Computer Vision KEN-4255

Assignment 1-Image Stitching

Maastricht University

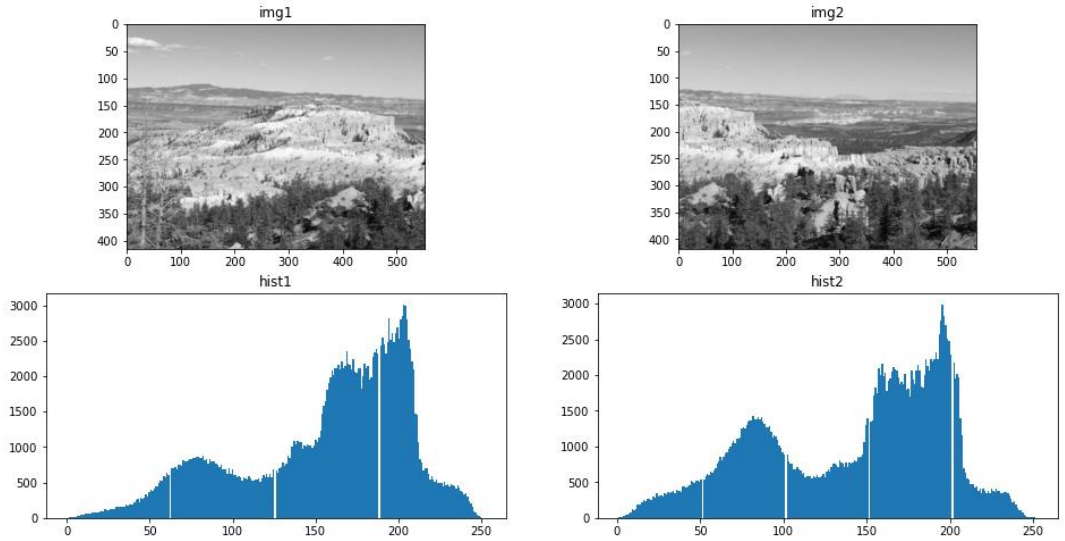
Athanasopoulos Nikolaos i6310104

Table of Contents

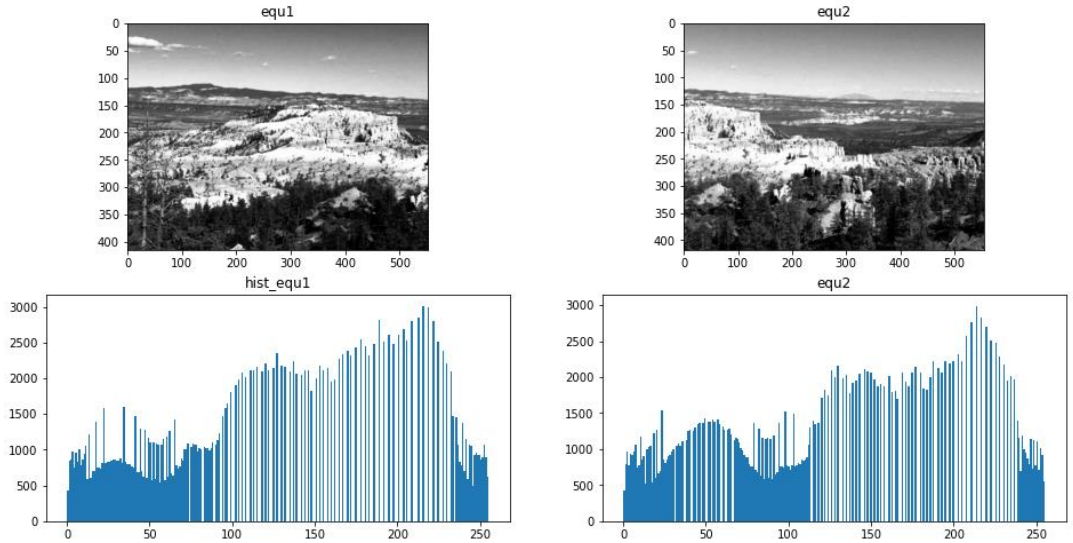
1: Harris Corners

Chapters 1-7 include results for the image pair given in the description of the assignment. However, chapter 8 includes results for many other image pairs, all of which are included in the zip file. To load them into the notebook, simply put them in the same directory as the notebook, making sure they have the same name as in the cv.imread(“ “) command.

Firstly, we present the images used and their respective histograms:

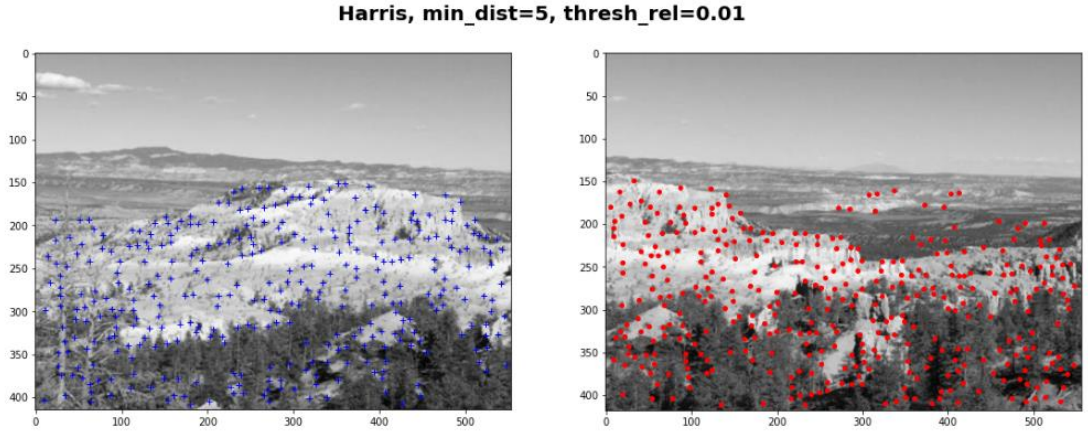


Equalizing their histograms via the cv.equalizeHist function gives us:

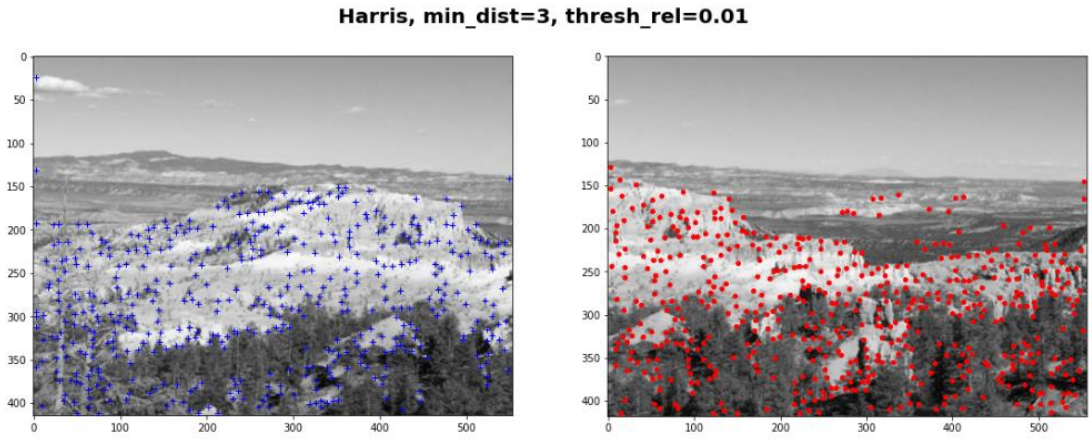


We will begin by applying the skimage corner_peaks after harris_corner function to the 2 images given in the description of the assignment. The first set of parameters are min distance which indicates the minimum distance used to aggregate the resulting corners and

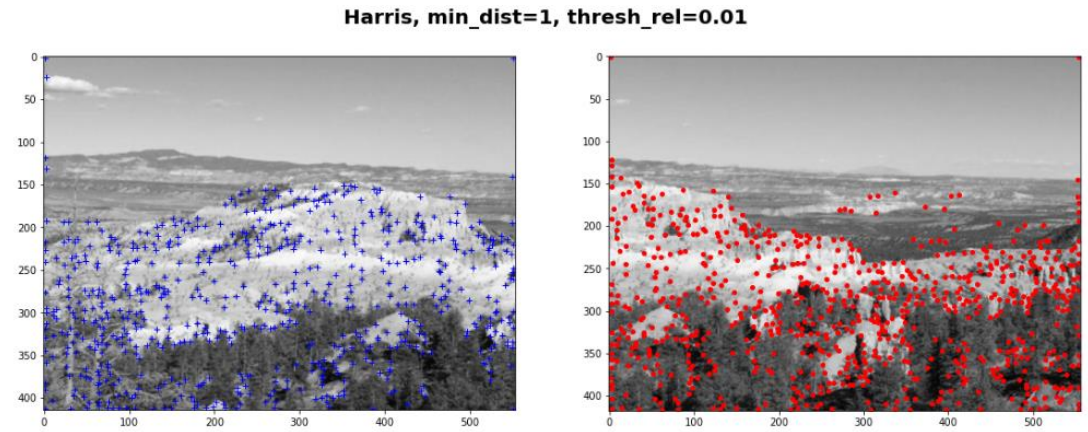
the threshold value which impacts the number of corners returned (the higher the threshold, the less corners are obtained). The results are:



Changing the minimum distance to 3 and keeping threshold_rel=0.01 gives us:

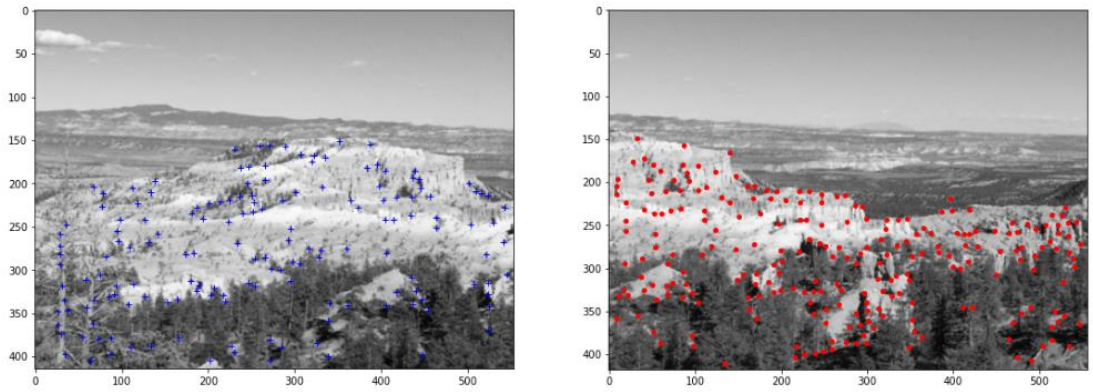


Changing the minimum distance to 1 and keeping the same threshold_rel gives us:



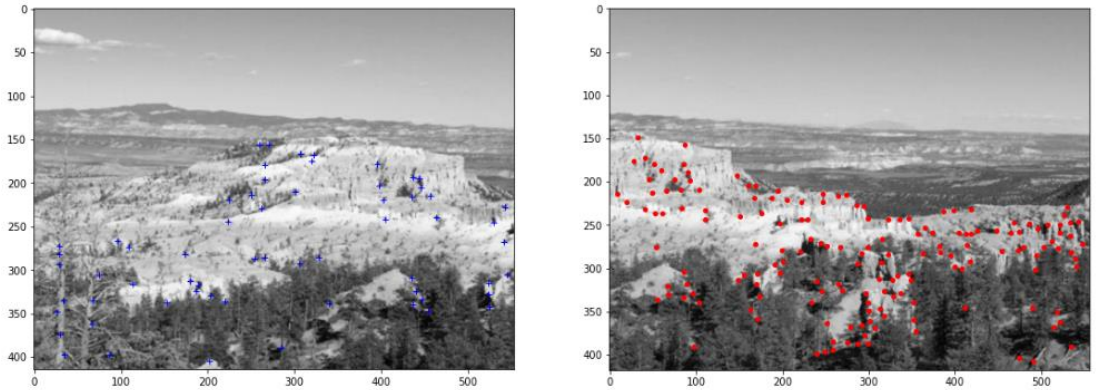
Setting the minimum distance to 5 and the threshold relative to 0.03 gives us:

Harris, min_dist=5, thresh_rel=0.03



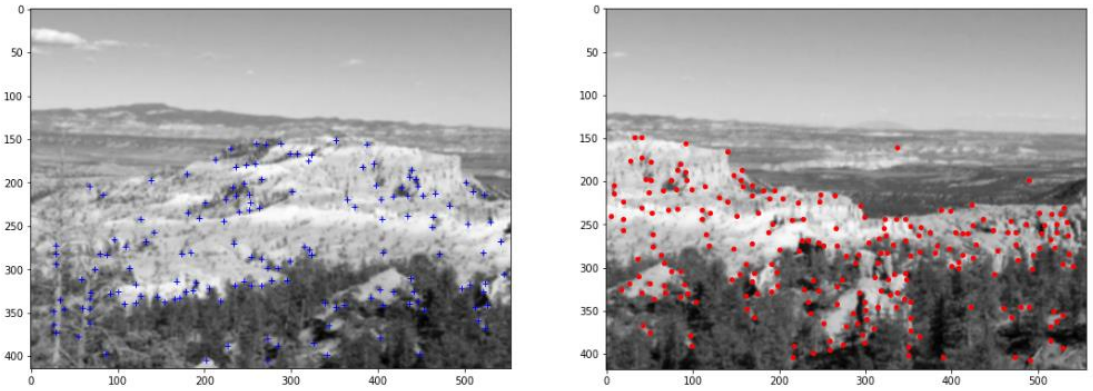
Setting the minimum distance to 5 and the threshold relative to 0.06 gives us:

Harris, min_dist=5, thresh_rel=0.06



Blurring the images with a Gaussian blur of (5,5) and then computing Harris corners for minimum distance=5 and threshold_rel=0.01 we obtain:

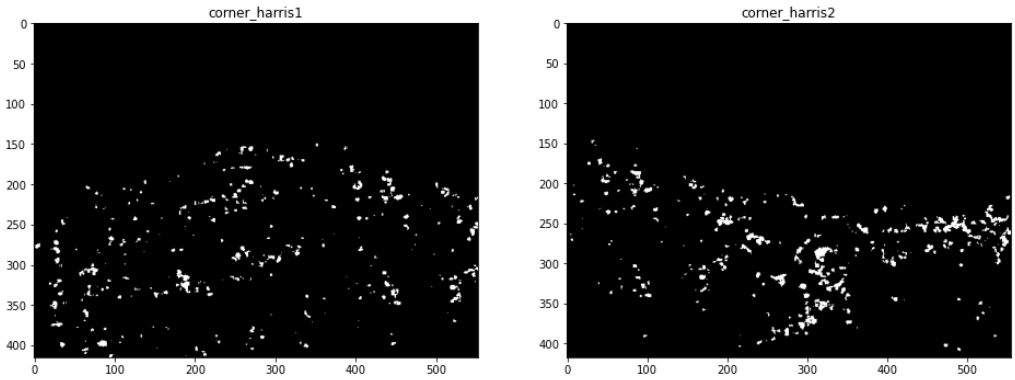
Harris, min_dist=5, thresh_rel=0.01 Gaussian Blur



