Computer Vision Assignment 1 - Image Stitching

Elif Yozkan - i6219160

13th May 2024

1 Objective

Develop a panorama application by stitching together pairs of images. In this report, I will go through the steps needed to accomplish the task and present my results.

2 Step 1 : Keypoint detection

Prior to image stitching, first of all, we need to extract features from both pairs of images. To this end, we use Harris corner detection, the identify the corners in the image. After we detect the corners, we apply minimum-maximum suppression (using

```
corner_peaks()
```

) to prevent obtaining corner points that are in the same neighborhood, or overlapping corner points. The obtained corners from the first image is shown in Figure 1.

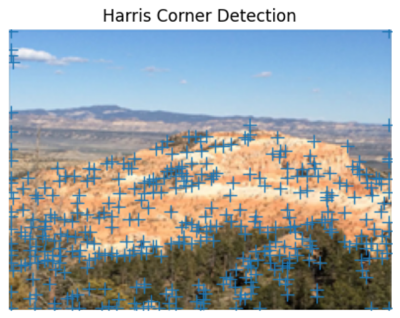

Figure 1: Detected corners

Next, we need to obtain the fixed-size patches that describe the detected corners. In order to obtain the list of patches with size n, we get square portions using the coordinates of the detected cornerpoint. Therefore, if our corner's

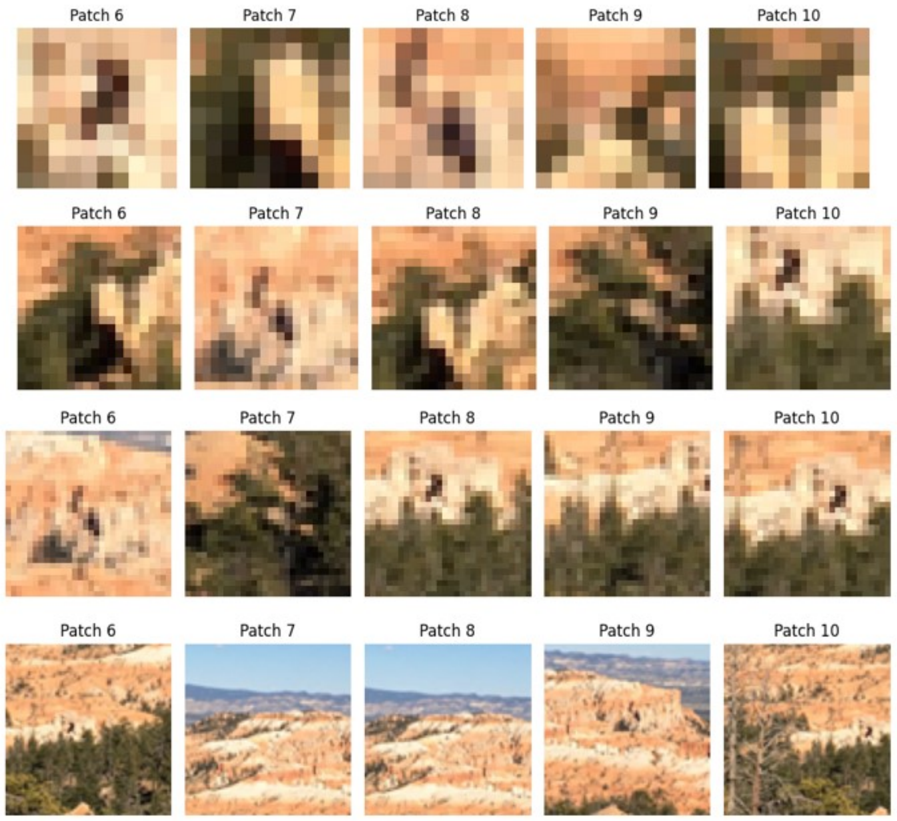

Figure 2: first 5 patches obtained from the first image using patch sizes 5,10,15 and 40 respectively.

coordinates on the image are x and y, we extend the corner coordinates to obtain the patch : $patch = image[(y-n):(y+n),(x-n):(x+n)]$. Since we perform addition and subtraction, our patch might exceed the borders of our image, so we simply discard them and do not add them to the final patch list. Patch size determines the detail that we choose to include in the description of the corner. Figure **??** illustrates the level of detail in the first 5 patches obtained using patch sizes 5,10,15 and 40 respectively. As seen, the level of detail increases as the patch size increases, however, we also generalize too much if the patch size gets too high, which can result in some non-matching keypoints to have similar descriptors.

We can also observe the impact of the patch size in the final stitched image. Figure 3 displays the final result obtained by stitching the images. The resulting stitched image when too small or large of a patch size (2 and 25) is utilized, does not seem right, as they are stretched and disoriented, due to the failure of finding correct matches.

