Image Stitching
Computer Vision (KEN4255)

Chinmay Rao (i6218054)

May 7, 2020

1 Introduction

Image stitching involves seamlessly combining multiple images into a single panoramic image by using points of interest to estimate a transformation mapping coordinates from one image to another such that the larger structure is preserved. This work deals with stitching together a pair of images by applying the RANSAC estimator to obtain an affine transformation. Section 2 discusses the approach in detail and highlights the specifics of its implementation. Section 3 records the experiments performed to assess the performance of the implemented RANSAC, including a sensitivity analysis. In section 4, a qualitative analysis of the method is performed to test its robustness on different test cases including a hard case.

2 Methodology

It is worth mentioning that the problem of image stitching in this work is limited to two images that are captured with only a shift in camera position and no camera rotation. Under this constraint, an affine transformation is to be obtained without any need for perspective warping. As for the implementation in program, all the code is written in Python 3 relying heavily on *Numpy* for numerical computation and *Matplotlib* for visualization. The OpenCV library is also used although specifically for image import and colour conversion, for Harris corner detection and for computing SIFT descriptors.

2.1 Feature Detection

The first step in the image stitching pipeline is feature detection. The Harris corner detection method is used here to extract key-points from an image. This is implemented using the OpenCV function *cornerHarris()* with a displacement block of size 2, Sobel window of size 3x3 and the k parameter value 0.04. The map thus obtained is passed through a threshold, and keypoint coordinates are extracted using the OpenCV function *connectedComponentsWithStats()*.

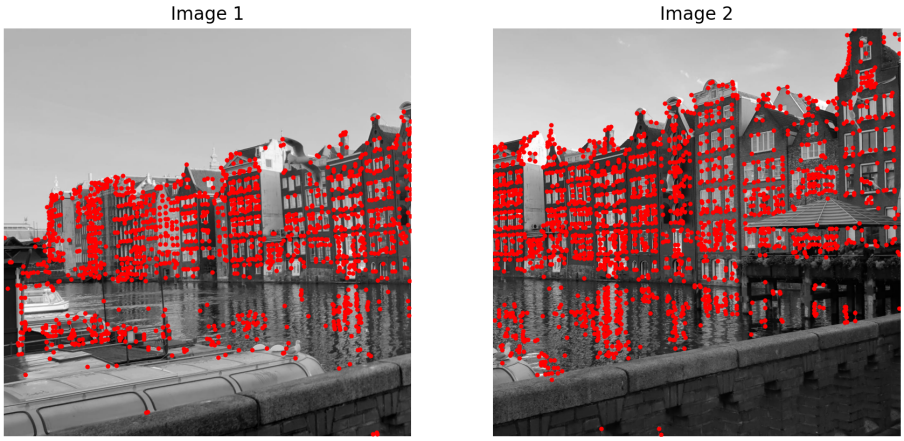


Figure 1: Extracted Harris corners

This operation is performed on both images resulting in two separate sets of keypoints as shown in figure 1.

2.2 Feature Description

Once the keypoints from both images are extracted, they need to be described in a vector form that captures the local pixel neighbourhood information. In this work, two types of descriptors are used separately. The first descriptor uses a patch of RGB values around the keypoint given a fixed patch size. For example, for a patch size of 9, the neighbourhood is of shape [9,9,3], which is then flattened into a vector of size 243. This descriptor is implemented entirely in the code. The second descriptor used here is SIFT. Using the standard OpenCV SIFT descriptor, a vector description of size 128 is obtained for each Harris keypoint. In the code, SIFT descriptors are computed using the *compute()* method of the OpenCV class *cv2.xfeatures2d.SIFT_create()*.

The two descriptors are not compared with each other directly, but are rather used to assess RANSAC separately.

2.3 Finding Correspondence

The feature descriptions for both sets of keypoints obtained after the previous step are then used to find correspondence. This is based on the fact that since the descriptor vector encodes the local neighbourhood information of a given keypoint, a pair of keypoints belonging to the same structure in the two images will have similar descriptions.

This feature matching is implemented as follows. First, each feature descriptor vector is normalized to unit length as shown in equation 1.

$$x^i_{norm} = \frac{x^i}{||x^i||_2} \tag{1}$$

where x^i is the descriptor vector for i^{th} keypoint, and $||x^i||_2$ is its $l2$-norm. The descriptors from the first image can now be matched with those from the second image. This can be achieved in two different ways: using correlation or using Euclidean distance. Correlation between two normalized descriptors is computed as shown in equation 2.

