Computer Vision 1
Image Alignment & Stitching

Gongze Cao, Kai Liang, Xiaoxiao Wen and Zhenyu Gao
Faculty of Science, University of Amsterdam

1 Introduction

In this assignment, we cover some topics about image alignment and stitching. We begin with the implementation of the RANSAC algorithm to compute the affine transformation between two images and we compare the performance of the self-made image transformation method with the built-in `imwarp` function in section 2. Furthermore, using the aforementioned functions, we then implement the stitching function in section 3, where the affine transformation of the two input images are computed, and accordingly the two images are stitched together based on the computed translation offsets.

Throughout the assignment, we distribute our work fairly in terms of coding and writing the report.

2 Image Alignment

2.1 Question-1

1. The detailed implementation can be found in `keypoint_matching.m`. This function takes two images as input, and returns the SIFT features of the two images as well as the matching points with their distances in the feature space.

2. The visualization of matching points is shown in Figure 1.

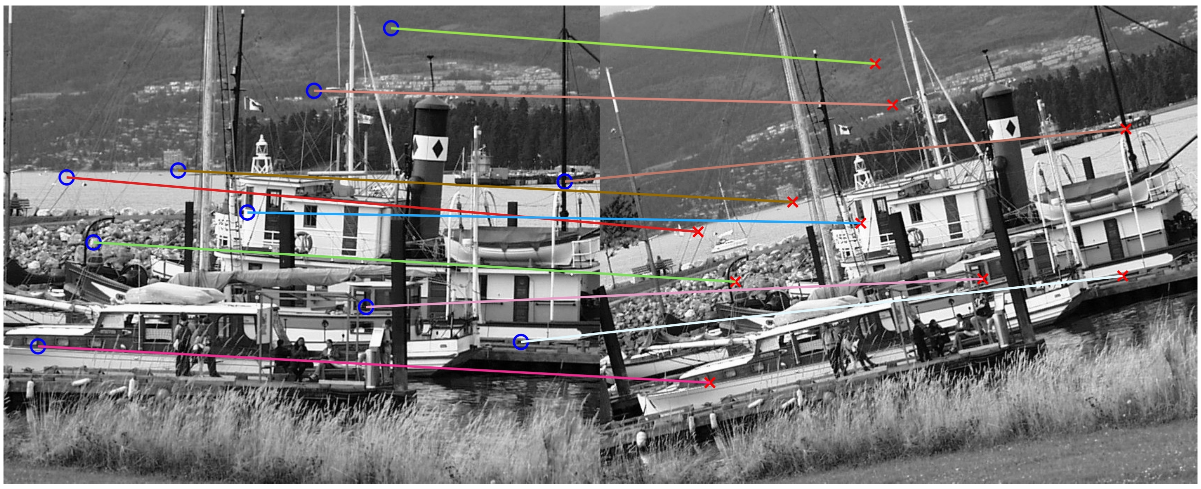

Figure 1: *Visualization of* 10 *pairs of matching keypoints in boat1.pgm and boat2.pgm.*

As can be inferred from Figure 1, the selected keypoints in *boat1.pgm* match with the corresponding points in *boat2.pgm* with few outlier, indicating that the function works as expected.

3. The detailed implementation can be found in `RANSAC.m`. For visualizations, the transformation of each of the two images is shown in Figure 2, the comparison of transformation results between the built-in `imwarp` and our `im_transform.m` is shown in Figure 3 (as can be inferred from the figure, there are some black pixels on the transformed image when using our `im_transform.m`

function, as if we do not use the linear interpolation, some pixels are missed during the transformation), and finally, the comparison of the image before and after the image transformation with the corresponding keypoints is shown in Figure 4.

