Computer Vision - Homework 4

Ioannis Gatopoulos 12141666, Philipp Ollendorff 11734078,
Konstantin Todorov 12402559, Vincent Roest 10904816

March 13, 2019

Introduction

In this report we deal with techniques for image alignment and image stitching to produce a panorama image. The process of image alignment takes two images of the same object and aligns the pixel coordinates to each other. This cancels out any rotation or shear of the underlying image caused by different camera angles or positions. The stitching part then combines the two images into one single image of greater size. If the alignment step was done right there should be a seamless contour at the intersection and therefore a final image that looks natural. We used SIFT to analyse key points of interest in both images and RANSAC to find the optimal transformation to align them. The stitching part is straightforward as it only relies on estimating the new image size and adding them next to each other.

1 Image Alignment

For extracting SIFT keypoints from an image, we used the `vl_sift` implementation [Vedaldi and Fulkerson, 2008].

Question 1.1

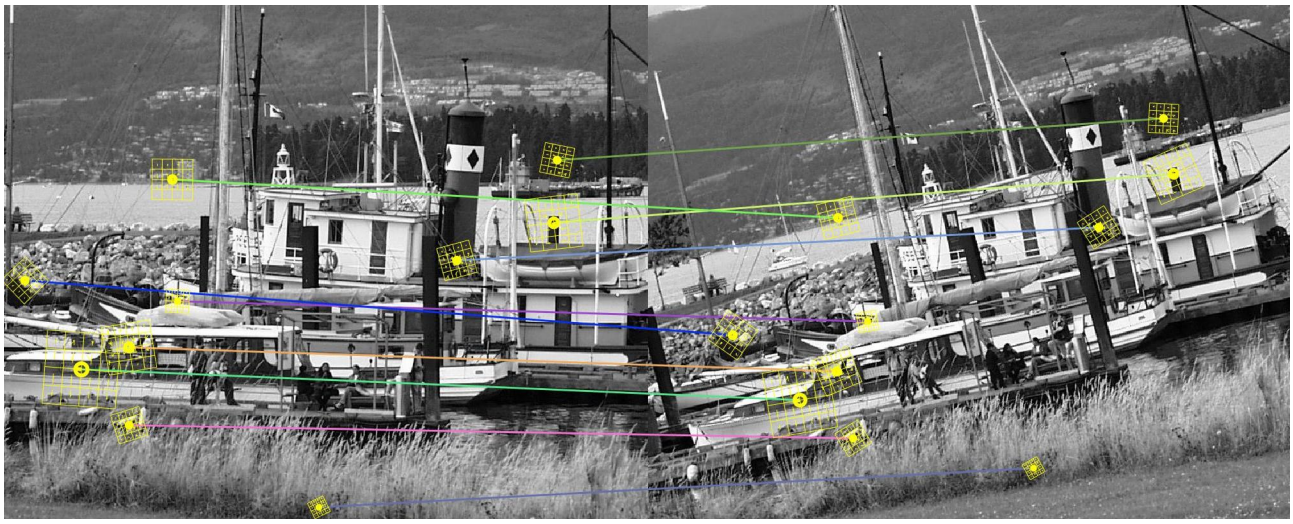
As visualized in Figure 1, there are, in total, 947 matching points between the two represented images. These matching points, or SIFT keypoints, are illustrated as the center of the yellow circles. Their diameter represent the scale (diameter and scale change accordingly) and the direction of the illustrated radius the orientation.