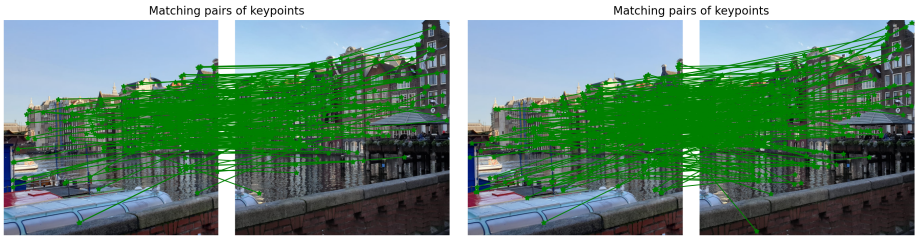
$$corr(x^i_{norm}, y^j_{norm}) = x^{i\,T}_{norm}.y^j_{norm} \tag{2}$$

where x^i_{norm} is the normalized descriptor vector for i^{th} keypoint form first image and y^j_{norm} is the same for j^{th} keypoint from the second image.
A matching pair of normalized descriptors will have high correlation and low Euclidean distance between them. Using either of the two measures, a "similarity matrix" is constructed where the rows represent the indices of keypoint descriptors from first image and columns represent those from the second image. Depending on the choice of the similarity measure, the values of this matrix are either euclidean distances or the correlation values between the (normalized) keypoint descriptors of corresponding indices.

```
def get_matchings(similarity_matrix, threshold):
    # Initializes the list of correspondences
    matching_pair_indices = []

    for i in range(similarity_matrix.shape[0]):
        # Find best match for i
        match_for_i = np.argmax(similarity_matrix[i, :])

        # Check if the vice versa is also true
        if i == np.argmax(similarity_matrix[:, match_for_i]):

            # Apply threshold
            if similarity_matrix[i, match_for_i] >= threshold:

            matching_pair_indices.append(tuple(i, match_for_i))
            similarity_matrix[:, match_for_i] = -99

    return np.array(matching_pair_indices)
```

Listing 1: Function for obtaining correspondence using correlation as the measure

Listing 1 shows the matching function that uses correlation as the similarity measure. It is an implementation of two-way nearest-neighbour matching without replacement. Given the similarity matrix, it checks each keypoint descriptor from first image for the best matching descriptor from second image. It then checks if the vice versa also holds (i.e. two-way correspondence), and then applies the threshold. If successful, this pairing is stored into memory and its constituent keypoint indices are not used further in the loop.



(a) When using RGB patch descriptor **(b)** When using OpenCV's SIFT descriptor

Figure 2: Matching pairs of keypoints

Figure 2a shows the matches obtained when the RGB type descriptor is used and figure 2b shows the matches for OpenCV's SIFT descriptor in base case configurations that shall be mentioned later in section 3. Although both correlation-based and distance-based matching methods are implemented completely in the code, only the correlation is used as a similarity measure in the experiments and analysis.

2.4 RANSAC

The RANSAC algorithm is implemented completely in the code, as required. It takes four parameters: the *sample size* that defines the number of random matching pairs to be sampled every iteration, the *number of iterations*, the *tolerance* that defines the maximum allowable error for a matching pair to be considered an inlier, and finally, the *inlier threshold* that is used to classify a model as good or bad based on the number of its inliers. The number of iterations is not optimized based on any given inlier probability, but rather is set to be sufficiently high. The iterative process terminates either if the specified cycles of iterations are completed or the sample space is exhausted. Model fitting is performed using the least-squares method and is applied in the code using the Numpy library function *linalg.lstsq()*. The RANSAC estimation function maintains a list of good candidate models along with their corresponding average inlier residuals and inlier indices. It then returns as the best model, i.e. the one with least residual value.

Figure 3 shows the result of RANSAC with the base case configuration using RGB patch descriptor. The blue points and lines represent the inliers.

2.5 Warping and Stitching

The output of the RANSAC algorithm is an 3x3 affine transformation matrix. The OpenCV library function *warpPerspective()* is used here for warping the second image using this transformation.