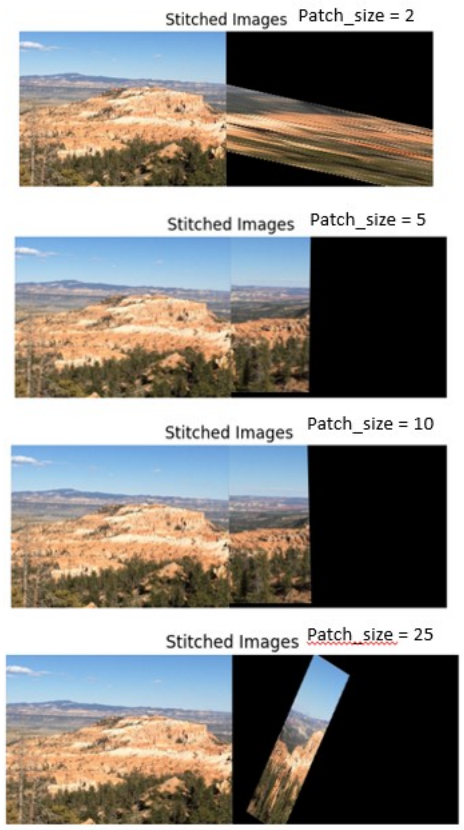

Figure 3: Stitched images, using descriptions with different patch sizes

3 Step 2 : SIFT

We perform, Scale-invariant feature transformation to the corners we detect using the Harris corner detection, in case two images to be stitched are in different scales. We iterate over all the keypoints extracted from the Harris corner detector and then apply the transformation using

```
cv2.sift.detectAndCompute()
```

which gives us the transformed descriptions of the corners. Similar to the patch extraction, we ignore the patches which exceed the borders of the image. Patch size behaves similarly to the Harris Corner Detection, as the increasing number of patch size increases the level of detail. (You can see the code for patch experiment with SIFT, it is not added here to prevent repetition.)

4 Step 3 : Distance Calculation

To find the matches between the keypoints, we need to measure how similar the keypoints are. Thus, we compute the distances between the flattened version of the feature points, as we compute the distances on vectors. The choice of distance metric in the implementation is controlled by the boolean variable *euclidean = True*. The distances are computed as follows :

1. Normalize the description matrices using Min-Max scaling : $x' = \frac{x - x_{min}}{x_{max} - x_{min}}$
2. Then for all possible pairs of keypoints compute the euclidean distance : using

```
np.linalg.norm(keypoint1,keypoint2)
```

3. Alternatively, for all possible pairs of keypoints, compute the normalized correlation : $\frac{keypoint1}{|keypoint1|} \times \frac{keypoint2}{|keypoint2|}$

5 Step 4 : Finding Best Matches

Stitching the images are more efficient and accurate if we only consider the best or relevant matches to estimate the affine transformation matrix. In this respect, in the next step we filter the keypoint pairs that have a better matching than other possible matches, using the distance metric. The selection is either based on a threshold, where we select matching pairs which have distance less than the defined threshold, or we sort the distances of the pairs in ascending order and select the top k matches with smallest distance. The resulting matches with varying top k values and varying thresholds are shown in Figures 4 and 5 respectively.

The choice of the parameter threshold or top k plays a significant role on the process of image stitching. If we choose too large value for both variables, we might end up having matches that are not very similar to each other, thus it might lead to incorrect estimations. On the other hand, if we select threshold or top k to be very low, then we might not have enough keypoint matches to compute the correct transformation matrix. The final resulting stitched image tested with varying choices of top k parameters is shown in Figure 6.

We observe that increasing number of matches result in poorer performance in image stitching. This is probably due to as well as better matching keypoint pairs, some irrelevant keypoints were included in the affine matrix transformation estimation.

