0,1,ksize=5)
3) Calculate the magnitude and orientation of each pixel
4) For each detected interest point, consider a 16*16 window centered at this point. And divide this window in 4 * 4 cells. The magnitudes of this window are further weighted by a 2D Gaussian function (obtained from cv2.getGaussianKernel(ksize = feature_width, sigma = feature_width//2) and multiply with its transpose)
5) For each cell, cast the orientations of this cell into 8 bins according to the degree of each pixel. For each bin, accumulate the magnitudes of the pixels belonging to this bin.

6) Concatenate these 128 values into one vector and then normalize the vector using np.linalg.norm.

2.2.2 Results

This algorithm output a SIFT descriptor with a matrix k * 128, where k is the number of interest points.

2.2.3 Discussion

I write the algorithm code refers to the guidance and reference book , check and revise it a lot of times, but it seems that the accuracy of my result is lower than the reference provided in the guidance file. I will try to improve the performance

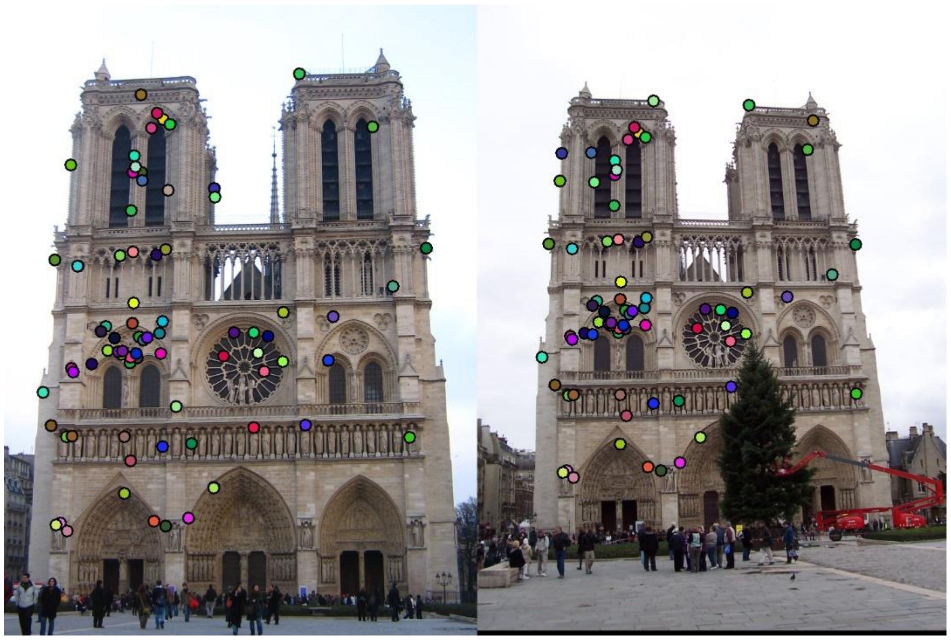
2.3 Feature matching

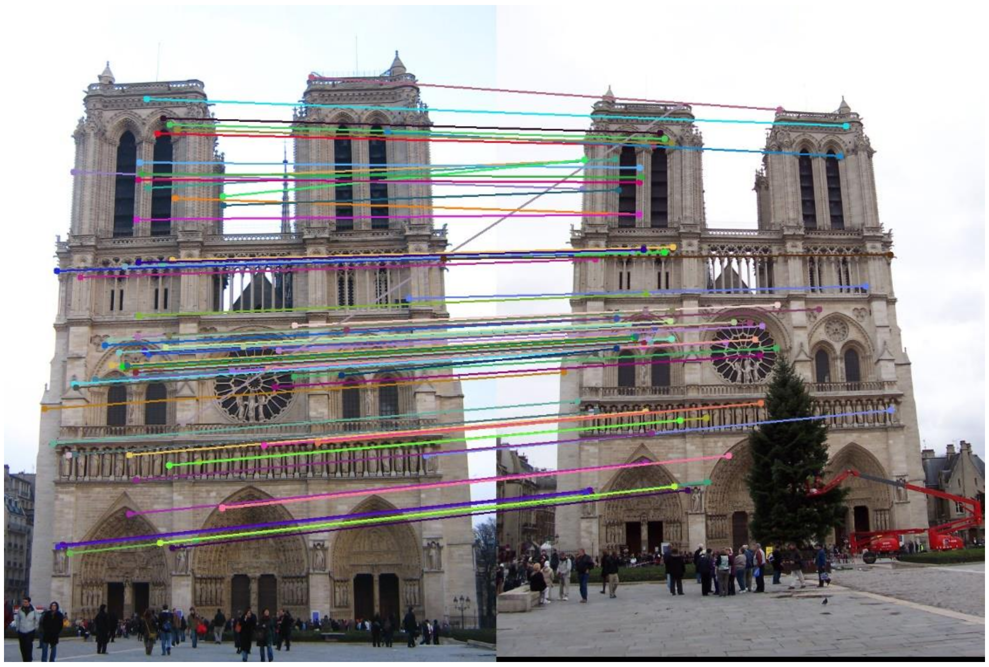
2.3.1 Algorithm

1) Number of features is the minimum number of the two feature matrices
2) Compute the feature distance (Euclidean distance) between the key-points.
3) Compute the confidence for each point pairs with the ratio test.
4) Set a threshold value and find all matching pairs

3.Result

Result 1



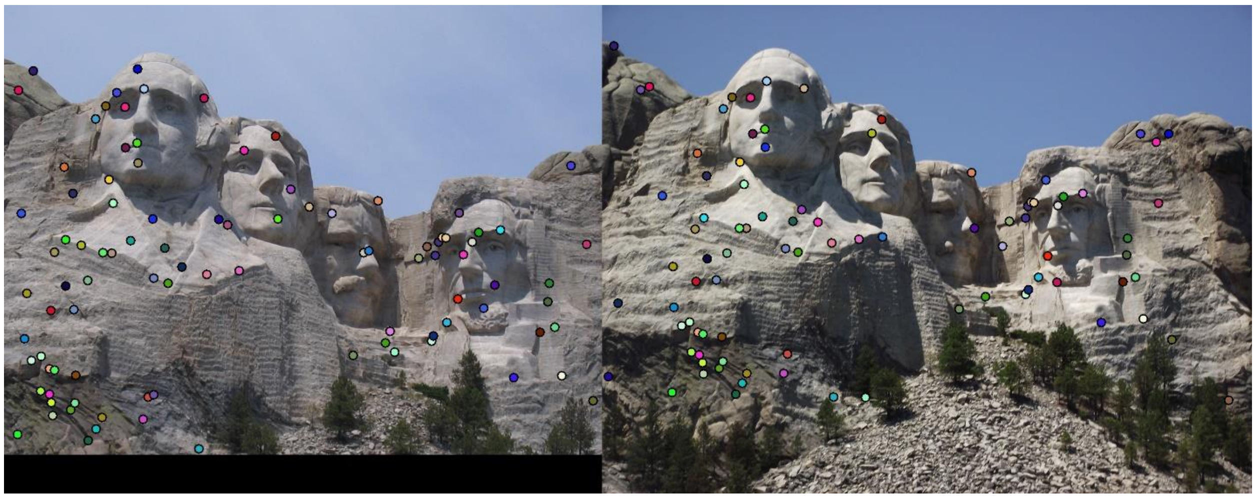


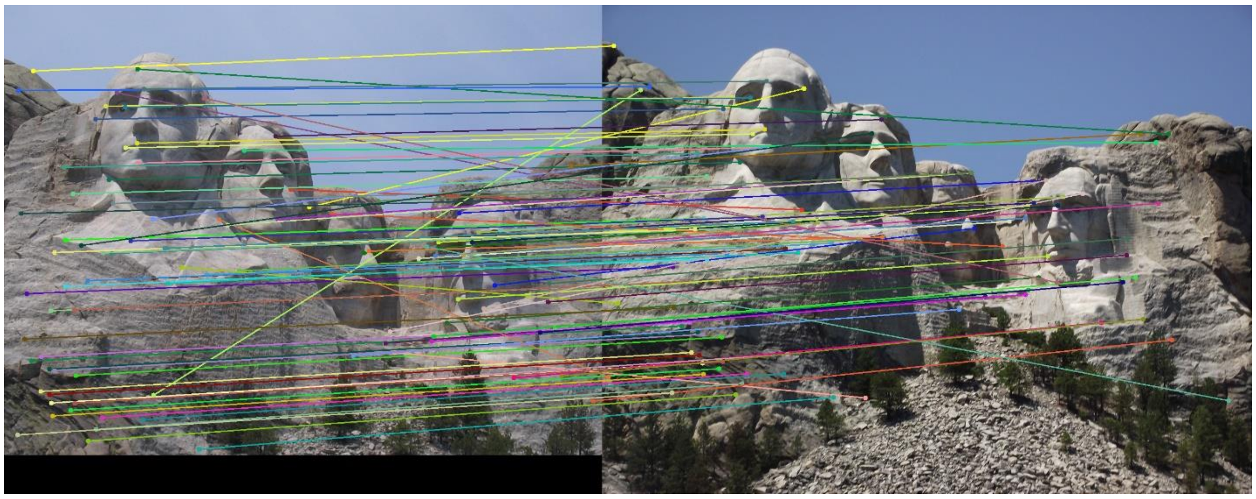
90 total good matches

10 total bad matches

90.00% accuracy.

