EN2550 - Fundamentals of Image Processing and Machine Vision

Assignment II

Fitting and Alignment

Name: Sumanasekara W.K.G.G.
Index: 190610E

Note: All the files relevant to this assignment can be found in `https://github.com/GevinduGanganath/EN2550/tree/main/Assignment%202`

Question 1

In the first part of the assignment, a circle is estimated using the RANSAC algorithm. Objective is to write a code for RANSAC manually. Figure 1 shows the python code written for the RANSAC algorithm.

```python
def RANSAC_Circle(data_points):
    thres = np.std(data_points)/16  # threshold for RANSAC
    num_iterations = np.log(1 - 0.95)/np.log(1 - (1 - 0.5)**3)
    iterations_done, max_inlier_count, selected_model = 0, 0, None

    while iterations_done < num_iterations:
        iterations_done += 1
        np.random.shuffle(data_points)  # randomly selecting 3 data points
        sample_data = data_points[:3]
        xc,yc,radius,_ = cf.least_squares_circle((sample_data)) # estimating a circle with selected data points
        center = (xc, yc)
        error = np.abs(radius - np.sqrt(np.sum((center - data_points[3:])**2, axis=1))) # computing error of remaining data points
        inliers = error <= thres # camparing the error with threshold
        inlier_count = np.count_nonzero(inliers) # number of inliers
        if inlier_count > max_inlier_count: # selecting the best model
            max_inlier_count = inlier_count
            inlier_points = []
            for index, inlier in enumerate(inliers): # filtering the inlier points
                if inlier == True:
                    inlier_points.append(data_points[3:][index])
            inlier_points = np.array(inlier_points)
            selected_model = (center, radius, data_points[:3], inlier_points)

    # refitting with all inliers
    xc,yc,radius,_ = cf.least_squares_circle(np.concatenate((selected_model[2], selected_model[3]), axis=0))
    best_model = ((xc, yc), radius, selected_model[2], selected_model[3])
    return best_model
```

Figure 1: Code snip of RANSAC algorithm

RANSAC parameters were set as follows:

- Threshold: Standard deviation of data points divided by 16 (selected by trial and error)
- Number of samples (s): 3
- Probability (p): 0.95
- Outlier ratio (e): 0.5
- Number of iterations $= \frac{log(1-p)}{log(1-(1-e)^s)} \approx 22$

Data points are shuffled at each iteration and the first three data points were selected to estimate the circle. *least squares circle* function of the *circle fit* python library was used to compute the center and radius. Then the distance between the center and the remaining data points are calculated. By comparing those values with the radius $\mp$ threshold we can count the number of inliers. The iteration which gives the highest number of inliers was selected and using those inliers best model can be calculated.
Figure 2 shows the results. Circle obtained by RANSAC algorithm (red) is giving better results than the best samples (green).

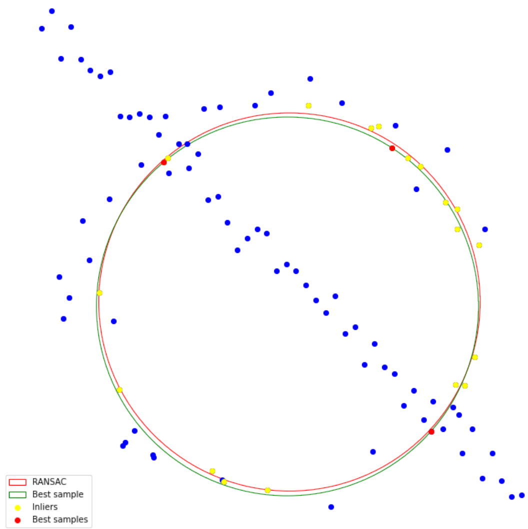

Figure 2: Question 1 results

Question 2

In this question the expectation is to superimpose a flag to an architectural image. The procedure for this is to first take four ponts on the architectural image to compute the homography which maps the flag to architectural image. Then we can wrap the flag and blend it with architectural image.

Figure 3 shows the results of the superimposing. The left column shows the point selections while the right column shows the flag superimposed into the selected locations (Image quality has been reduced slightly due to the resizing in latex)

Question 3

(a) In the first part of question 3, it is required to compute and match SIFT features between Graffiti image 1 and 5. We can directly occupy openCV built-in functions for this purpose. First we have to find the keypoints and descriptors of each image. Then we can match these descriptors and filter them based on the distance to obtain the best results. Figure 4 shows the results of SIFT feature matching.

(b) To calculate the homography using the RANSAC algorithm, first we shuffle the point set and then select four point correspondences. After that we can compute the homography for those points. Now we can wrap an image with computed homography and compare it with other image based on the distances. By this method the inliers can be filtered and finally we recompute the homography using all inliers. [1]

The accuracy of this method it highly depends on the feature matching. Feature matching between first and second image of Graffiti image set has a good accuracy, as a result we can obtain a similar homography to the given. since the feature matching between first and fifth image is not that much accurate, homography is also deviate from the given.

