Assignment 4

Dana Kianfar - Jose Gallego Posada
11391014 - 11390689
{dana.kianfar, jose.gallegoposada}@student.uva.nl

1 Image Alignment

In this section we used the `vl_sift` implementation provided by [1] for extracting SIFT keypoints [2] from an image.

For the task of image alignment, we extracted SIFT keypoints and descriptors from two images which we will denote as *left* and *right*. We use `vl_ubcmatch` to find candidate matches between SIFT keypoint descriptors extracted from the two images. `vl_ubcmatch` find a match by the minimal $L2$ norm of the difference between any two descriptors.

We denote the set of matches obtained as M, where each entry of M is a pair of SIFT keypoints obtained from *left* and *right*. We note that these matches are noisy and at times inaccurate. We visualize these results below using a random sample of 50 matched pairs of SIFT keypoints between the images in Figure 1. Erroneous matches can be seen easily when the slope of their connecting lines is very different than the other matches.

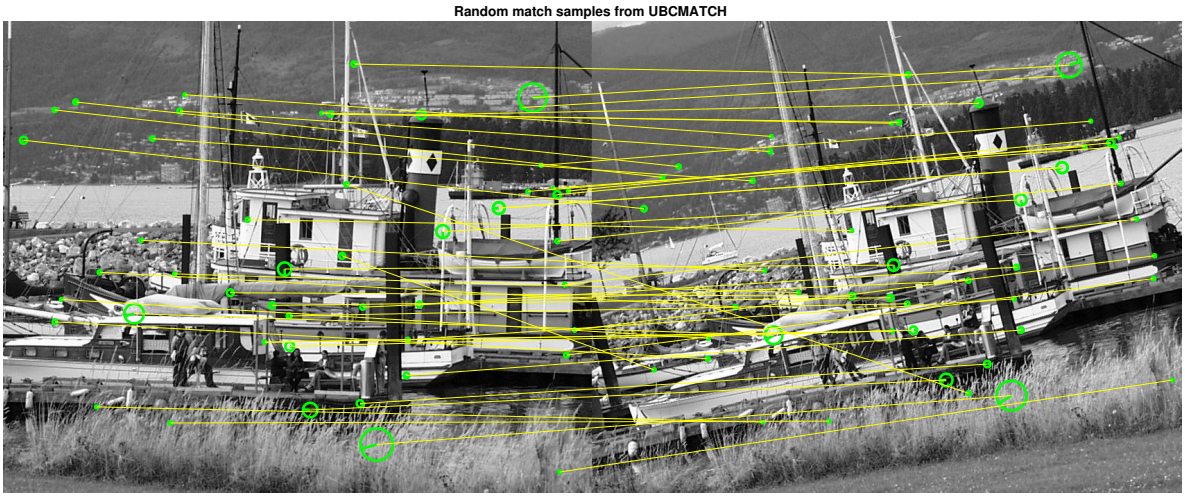

Figure 1: 50 random samples from `ubcmatch` results

We use our implementation of RANSAC to estimate the parameters of an affine between our left and right images. This affine transformation can be represented by a transformation matrix $W \in \mathbb{R}^{2\times 2}$ and a translation vector $T \in \mathbb{R}^2$, such that a point in $\begin{bmatrix} x \\ y \end{bmatrix}$ can be transformed as $W \begin{bmatrix} x \\ y \end{bmatrix} + T$.

This system has 6 unknowns, prompting us to use 3 pairs of points to obtain 6 equations. RANSAC involves an iterative voting scheme of N iterations, where we randomly sample $P = 3$ pairs of points from M and solve a linear system of equations to estimate the affine transformation parameters. We estimate N using a heuristics described in [3], expressed below:

$$N > \frac{\log(1-P)}{\log(1-(1-e)^k)}$$

where $P = 0.95$ is the desired probability of success after N trials (confidence level), $e = 0.5$ is the proportion of outliers and $k = 3$ is the number of sampled pairs. This heuristic essentially finds a lower bound for N that minimizes the probability that all samples are outliers, thus ensuring that at least one of the matches used to calculate the transformation is correct. In our experiments, the value for the number of iterations was defined using the aforementioned values for P, k and e.