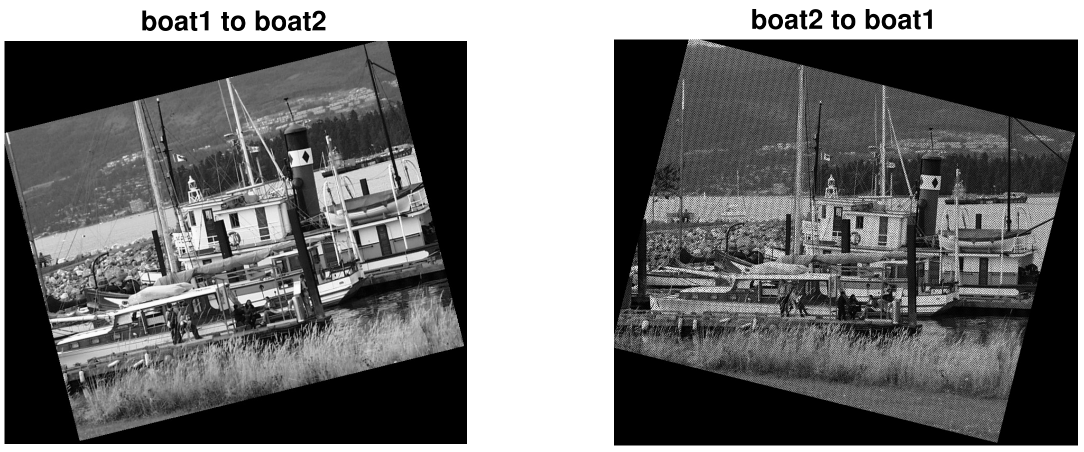

Figure 2: *Image transformation of boat1.pgm and boat2.pgm separately.*

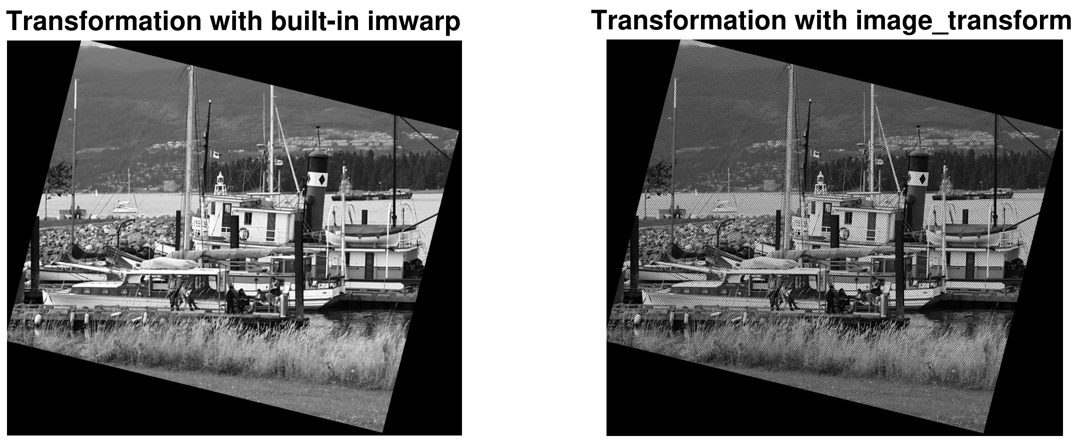
Transformation with built-in imwarp
Transformation with image_transform

Figure 3: *Comparison of image transformation results between `imwarp` and `im_transform`.*

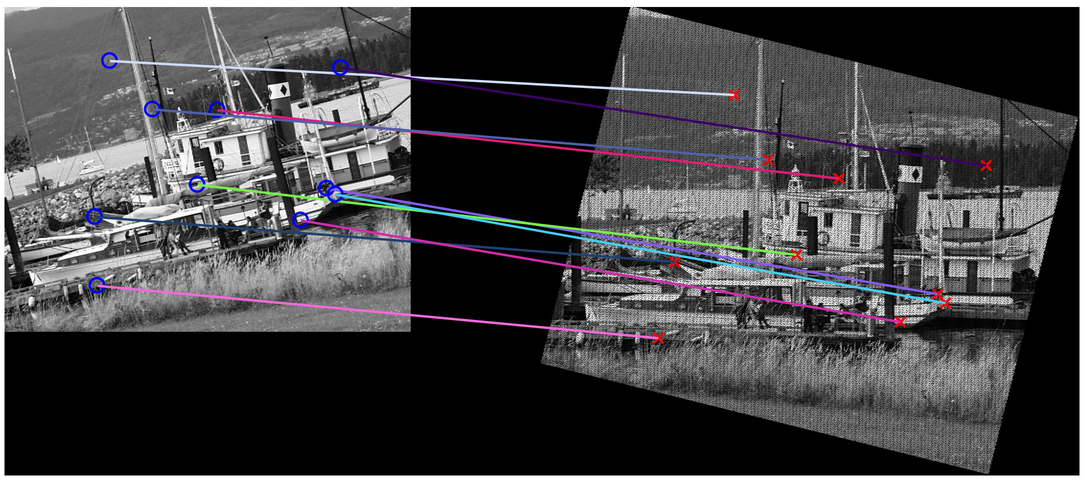

Figure 4: *Comparison of before and after the image transformation of boat2.pgm.*

2.2 Question-2

1. Given that:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

   for each pair of matching points (x, y) and (x', y'). As we have 6 unknown variables m_1, m_2, m_3, m_4, t_1 and t_2, we at least need 6 equations to solve the linear system, indicating that we need at least 3 pairs of matching points. We think that the number of pairs should be set to $7 - 10$ in practice because of the the exponentially growth of computation.

2. The number of interactions N depends on the lowest number of matching points m (the sampling number of RANSAC is 10), the percentage of inliers w ($w = 0.5$ here) and the selected probability p (this value is an input of RANSAC function), which is shown in Equation 1 [1].

$$N = \frac{\log(1-p)}{\log(1-w^m)} \tag{1}$$

   We empirically set the value of p to around 0.8 to have good and stable results in average, resulting in around 1600 iterations in total.