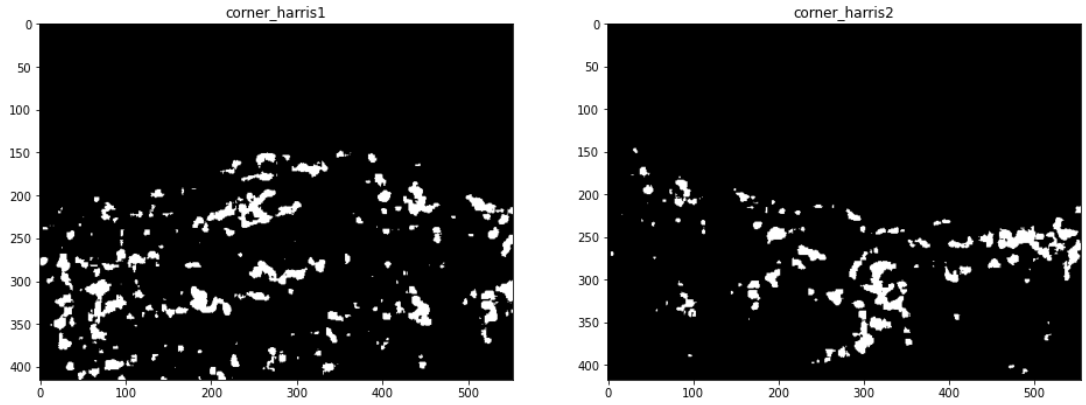
We can see that for minimum distance<5 we have a lot of corners around the same area, for example the trees on the left side of the left image. This is something that we want to avoid as too many similar corners in the same spot do not give us more interesting patches for the

stitching later, because the righthand image does not contain these corners. For these 2 specific images we are more interested in the corners that appear in the common section of the images. Of course, this is subject to change with another pair of images, but this is something that we have to pay attention to in each instance. Moreover, the threshold_rel parameter is a direct way to influence the number of corners obtained by the function. We can see that setting this parameter to around 0.03 to 0.06 gives us the best results: not too many corners but also not too few either. Of course, blurring the image with a Gaussian blur and then computing Harris corners yields less corners than the straightforward Harris method with the same parameters, which confirms what we have learned in theory.

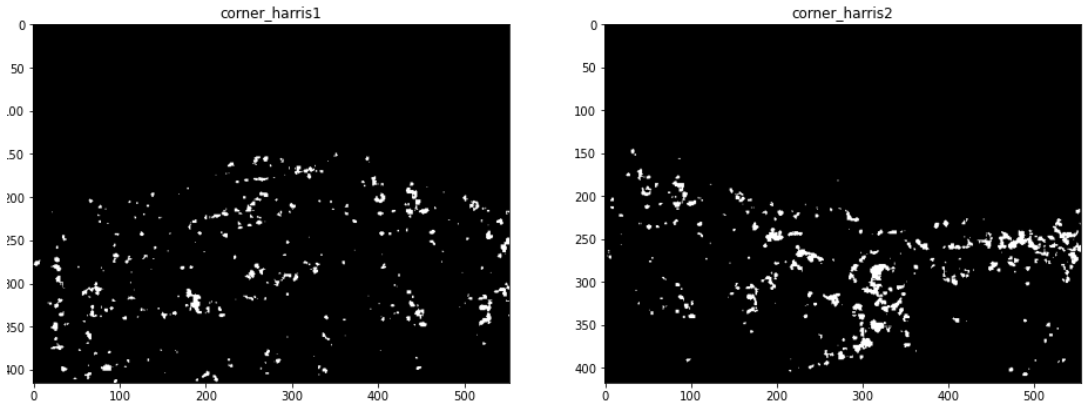
If we want to look at the binary form of those harris corners, with blocksize=5, ksize=3 and k=0.1, we obtain:



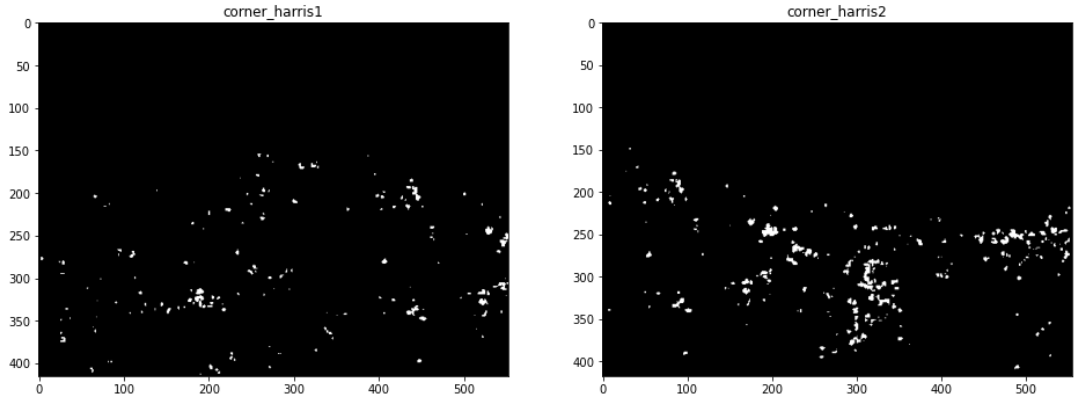
Setting blocksize=9 and keeping all the other parameters the same we obtain:



Setting blocksize=5 again and k=0.05 we obtain:



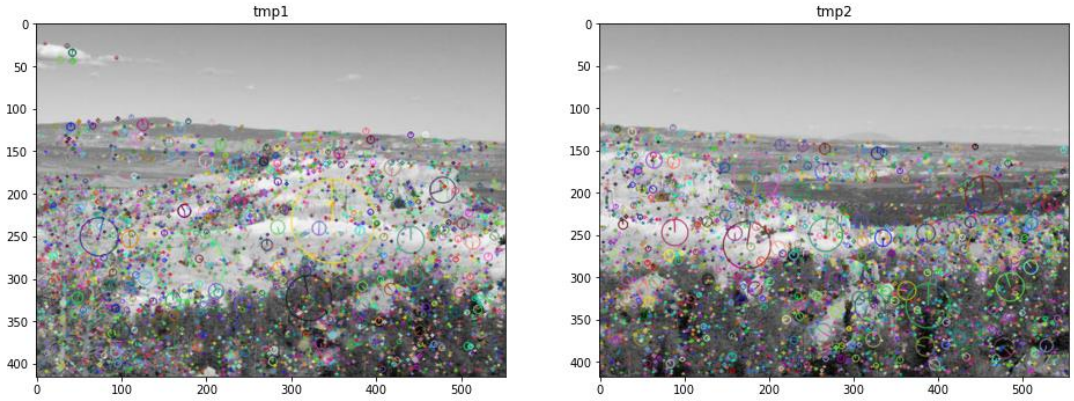
Setting blocksize=5, k=0.05 and ksize=9 we obtain:



We can see that higher ksize leads to fewer corners obtained, and that higher blocksizes lead to thicker patches of corners obtained.

2: SIFT Descriptors

To compute the sift descriptors for the images, we will use the class cv.SIFT_create and the function sift.detectAndCompute, both of which are now in the main repository of opencv. After this, we can draw the rich keypoints with the function cv.drawKeypoints. This yields the following result:



Of course the images are cluttered, but the best descriptors are filtered in the next step of the assignment, via the Euclidean distance and the normalized correlation, so there is no need to pass any parameters to the function.

3: Descriptor Distances

For the normalized correlation, the formula is:

$$d(H_1, H_2) = \frac{\sum_k [(H_1(k) - \overline{H_1})(H_2(k) - \overline{H_2})]}{\sqrt{\sum_k (H_1(k) - \overline{H_1})^2} * \sqrt{\sum_k (H_2(k) - \overline{H_2})^2}}$$

Where:

$$\overline{H_i} = \frac{1}{N} \sum_{k=1}^{N} H_i(k), \qquad i = 1,2.$$

In our case, each descriptor has 128 bins (typical SIFT descriptors) so N=128. (In python, we start from k=0 so N=127 for 128 bins). For easier code readability and running efficiency, we can use cdist from scipy.

In the case of Euclidean distance, we can use the function distance_matrix from scipy, after normalizing each descriptor through its max and min value (this is done for every row in the descriptor arrays). The formula is the typical formula for Euclidean distances between vectors of N coordinates.

4: Best Matches

From the ndarray of Euclidean distances, we can apply a threshold distance and keep the respective best matches each time. This threshold parameter is of course relative for each image pair that we want to stitch. In addition, this parameter was found to be the most important one with regards to the stitching accuracy. It was further discovered that in order to produce a quality image stitch, this parameter had to be set to a value that would provide at most 150-300 pairs of points (typically at most 150). If this value was set too high (and thus a lot of matches were kept for RANSAC, for example 10,000), then RANSAC sometimes produced a highly distorted image result. This could be sometimes solved by setting the RANSAC maximum iterations high, but of course this also increased the time to complete the estimation of the transformation. So, in order to provide a good image stitch, we have to consider at most 150 pairs of points-matches (either by setting a Euclidean distance threshold, or by considering the first best 150 matches by sorting the ndarray). For the purposes of this assignment, the method with the threshold was selected, as it provides yet another parameter that we can tweak and see what results it influences. For the image pair in the description of the assignment: this threshold was found to be optimal at around 0.5-0.6. For values lower than this, the image stitch was still very good. For values more than 0.8, the image stitch sometimes fails.

5: RANSAC Implementation

The methodology for RANSAC in this report is straightforward. The code is explained in the comments of the Jupyter Notebook. The other parameters introduced here are the init size (the amount of points that estimate the affine transformation), the maximum number of RANSAC iterations and the tolerance under which 2 points are considered inliers. The init size does not influence the estimation of the affine transformation if we keep all the other parameters constant. Even setting the init size to 10, we can see that still the affine transform can be efficiently calculated, so this parameter is not that important to the overall problem. The maximum number of RANSAC iterations has to be set at a sufficiently high value in order to produce good results in every case. In this report, this value can be set to 10,000 and produce good results. However, if we set the Euclidean distance threshold high (and thus consider a lot of good matches), then the number of iterations has to be increased in order to estimate a good transformation (setting it to 100,000 iterations is good for almost every case).

6: Warp image 2 onto image 1

In order to produce a good stitch, the estimated transformation alone does not suffice for the warping. We also have to translate the image result in order for it to fit the initial image. This is done with by defining the function warpTwoImages in the Jupyter Notebook. Thus, the final stitching can be seen. The variables result and resultcolor contain the stitching, while the variables seamless, seamlessgauss and seamlessmedian apply blurring in order to remove the seams from the stitching (those seams do not appear at every image pair, only at the cases when the image pair is not entirely parallel to each other, so a perspective transform is apparent).