We present out result using RANSAC in Figure 2. We observe that the matches are more precise than those in Figure 1 produced by `ubc_match`. As the two images have underwent an affine transformation, our matches can be represented by consistent lines connecting the matched keypoints. The results produced by our RANSAC implementation are as expected, connecting correct matching points. Note that this result is not exact, but up to a 10 pixel neighbourhood.

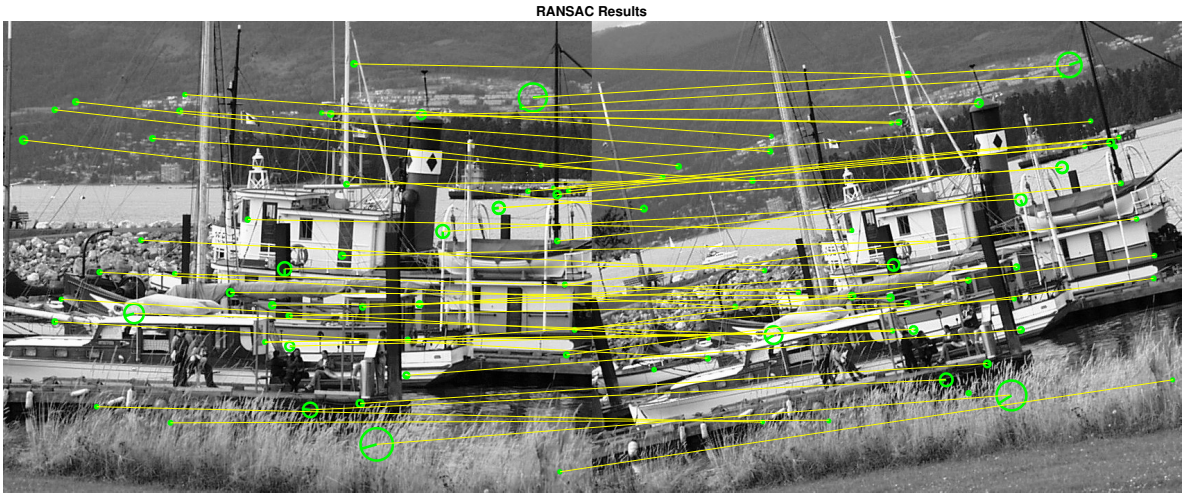

Figure 2: 50 random matches from RANSAC with $N = 23$

Using the affine transformation parameters W and T that we obtained using RANSAC, we can transform *right* to the imitate the pose of *left*. We implemented our own image transformation function that performs this transformation, and uses nearest neighbor interpolation on ambiguous pixels, i.e. pixels in the transformed *right* image that were not mapped *left* due to the quantization of pixel coordinates.

The nearest neighbor interpolation simply takes a mean of the 8 immediately surrounding pixels centered around the undefined pixel. We made the assumption that all pixels in this window were at a distance of 1 from the center pixel. In our implementation we included additionally median interpolation. This is due to the fact that the particular points that were not mapped usually have an outlier value with respect to their neighbors and a median filter is well suited for removing this kind of noise. We display our results in Figure 3.

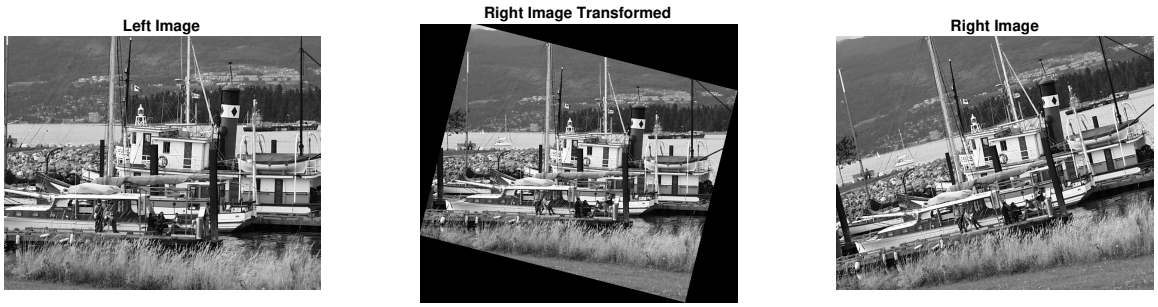

Figure 3: Transformed *right* image to match the pose of *left*

We also compare our transformation function with MATLAB's `maketform` and `imtransform` and present our results in Figure 4. Note that the two plots are very similar.