(c) Graffiti image 1 was stitched into the fifth one. Initially, image 1 was wrapped using the given homography. Then we can blend it with the fifth image. However, while

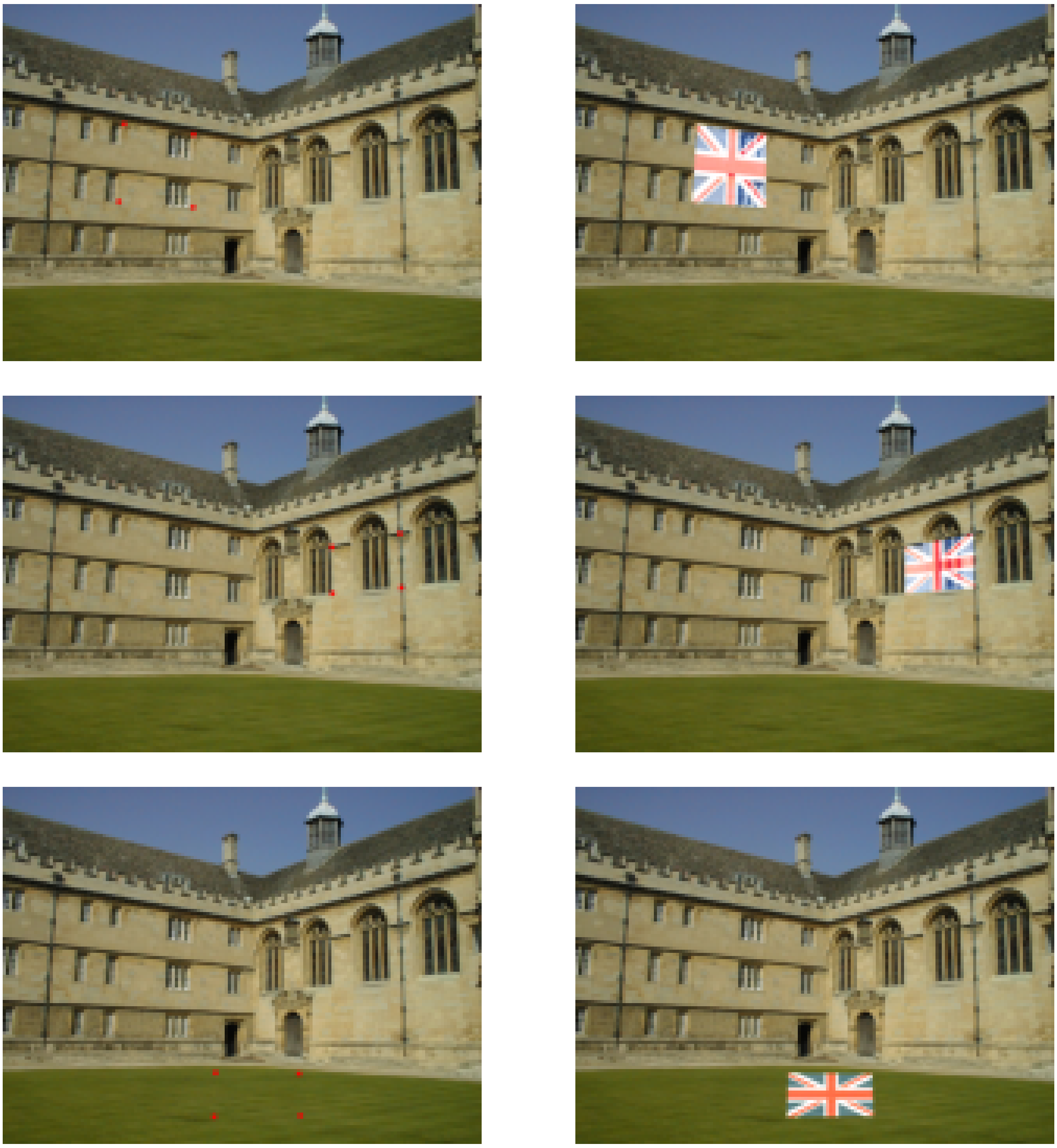

Figure 3: Question 2 results

Figure 4: SIFT feature matching

Figure 5: Image stitching

this blending the brightness of the stitched area was increased drastically and gave an unexpected appearance. Therefore, the brightness of the stitching area of image 5 was reduced. Result is shown in figure 5. In the stitched image we can clearly observe a part on a vehicle which has been come from image 1.

References

[1] R. Zhang, "Automatic computation of a homography by ransac algorithm," Accessed on: April. 19, 2022. [Online]. Available: https://engineering.purdue.edu/kak/courses-i-teach/ECE661.08/solution/hw4_s1.pdf.