Result 2



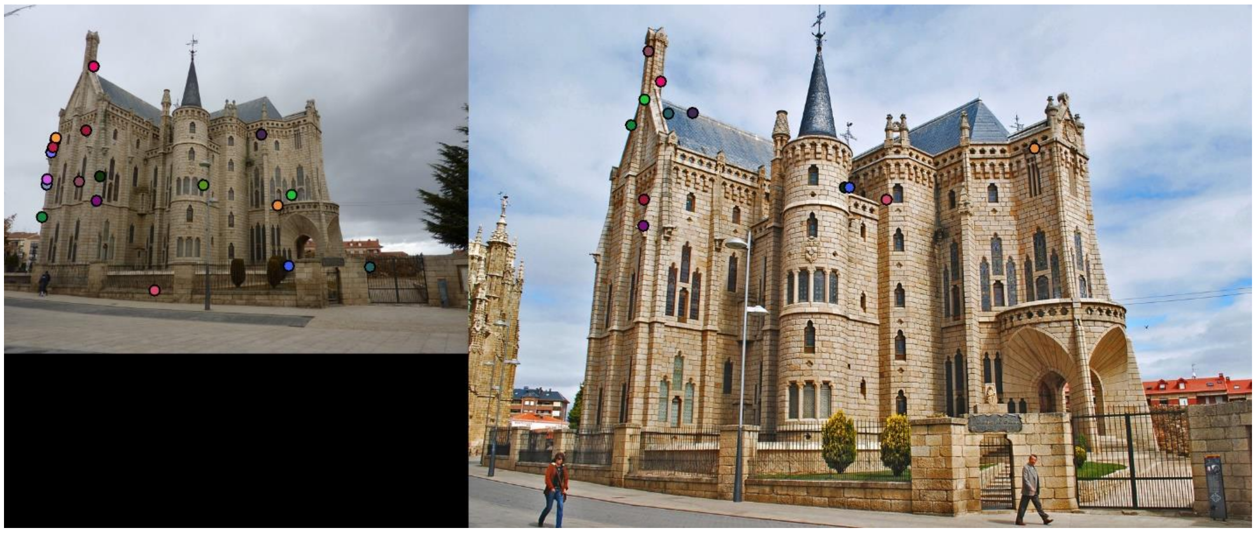


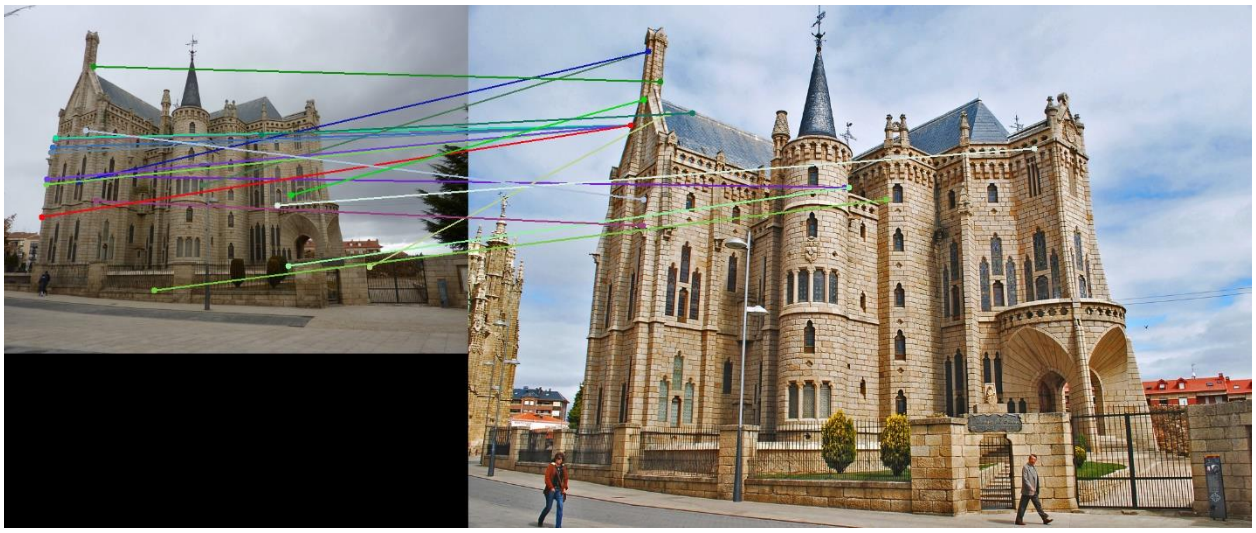
75 total good matches,

25 total bad matches.