Figure 4 shows the original first image and the warped second image, both with some extra zero-valued region (in black) allocated before stitching them

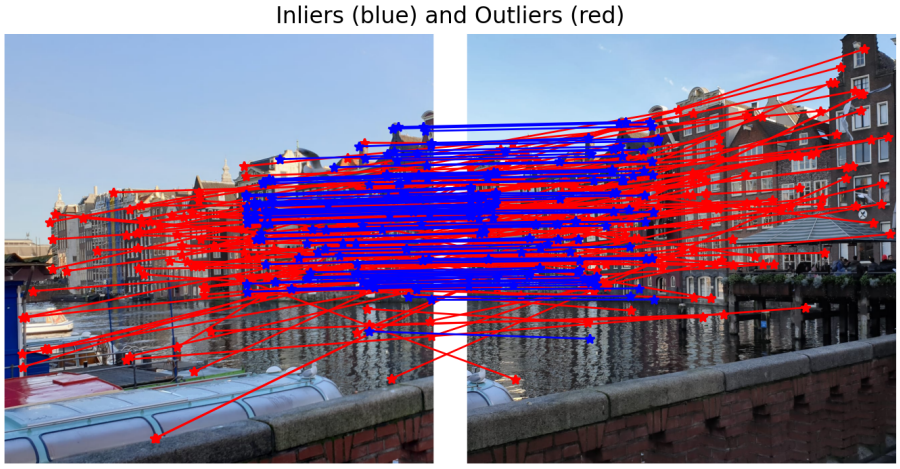


Figure 3: Inliers and outliers (when using RGB descriptor)

together. The final stitched image in the end is obtained by taking all the non-zero valued pixels in the warped image and writing them onto the first image.

2.6 Metrics

To quantify RANSAC's performance for a given pair of images and a given set of parameter values, four different measures are defined and recorded, and are enlisted below.

1. Number of inliers: Total number of the matching keypoint pairs that lie within the tolerance region of the model
2. Number of outliers: All the remaining pairs of matching keypoints
3. Average residual for inliers: Calculated before refitting the model on inliers
4. Average Euclidean distance: Average of distance between the ground truth (i.e. keypoint locations in the first image) and the hypotheses of the final refitted model (i.e. locations of corresponding keypoints from second image after warping)

The number of inliers and outliers are not used directly for some of the analysis, but rather the ratio between them is used.

3 Experiments and Analysis

As mentioned earlier in section 2, two descriptors - RGB patch and OpenCV SIFT - are used separately to assess the RANSAC implementation. Hence, for

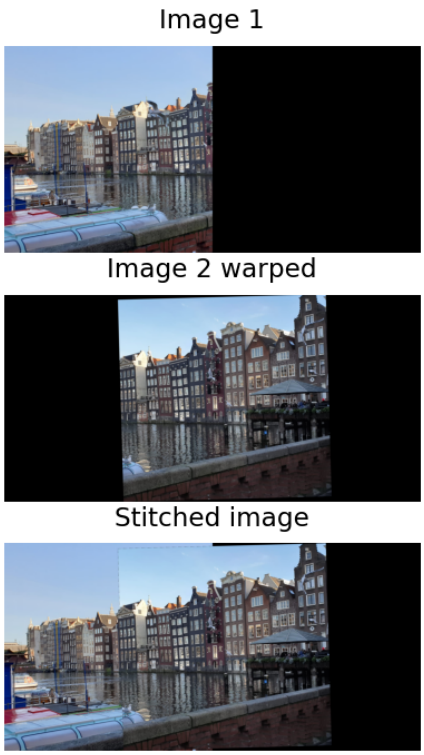


Figure 4: Stitching result

performing the experiments, two separate base cases are established, one for each descriptor. Although the sensitivity experiments in 3.2, 3.3, 3.4 and 3.5 are conducted using both the descriptors separately, only the ones pertaining to the RGB path descriptor are mentioned in subsections 3.3, 3.4 and 3.5 for the sake of brevity of the report. However, the results for both sets of experiments can be generated using the supplied code.
The base case configuration when using the RGB patch descriptor is as follows:

- Patch size = 11
- Correlation threshold for matching = 0.980
- RANSAC's sample size = 3
- RANSAC's tolerance = 40
- RANSAC's inlier threshold = 15

Base case configuration when using OpenCV SIFT descriptor is as follows:

- Correlation threshold for matching = 0.950
- RANSAC's sample size = 3
- RANSAC's tolerance = 40

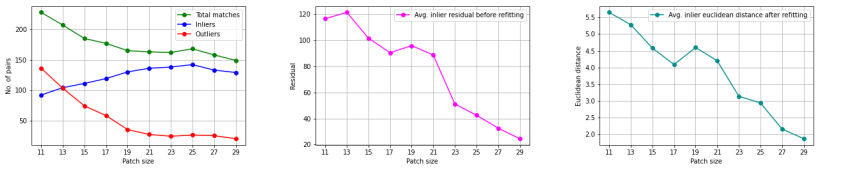
- RANSAC's inlier threshold = 15

The number of iterations for RANSAC is set to a fixed value 1000 in all experiments and is not used for sensitivity analysis. To ensure the consistency of results across multiple runs of the code, a fixed random seed for Numpy's random generator is set at the beginning of the program.