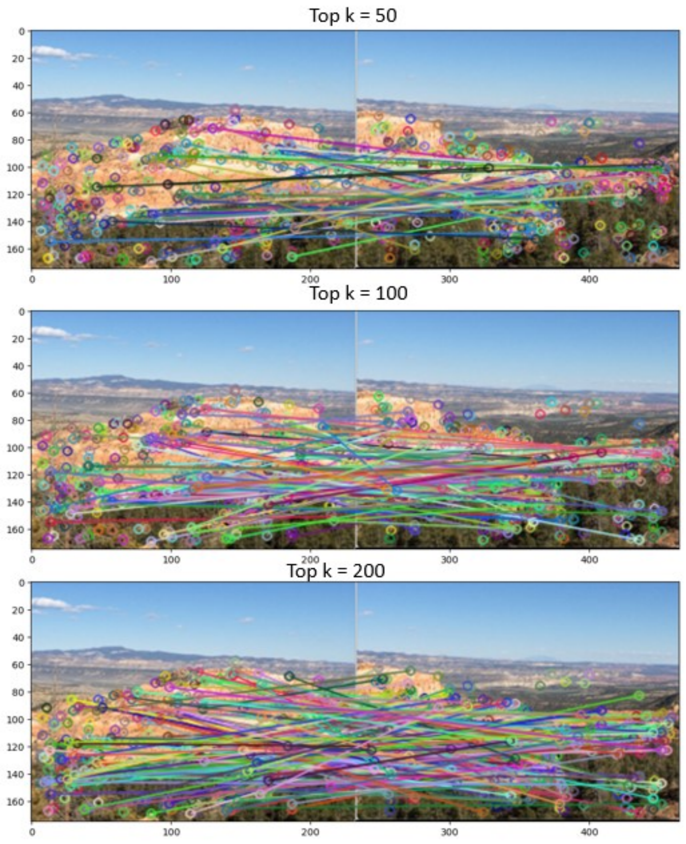

Figure 4: Best matches obtained with top k value of 50, 100 and 200.

6 Step 5 : Affine Transformation Matrix Estimation Using RANSAC

One of the most crucial steps is to estimate the affine transformation matrix to warp one image onto another to obtain the final stitched image. For this purpose, we utilize the RANSAC algorithm. My implementation steps of the RANSAC algorithm is as follows :

For each iteration of the algorithm (max iteration = 1000) :

1. Sampling : We first randomly sample s pairs and mark them as inliers.
2. Fit the model : Fit the least squares model based on the s selected inliers.
3. Find inliers: Iterate over the remaining source points not marked as an inlier, find other candidate inliers by comparing the L2 Norm between the predicted destination keypoint using the estimated affine matrix ,obtained

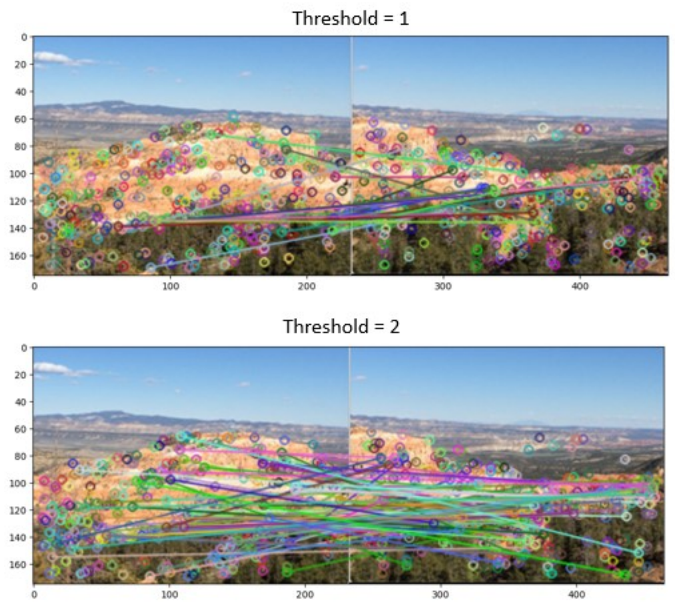

Figure 5: Best matches obtained with threshold values of 1 and 2.

   from the model, and the actual destination keypoint. If the distance is below 1, mark the pair as an inlier.

4. Update : Update the estimated affine transformation matrix and the inliers.

5. Return the affine transformation after the number of iterations has been reached

One of the experiments of RANSAC procedure included executing the algorithm with varying values (3,4,5) of initial sample size s. In Table 1, we see the resulting number of found inliers and the outliers. We see that the highest number of inliers are obtained when $s = 5$, and the lowest with $s = 4$. Naturally, it is understandable that with higher value of s, it is more likely to capture higher number of inliers however, the case when $s = 4$ does not fit into this idea. I think it might be because of the randomness factor in RANSAC.

	s = 3	s = 4	s = 5
no. inliers	45	23	48
no. outliers	55	77	52

Table 1: RANSAC Affine Transformation Matrix Estimation Results

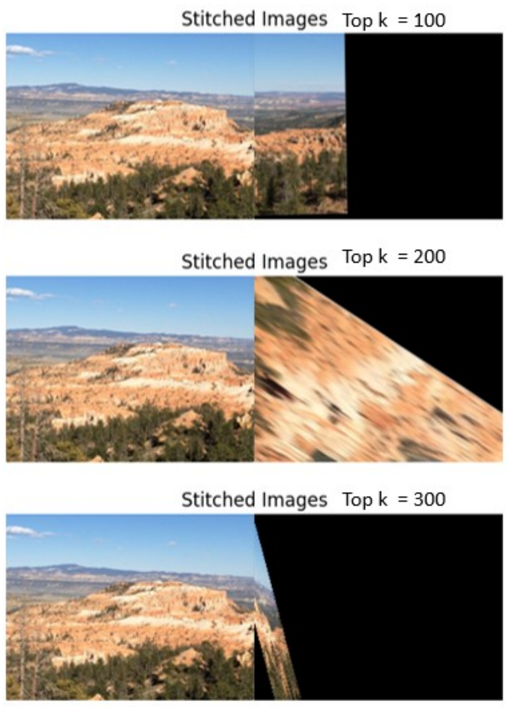
Stitched Images Top k = 100
Stitched Images Top k = 200
Stitched Images Top k = 300