75.00% accuracy.

Result 3



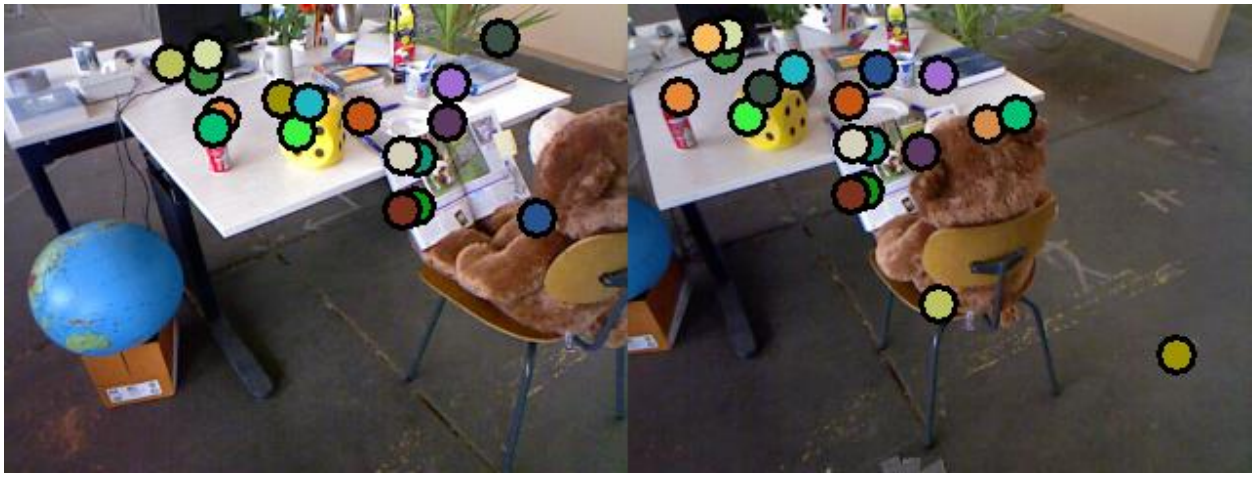


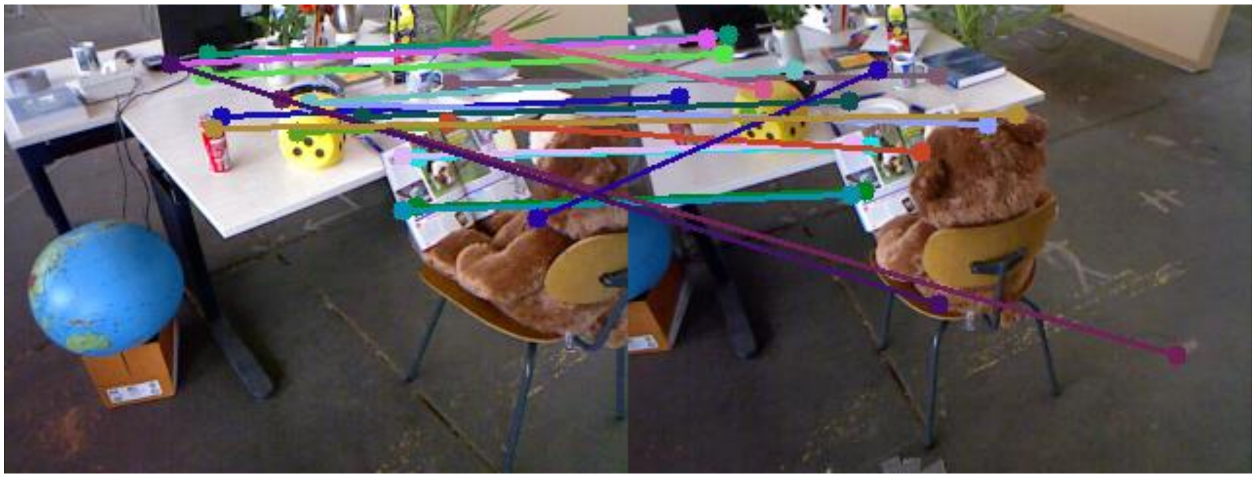
2 total good matches,

18 total bad matches.