3.1 Effect of Descriptor Patch Size

Figure 5 shows the effect of increasing the patch size of the RGB descriptor on the RANSAC metrics. Patches of sizes 11, 13, 15, 17, 19, 21, 23, 25, 27 and 29 were considered for this test. As observed in figure 5a, increase in patch size results in a greater number of inliers, while the number of outliers decrease. This can be explained by the fact that greater patch sizes encode information from bigger neighbourhoods thereby providing the description with more context. This would result in higher quality matches, increasing the proportion of inliers. The total number of keypoint matches, however, appear to decrease, probably because with increase in patch size, the resulting vectors of greater complexity need to be compared given a fixed correlation threshold.

The average residual and the final average euclidean distances for the inliers follow a decreasing trend as the patch size increases. The reason for this seems to be the one mentioned earlier, that more local information gets used resulting in higher number of inliers.



(a) On matches,inliers and outliers **(b)** On average residual **(c)** On average euclidean distance

Figure 5: Effect of RGB descriptor's patch size on RANSAC metrics

Equation 3 is used express the sensitivity of RANSAC's average inlier euclidean distance to the RGB descriptor's patch size. This equation, however, is generalized to compute sensitivity of the inlier to outlier ratio and average inlier residual as well. The exact values are shown in table 1.

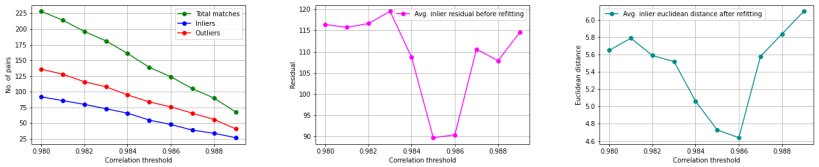
$$sensitivity = \frac{\%change\ in\ avg\ euclidean\ dist}{\%change\ in\ patch\ size} \tag{3}$$

Inlier-outlier ratio	Avg. residual	Avg. Euc. distance
2.48	-1.60	-1.14

Table 1: Sensitivity of the metrics to RGB descriptor's patch size

3.2 Effect of correlation threshold

Figures 6 and 7 show the effects of increasing the correlation threshold value when using the RGB descriptor and OpenCV SIFT descriptor respectively.



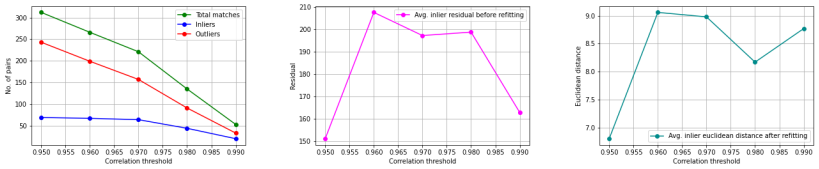
(a) On matches,inliers and outliers **(b)** On average residual **(c)** On average euclidean distance

Figure 6: Effect of correlation threshold on RANSAC metrics when using RGB descriptor

Correlation threshold	0.98	0.981	0.982	0.983	0.984	0.985	0.986	0.987	0.988	0.989
Inlier-outlier ratio	0.676	0.671	0.689	0.675	0.694	0.654	0.631	0.590	0.607	0.658

Table 2: Inlier to outlier ratio for different threshold values when using RGB descriptor

In both cases, the number of inliers, outliers and the total matches drop gradually due to the increased constraint that a higher threshold imposes for successful matching. The inlier to outlier ratio in the RGB descriptor case remains relatively constant as seen in table 2, whereas in the SIFT descriptor's case, this value seems to increase but not significantly.