7: Accuracy of Transformation

A good stitch is this:

For the image pair of the assignment description we have the following results:

thresh_dist	niterations	init_size	tolerance	Inliers	Outliers	Average Residuals Inliers	Final Euclidean Distances	Good Stitch
0,6	20000	5	1	158	0	15,37	15,37	YES
0,4	20000	5	1	62	0	4,63	4,63	YES
0,8	20000	5	1	254	1	20,87	20,87	YES
1	20000	5	1	342	16	15,62	4885,56	YES
1,2	20000	5	1	475	24	1075994,91	1077430,14	NO
0,8	20000	3	1	255	0	40,31	40,31	YES
0,8	20000	10	1	251	4	18,06	18,06	YES

0,8	20000	5	2	255	0	17,11	17,11	YES
0,8	20000	5	0,5	244	11	14,55	15,88	YES
0,8	5000	5	1	254	1	17,81	17,83	YES
0,8	1000	5	1	251	4	15,76	15,88	YES
1	1000	5	1	343	15	36,51	4856,21	SOMEWHAT (Slight Distortion
1	20000	5	5	347	11	26168,83	26582,98	NO
1,5	20000	5	0,1	997	107	477883,92	480442,56	NO
1,2	20000	5	0,1	462	37	442260,52	443206,97	NO

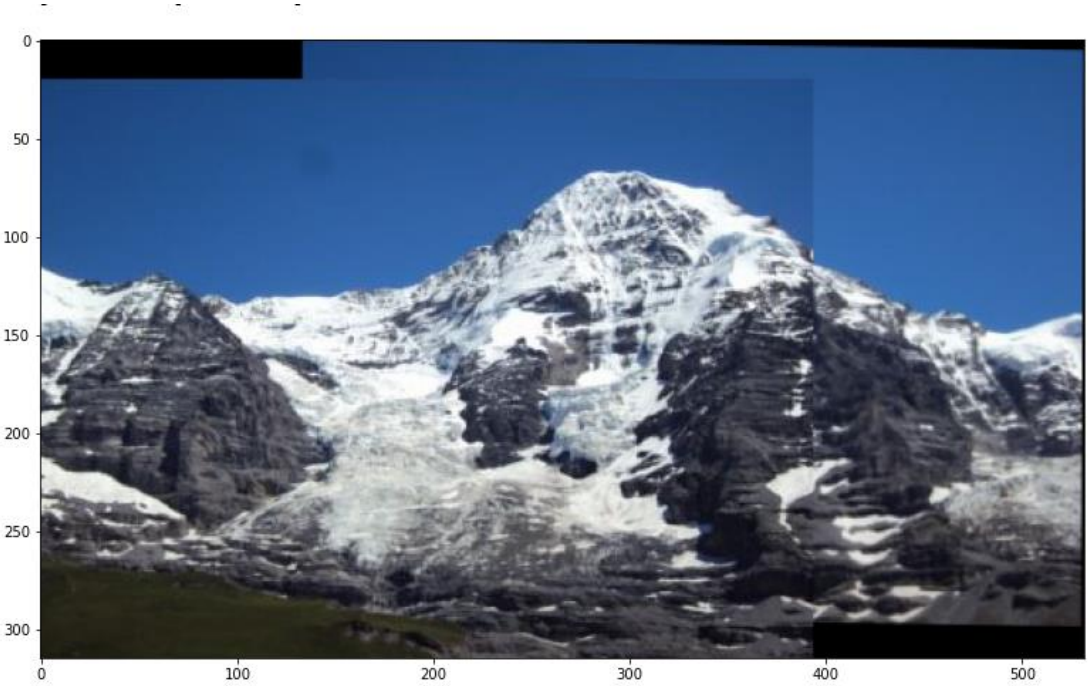
From this table we can see that overall the most important parameter in the image stitching is the threshold distance in the Euclidean distance matrix between all the SIFT keypoints. If this parameter is set high, then despite what values we set at all the other parameters, the image stitching fails. The tolerance under which we consider a point as inlier does matter, and if we set it too high (above 3) then the image stitching fails. The init size for the affine estimation does not matter, as long as it is above 3. The number of iterations should be set above 5,000 (ideally above 20,000, and even 100,000 is not excessive), to work safely in every case.

The aforementioned values are of course relative to the input images. In the next chapter, other parameter values are provided for other image pairs.

8 Experiments with Various Images

The same table will be provided for the following images:

The 2 images are to be stitched. A good stitch is this:

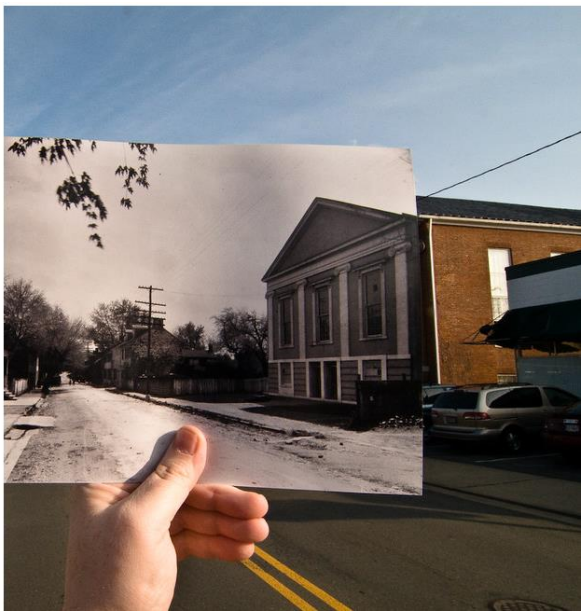


The table this time is:

thresh_dist	niterations	init_size	tolerance	Inliers	Outliers	Average Residuals Inliers	Final Euclidean Distances	Good Stitch
1,2	20000	5	0,1	501	72	23185,69	23507,36	NO
1	20000	5	0,1	445	74	84,9	89,16	YES
0,6	20000	5	0,1	277	42	19,33	20,16	YES
0,6	10000	5	0,1	267	52	44,43	45,94	YES
0,6	1000	5	0,1	248	71	32,2	33,92	YES
0,6	20000	3	0,1	311	8	133,38	133,51	YES
0,6	20000	10	0,1	246	73	7,29	8,18	YES
1,2	100000	10	0,1	456	117	3844,33	4108,67	NO
1,2	100000	5	0,1	535	38	24218,2	24418,85	NO
1	20000	3	1	514	5	139,82	143,35	YES

Again we can see that the stitching fails if the thresh_dist is above 1, despite what we set the other parameters to. The same conclusions can be made as the previous image pair.

Now we will take a look at “look into the past image stitching”. The images we will use are:



And the desired output is:

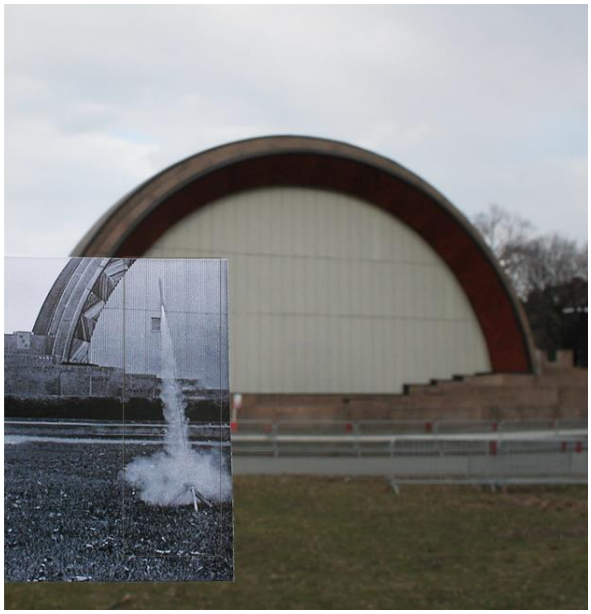
A good stitch is this:

The table this time is:

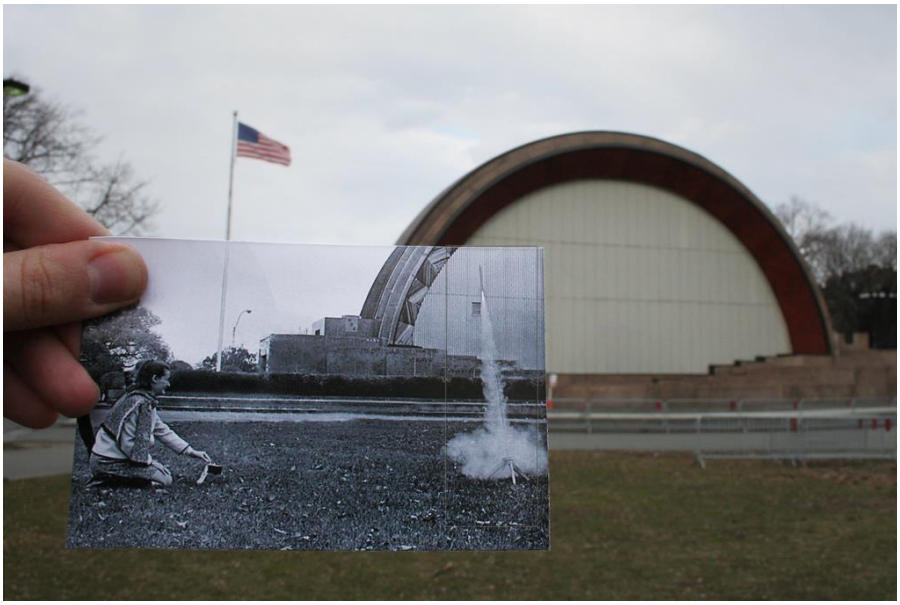
thresh_dist	niterations	init_size	tolerance	Inliers	Outliers	Average Residuals Inliers	Final Euclidean Distances	Good Stitch
1	20000	3	1	940	31	57741447,72	57742574,08	NO
0,8	20000	3	1	502	14	1978735,2	1979833,4	NO
0,6	20000	3	1	276	5	449927,76	451117,91	NO
0,4	20000	3	1	205	1	0	1041,3	YES
0,4	20000	5	1	205	1	0	1041,3	YES
0,6	100000	5	1	269	12	22563140,87	22563793,82	NO
0,6	100000	10	1	237	44	188820,56	189352,47	NO
0,4	20000	5	3	205	1	0	1041,3	YES
0,6	100000	5	0,1	262	19	314107,76	316902,43	NO
0,6	100000	5	0,01	272	9	5854451,07	5857340,91	NO
0,4	100000	10	1	205	1	0	1041,3	YES

Again we see that if we set the thresh_dist too high, then the stitching fails. This time, the maximum thresh_dist is lower than the 2 previous image pairs (it is 0.4 max, anything above will result in failed stitch).