Figure 1: All the matching points between the two images

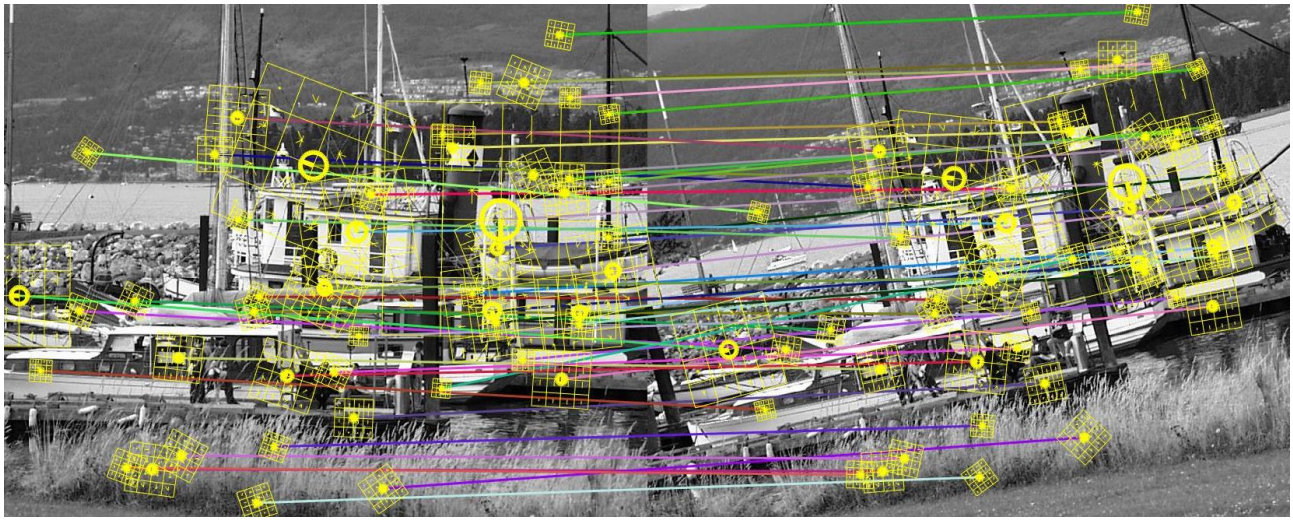
Question 1.2

In the following Figures, we illustrated 10 (random), 50 (random) and all matching points connected with a randomly colored line, so that it will be easier for matching points to be distinguished. For the first two images,

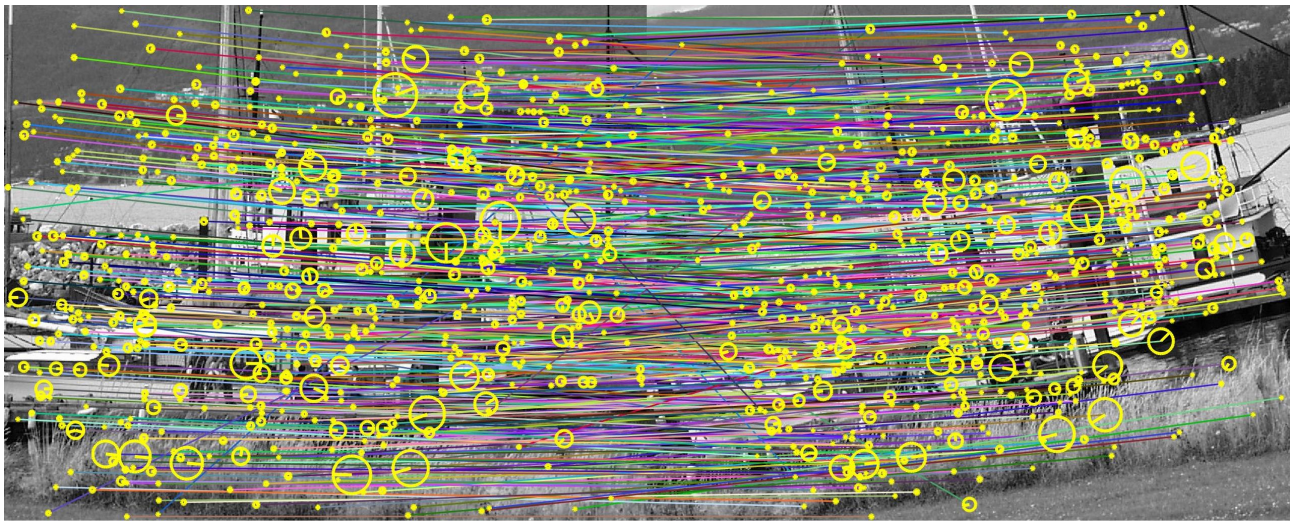
we also overlay the descriptors (4×4 frame with fixed orientations). It is important to note that these matches are noisy and at times inaccurate.