(a) On matches,inliers and outliers **(b)** On average residual **(c)** On average euclidean distance

Figure 7: Effect of correlation threshold on RANSAC metrics when using OpenCV SIFT descriptor

Correlation threshold	0.950	0.960	0.970	0.980	0.990
Inlier-outlier ratio	0.283	0.336	0.487	0.483	0.606

Table 3: Inlier to outlier ratio for different threshold values when using OpenCV SIFT descriptor

The trend in average inlier residual value and average inlier euclidean distance is mixed in both case, with no consistent increase or decrease. Using

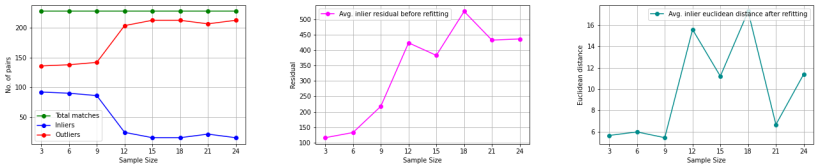
these results, the sensitivity of the RANSAC metrics to the matching threshold is given in table 4.

	Inlier-outlier ratio	Avg. residual	Avg. Euc. distance
RGB	-1.90	3.32	11.44
SIFT	20.19	3.4	7.29

Table 4: Sensitivity of the metrics to the matching threshold for both descriptors

3.3 Effect of RANSAC's Sample Size

The plots for the RANSAC metrics with respect to the sampling size are shown in figure 8. The number of inliers appear to drop as the sample size increases. This may have been caused due to a large initial number of bad matches which would result in samples with outliers. For a small sample size, it is possible to obtain samples containing only good matching pairs that result in good models. As the sample size increased, so did the number of bad pairs per sample making the initial model fitting sub-optimal affecting the final refitting. The average inlier residual (computed before refitting), hence, also increases as seen in the figure. However, the euclidean distance shows neither an increasing nor decreasing trend.



(a) On matches,inliers and outliers **(b)** On average residual **(c)** On average euclidean distance

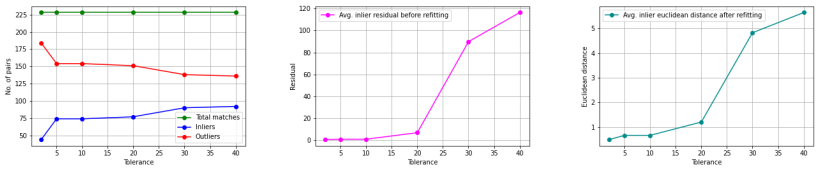
Figure 8: Effect of RANSAC's sample size when using RGB patch descriptor

3.4 Effect of RANSAC's Tolerance

Increasing RANSAC's tolerance means making the criteria for a matching pair to be considered an inlier less strict. This naturally results in an increasing number of inliers as seen in figure 9a. However, expanding the tolerance region can increase the possibility of the model fitting sub-optimally as more of the bad matching pairs are classified as inliers. Hence, despite a higher number of inliers, the average residual error of the model increases, and so does the average euclidean distance as observed in figures 9b and 9c.

3.5 Effect of RANSAC's Inlier Threshold Number (T)

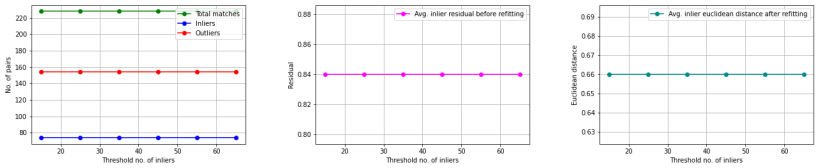
Increasing the inlier number threshold in RANSAC, as seen in figure 10, didn't result in any changes in the metric values. This is due to the existence of an



(a) On matches,inliers and outliers **(b)** On average residual **(c)** On average euclidean distance

Figure 9: Effect of RANSAC's tolerance when using RGB patch descriptor

already high number of inliers for the best model, i.e. the model with least average residual. By increasing the threshold number, the sub-optimal models no longer get considered as candidate models. At a certain point when just one candidate model exists, further increasing the the threshold only results in an empty list of candidates.