Figure 6: Resulting stitched image with varying top k values for best match selection

The results obtained from the RANSAC procedure with varying values of s, are also projected onto the resulting final stitched image. Figure 7 shows the resulting panorama with differing s. We see that when $s = 4$ the stitched part looks slightly more tilted and we see relatively better result when $s = 5$.

7 Step 6 : Warping and Stitching

Once we have obtained the affine transformation matrix using the RANSAC procedure, the next step is to warp the second image onto the first one to finalize the stitching process. In this step, I could not achieve the correct result with *cv2.WarpPerspective()*, so I utilized *cv2.WarpAffine()* instead. I basically warp image 2 using the affine matrix, resize the image to match the shape of the first image then horizontally stack them together.

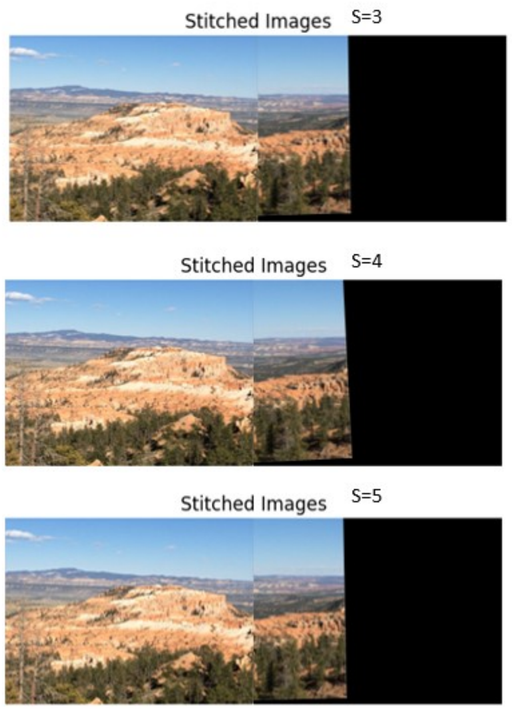

Figure 7: Final stitched image when RANSAC procedure is executed with differing initial sample points (s= 3,4,5)

8 Step 8 : Testing the Pipeline with Different Images

The developed pipeline was tested on three additional images, the results are shown I figures 8, 9 and 10. I realized that the ”middle part” in the images get lost, as they shrink in size, leading to minor mismatches and loss of information. I think it might be related to the computation of the affine transformation matrix.

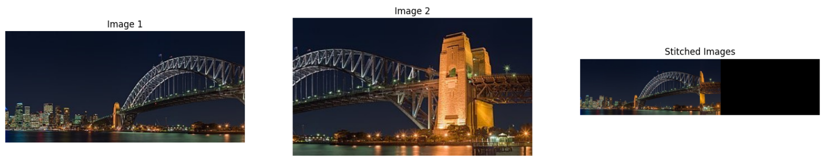

Figure 8: Sydney Bridge - Stitching

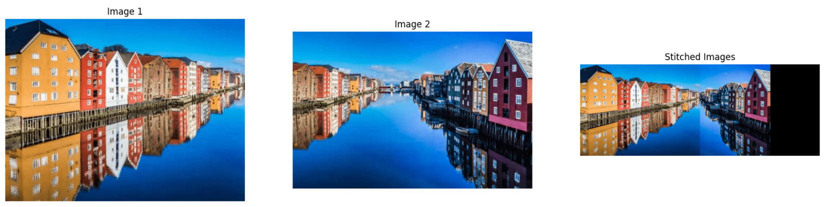

Figure 9: Scandinavian Houses - Stitching

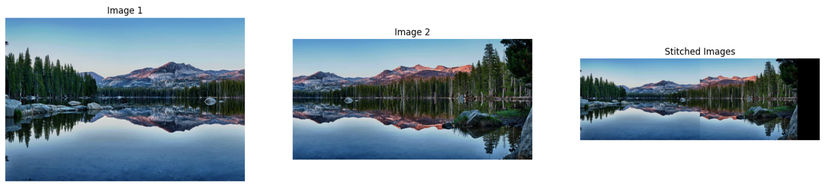

Figure 10: Forest - Stitching