3 Image Stitching

3.1 Question-1

1. The detailed implementation can be found in `stitch.m`. The function takes two images as input, and first transform one of the images based on the other image, then returns the stitched image.

2. As required, the function also comes with a demo function to visualize the stitched image alongside the image pair, which are shown separately in Figure 5 and Figure 6.

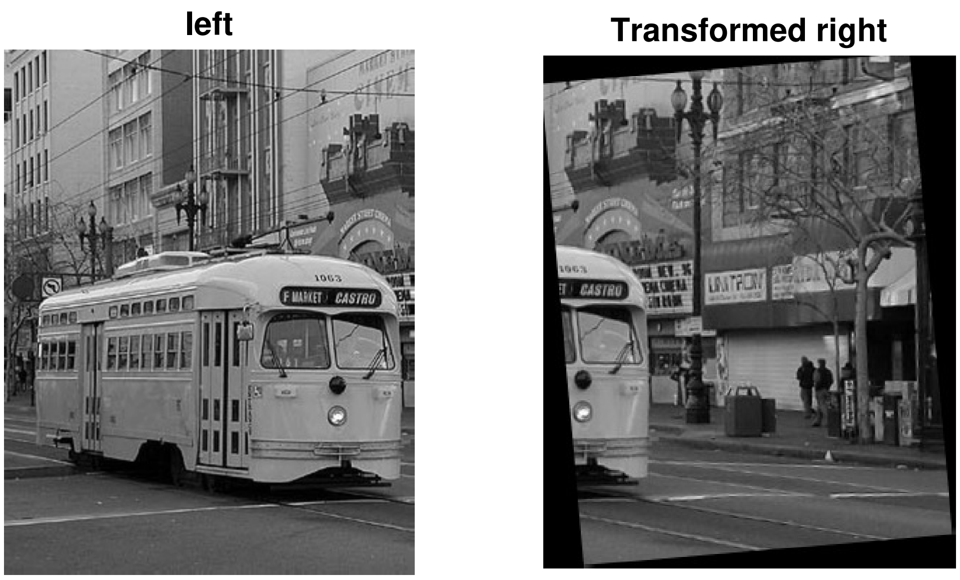

Figure 5: *left.jpg and transformed right.jpg.*

   As can be inferred from Figure 6, our stitching function is able to perfectly stitch *left.jpg* and (the transformed) *right.jpg* together.

4 Conclusion

In the previous sections, we first conduct the keypoint matching using SIFT features of two images and match the distances of keypoints in the feature space. We also visualize the matched points by drawing

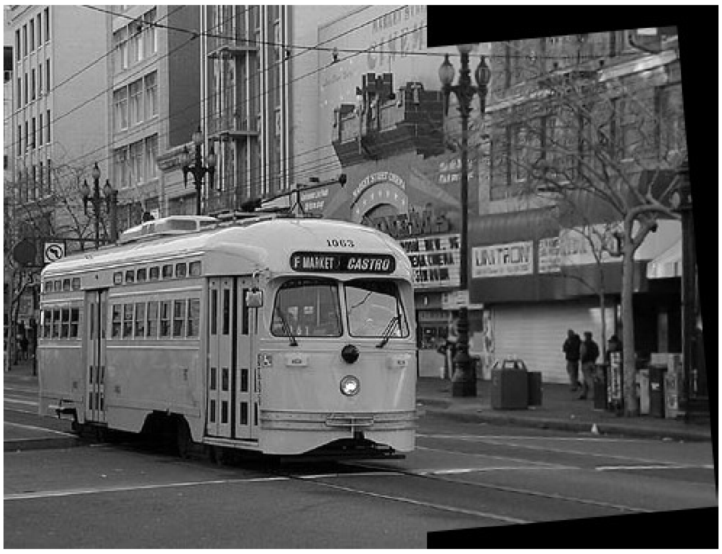

Figure 6: *Stitched image of left.jpg and transformed right.jpg.*

lines horizontally between the keypoints detected. The results shows that our algorithm is effective to detect similar keypoint with high accuracy.

We then compute the linear transformation to align all the detected keypoints together by solving an equivalent linear equation. Using the parameters obtained, we compare the warped results between the inbuilt `imwrap` and our self-implemented image transformation algorithm. The results imply that the near-neighbor interpolation is problematic as it will miss some pixels in the image grid.

Once we have the transformation parameters and the function to wrap images, we can stitch two images together. We first calculate the 'canvas' size based on the linear transformation and the size of each image to create such a canvas, and then combine both the unchanged image and the transformed image with respect to certain offsets together in the canvas. The stitched results show perfect alignment at the boundary of both images.

References

[1] T. Urbancic, M. K. Fras, B. Stopar, and B. Koler, "The influence of the input parameters selection on the ransac results," *International Journal of Simulation Modelling*, vol. 13, no. 2, pp. 159–171, 2014.