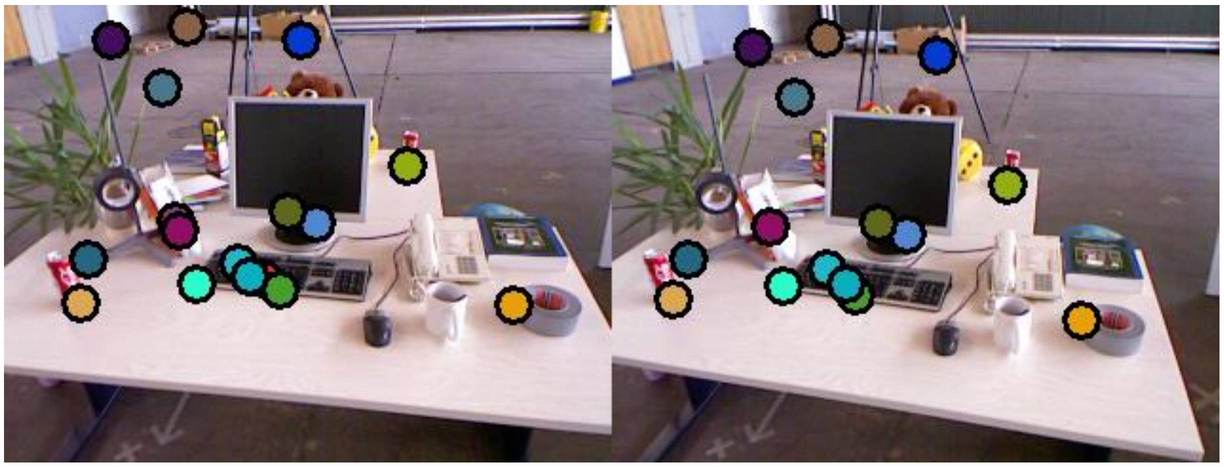
10.00% accuracy.

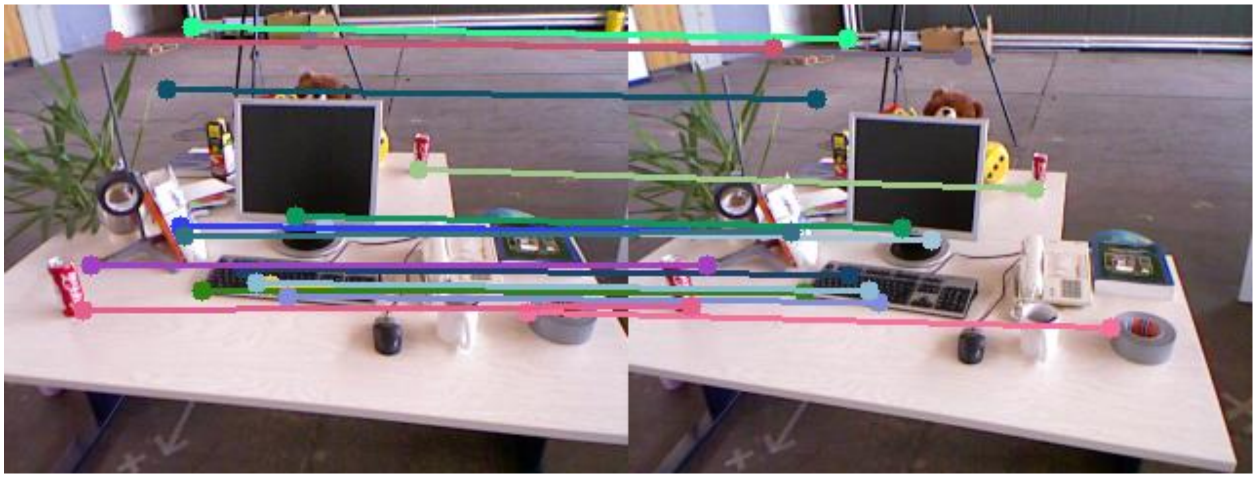
Result 4-test1





Result 5-test2





4.conclusion

In this project, Harris corner algorithm and SIFT algorithm were implemented. As we can see, in some cases, the accuracy is very low. The algorithm should be improved in applications.