Figure 4: Our transformation compared with MATLAB. *right* is transformed to *left*

Following the methodology described above, we also estimate the affine transformation parameters for transforming *left* to *right* and display our results in a similar fashion in Figures 5 and 6.

Figure 5: Transformed *left* image to match the pose of *right*

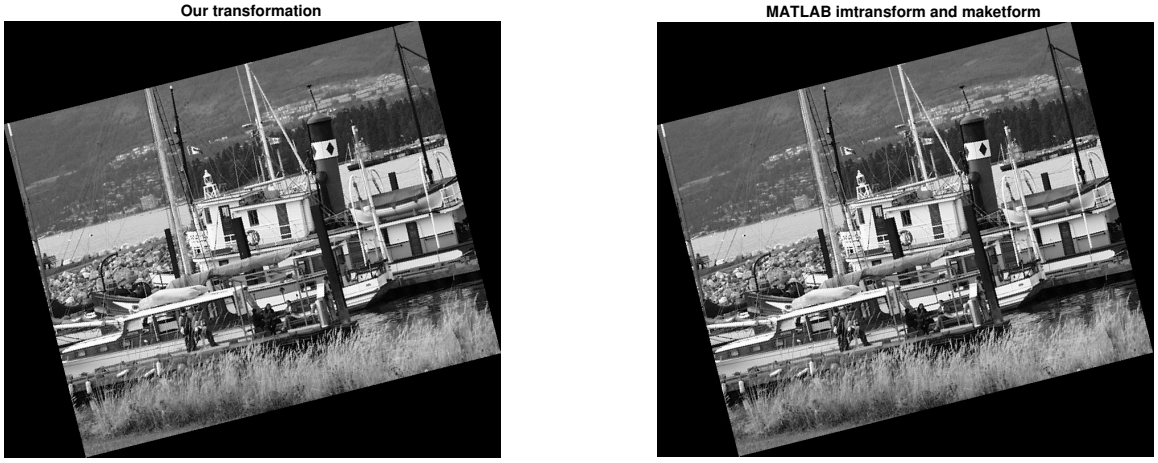

Figure 6: Our transformation compared with MATLAB. *left* is transformed to *right*

2 Image Stitching

Leveraging on the implementation performed in the previous section, we developed a stitching procedure that takes two images with overlapping fields of view $left$ and $right$ and creates ones image where *right* is stitched onto *left*. This requires that we perform the affine transformation of *right* to be able to represent it in terms of the coordinates of *left*.

To perform the stitching, we first obtain the transformation parameters W and T using RANSAC on SIFT keypoint matchings, as explained in the previous section. We then define the coordinate mapping function, denoted by $f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right)$, that takes as inputs the coordinate of each pixel in *right* and computes its coordinates with respect to *left* using W and T. This can be expressed as:

$$f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = W \begin{bmatrix} x \\ y \end{bmatrix} + T + \Delta$$

where $\begin{bmatrix} x \\ y \end{bmatrix}$ is any pixel coordinate in *right* and $\Delta \in \mathbb{R}^2$ is a vector that translates the image coordinates such that all coordinates are larger than or equal to $(1, 1)$ for correct storage of the images in MATLAB, i.e. to correct negative coordinates. The mapping is then performed by the following operation on each pixel coordinate from the *right* space:

$$I_L\left(f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right)\right) = I_R\left(\begin{bmatrix} x \\ y \end{bmatrix}\right)$$

The results of the application of this technique are displayed in Figure 7. Similar to the previous section, we perform interpolation on missing pixel values. The stitching of the images is performed correctly, as can be seen verified by the alignment of the lines on the street, the trajectory of the wire and the position of the red advertisement.

Note that the quality of the stitching depends directly on the RANSAC outcome. Now, given that the transformation found by the RANSAC procedure is based on N random sets of samples from the matches, the results of the stitching could vary slightly.

Figure 7: Stitching results with NN interpolation

Additionally, in Figure 8 we display the results of the algorithm using median interpolation. These do not display significant change even though median filtering is more theoretically suited.

Figure 8: Stitching results with median filtering

References

[1] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/, 2008.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. New York, NY, USA: Springer-Verlag New York, Inc., 2010.