(a) 10 random key-points matched and connected with a line



(b) 50 random key-points matched and connected with a line



(c) all key-points matched and connected with a line

Question 1.3 - RANSAC algorithm

We implement the RANSAC algorithm [Fischler and Bolles, 1981] following the steps outlined in the assignment paper.

The *normalization* trick In our results in the beginning (Figure 3) , we noticed that the transformed image contains some black dots all over it, meaning some pixels were left unfilled after calculating image2 to image1 with our best $\mathbf{M}$ and t parameters. This could be seen even more clearly when comparing our result with the MATLAB transformation in Figure 4. To resolve this issue, we decided to implement additional normalization, which would get all black pixels inside the figure and set them with the average value of all of their neighbouring pixels. By implementing this trick, our results improved and for now on, we used for all our results.

(a) image2 (b) Transformation image (c) image1

Figure 3: RANSAC transformation from image2 to image1 without normalization

(a) Our transformation (b) MATLAB transformation

Figure 4: Comparison of transformation from image2 to image1 without normalization

The results of transformation from image1 to image2 are shown in Figure 5. Also included is the comparison with MATLAB's implementation in Figure 6.

(a) image1 (b) Transformation image (c) image2

Figure 5: RANSAC Transformation result from image1 to image2

(a) Our transformation (b) MATLAB transformation

Figure 6: Comparison of transformation from image1 to image2

After this, we run the RANSAC transformation algorithm again, this time from image2 to image1. You can see the results in Figure 7 and the comparison with MATLAB's implementation in Figure 8. Notice the differences now when doing normalization from Figure 4a to Figure 8a.

(a) image2 (b) Transformation image (c) image1

Figure 7: RANSAC Transformation result from image2 to image1

(a) Our transformation (b) MATLAB transformation

Figure 8: Comparison of transformation from image2 to image1

Question 2.1 - How many matches do we need to solve an affine transformation?

For any affine transformation in two-dimensional space (i.e. on regular images) we need at least three pairs of matches and therefore six points in total. The reason can be easily seen in the formula shown in Equation 1:

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \end{bmatrix} \begin{bmatrix} m1 \\ m2 \\ m3 \\ m4 \\ t1 \\ t2 \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{1}$$

Set of equations for RANSAC affine transformation

We have six unknown parameters: $m1, m2, m3, m4, t1$ and $t2$, so we need at least six linear equations to fully solve the linear system. As each match gives us one solution to this matrix equation (i.e. two coordinates per image and therefore two linear equations) we will need three matches in total to find a definite solution. Note that we could also have more than three and then find an approximate solution to the overdetermined system.

The same conclusion can also be argued intuitively: A three-dimensional plane is fully defined by three linearly independent vector points and an affine transformation can be seen as three dimensional, when looking at combinations of rotation, translation, scaling and sheering.

Question 2.2 - How many iterations in average are needed to find good transformation parameters?

The number of iterations, k, depends on the problem at hand and more specifically, on these parameters of the problem:

1. **How big is the subset of P matches from the total set of matches T?**
   The subset size should be small enough to create different transformation parameters every iteration. A too large subset size will never change its parameters, as the underlying matches do not change. There is still a chance that the parameters are accurate at the very first iteration, but if they are not there is no chance of improving. A too small subset size will do so eventually, but the required number of iterations k will be large. This observation led us to the second important parameter:

2. **How many outliers exist?**
   If SIFT found only few outliers, most subsets in RANSAC will consist of inliers and, therefore, give good transformation parameters already. This results in fewer iterations k and is therefore the desirable state. In contrast, a higher percentage of outliers to inliers results in larger k. This ratio depends entirely on the specifics of the image and the SIFT algorithm.