(a) On matches,inliers and outliers **(b)** On average residual **(c)** On average euclidean distance

Figure 10: Effect of RANSAC's inlier number threhold when using RGB patch descriptor

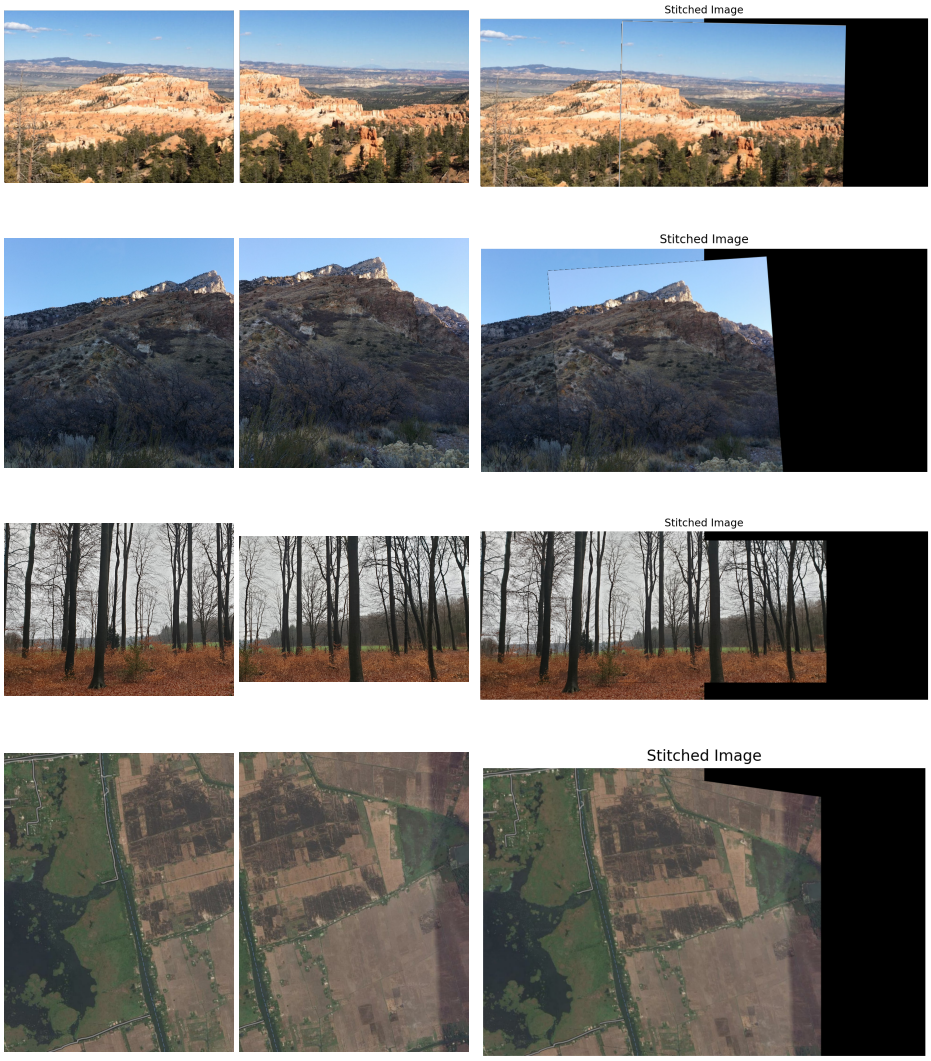
4 Qualitative Results

In addition to the experiments and sensitivity analysis performed to quantitatively evaluate the implemented RANSAC estimator, it was also applied on different test images to assess its performance qualitatively.

4.1 Simple Test Cases

A number of simple test cases were considered, including the standard test pairs, a pair of original photographs and finally a pair of satellite images. All of these test examples are considered "simple" due to a high degree of similarity among each pair.

Figure 11 shows the four test examples along with their corresponding outputs. All of these stitched panorama images were obtained using the RGB descriptor in the base case configuration. The warping in the first case includes slight shear in addition to translation. In the second case, the predominant components of the warping are scaling and rotation, while only a simple trans-