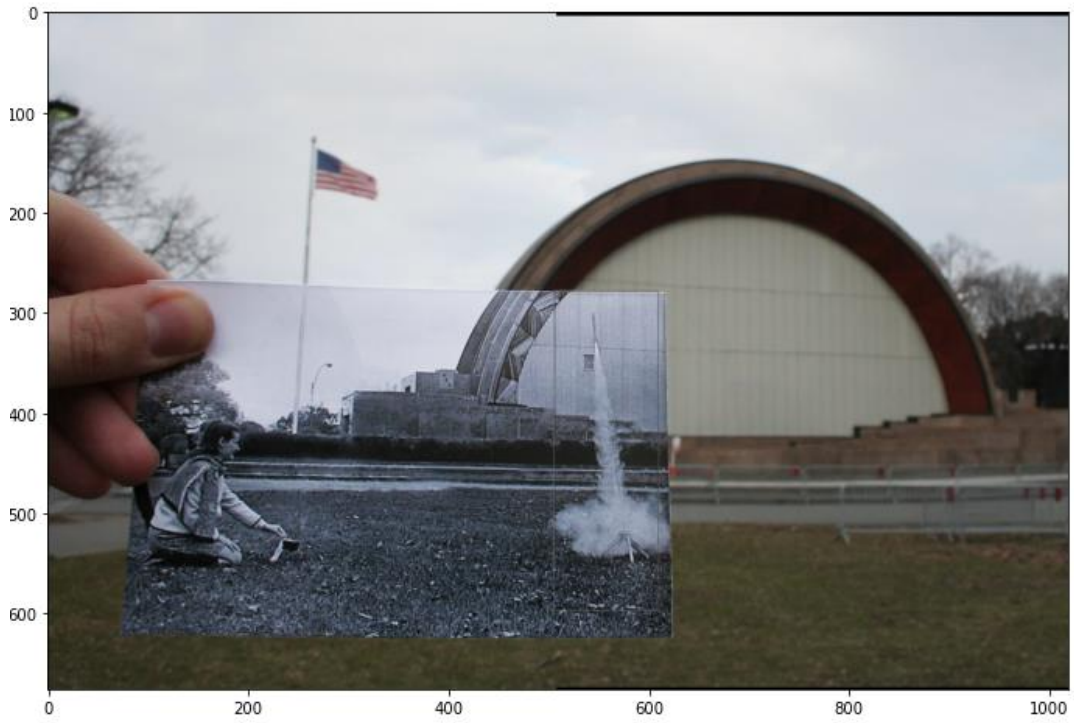
Another example is this image pair:



And the desired output is:



A good stitch is this:



The table this time is:

thresh_dist	niterations	init_size	tolerance	Inliers	Outliers	Average Residuals Inliers	Final Euclidean Distances	Good Stitch
0,4	100000	10	1	337	0	0	0	YES
0,8	20000	10	1	477	50	3659,54	9339,51	NO
0,8	100000	10	1	478	49	6472,79	10973,79	NO
0,8	100000	5	0,5	508	19	143403,08	143504,28	NO
0,6	100000	5	0,5	378	10	204323,22	204350,14	NO
0,5	100000	5	0,5	344	3	128,74	132,76	YES
0,55	100000	5	0,5	356	7	465,01	2916,84	YES
0,575	100000	5	0,5	364	11	366,31	5784,76	YES
0,59	100000	5	0,5	367	14	44632,98	44644,75	NO

We see that the thresh_dist must be at most 0.575 to get a good stitch. The other parameters' behavior is similar to those of the previous test cases.

So in conclusion we can say the following about the parameters' behavior:

The threshold distance should be set individually for every image pair, because a universal value does not suffice for every test case (however in most cases this value should be between 0.1-1.5). Regardless of the test case, this parameter should be set as much as to allow only about 100 matches (at most) from the Euclidean Distance Matrix, because as we

see in every case, too many more matches create many outliers and the image stitching is unsuccessful.

The maximum number of iterations does not have to be set very high, especially if the stitching application requires very low running time. However, if we set this value low (for example 1000) then the threshold distance should be as low as possible (for example 0.1-0.3).

The initialization size can be set to any value above 3, as this does not seem to influence the final result. However, we know from theory that this number should be close to 5, because this provides a more stable linear system for the solver.

The tolerance dictates with what error a point is considered an inlier. This value should be set close to 1 (and preferably lower than 1), as setting this value too high can influence the final result negatively (i.e. considering every point an inlier and thus producing a wrong result).

On 3 or more images, the procedure is more or less the same as when there are only 2 images. The basic difference is that we first stitch 2 images together, and then we stitch another image to the result of the first 2. This continues until we run out of images. Of course, the 2 images that are to be stitched each time need some repeated area covered, in order for the sift descriptors to work. Also, the images should not have much perspective distortion, because this cannot be represented in the Affine Transformation, and thus the stitching will be inaccurate.