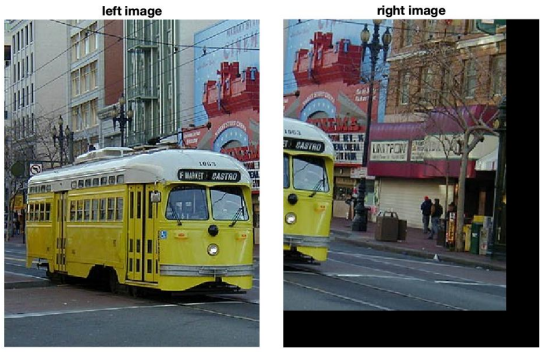
3. **How accurate does the transformation have to be?**
   In an overdetermined system the solution will always be approximate and therefore premature results might already be good enough for the desired use-case. Most improvements are made in early iterations with smaller steps later. So a high degree of accuracy will require larger k, while a rough estimate can be quickly found with smaller k.

In our own experiments for Part I we found reasonable solutions after just a handful of iterations ($k \simeq 5$), with almost unrecognizable improvements for any further iterations.

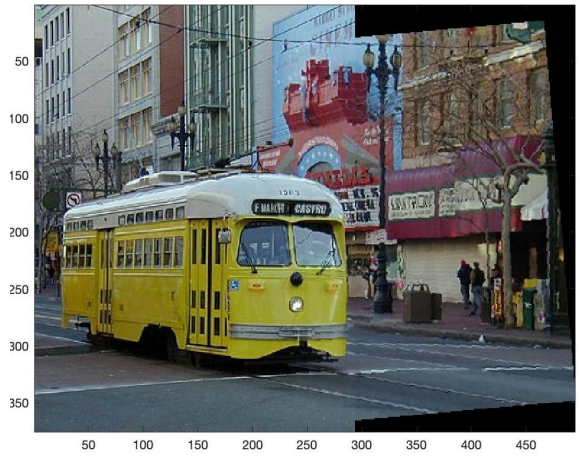
2 Image Stitching

In order to stitch together the images, the method described in the previous section is recycled. Keypoints are extracted to create matchings between the two images and they are connected through RANSAC. Instead

of the hinted procedure of calculating the transformed corners of the right image, this implementation relied on padding. The transformation to coordinate spaces was created through creating two padded images, the original left image and the transformed right image. This left image had black pixels where the image was not defined - which was exactly over halfway, after the left image was gone. Then, the result would be two images of same height and width (with both having half of its pixel values to 0). As a consequence, the merger of the two images, left and right, was simply a max operation: if the pixel value in the left image is larger than that of the right one (which is always the case in the left half), take those pixel values and otherwise take those in the right. The resulting stitched image can be seen in Figure 10. It is important to note that although RANSAC operates on the grayscale images, the pixel coordinates are still the same as in the colored version, so that we can simply apply the transformation to the colored version as well. The result can also be achieved by running the file `run_stitch.m`.



(a) Original scene



(b) Stitched scene

Figure 10: A peaceful tram scene magically appeared from two separate pictures

3 Conclusion

This report set out to use SIFT and RANSAC to find transformations between a pair of images and show their utility. First, these techniques were implemented and applied to find same keypoints between the two pictures, in terms of rotation. Then, the same technique was recycled to also merge together two pictures that belong to the same scene. As can be seen by the various illustrations, this method is very effective in achieving its purported goal, with relatively simple methods.

References

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

[Vedaldi and Fulkerson, 2008] Vedaldi, A. and Fulkerson, B. (2008). In *VLFeat: An open and portable library of computer vision algorithms.*