(a) Image 1 **(b)** Image 2 **(c)** Stitched panorama

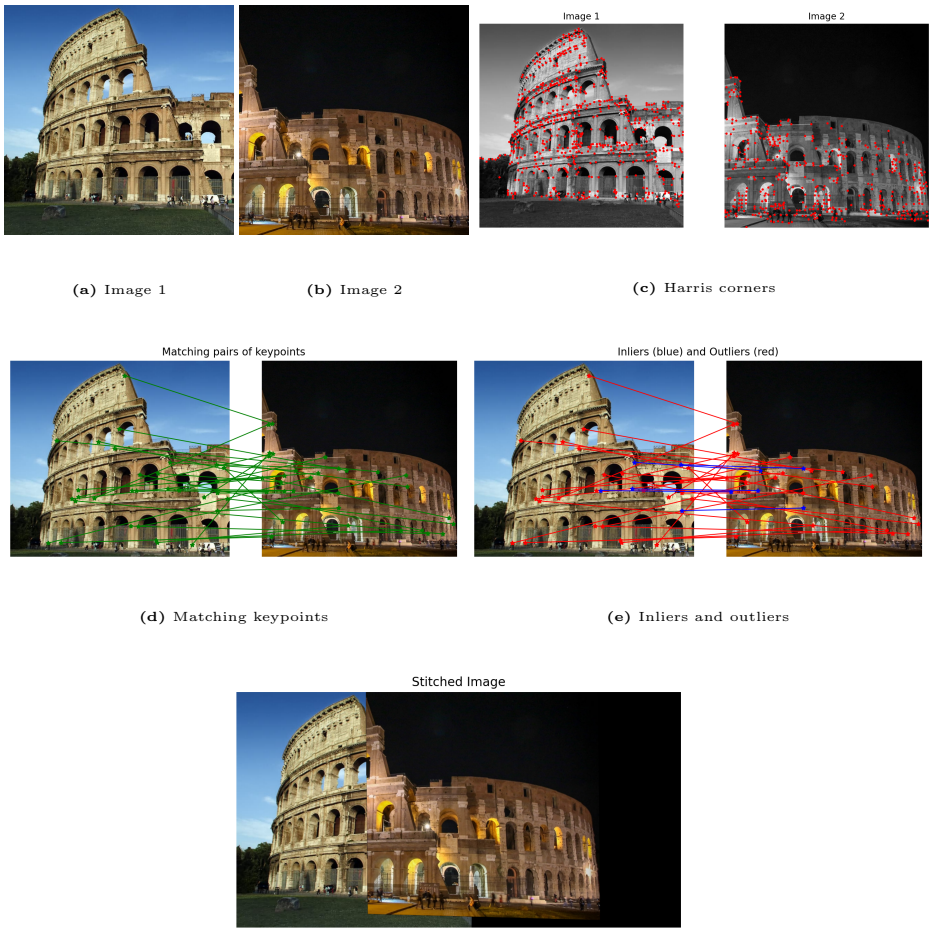
Figure 11: Results for simple test cases

lation is observed in the third case. The fourth test case, similar to the first one, displays shear in the warping.

4.2 Hard Test Case

To further test the robustness of the RANSAC implementation, a "hard" test case was considered. Here, the input pairs are pictures of the Colosseum, one of which was captured during the daytime and the other at night, and both

possibly at different times of the year. This is clearly a hard case because of multiple reasons. First, the background is completely different. The levels of illumination differ too making the texture and details of the Colosseum in the daytime image appear relatively sharp. The regions around the arches are difficult to match because of the presence of shadows in the first image and artificial yellow lighting in the second. Moreover, the bottom part of the images have different numbers and positions of people and other objects making feature matching even more challenging.



(a) Image 1 **(b)** Image 2 **(c)** Harris corners

(d) Matching keypoints **(e)** Inliers and outliers

(f) Stitched panorama

Figure 12: Results from all stages of image stitching for the hard test case

Figure 12 shows the test images as well as outputs from different stages of the image stitching pipeline. This result was obtained using the RGB descriptor with patch size 21, a correlation matching threshold 0.94, RANSAC sample size of 3, a tolerance of 50 and the inlier threshold value as 5. This produced a

relatively few matching pairs with merely 5 inliers and 26 outliers. However, these inliers were clearly sufficient to (almost) accurately warp the image. The final average inlier euclidean distance obtained was 4.098.

This approach worked successfully due to several reasons. First, due to a large similarity in the color of the Colosseum in both images despite all the differences, the RGB descriptors were sufficient to produce at least few good matches. Secondly, with trial and error, the correlation threshold was set such that there were not many matches (most of which would have been bad otherwise) and at the same time enough number of them to feed into the RANSAC estimator.

5 Conclusion

An image-stitching application was successfully built with a full implementation of a simple RGB feature descriptor, nearest neighbour matching and the RANSAC algorithm. Systematic experimentation was performed for sensitivity analysis of the RANSAC implementation. A qualitative evaluation was also performed to test its robustness on multiple test cases. Finally, the implemented method was tested on a hard test case on which it was shown to be very successful.