Computer Vision 1 Assignment 4

Dmitrii Krasheninnikov
University of Amsterdam
11719230
`dmkr0001@gmail.com`

Akash Raj Komarlu Narendra Gupta
University of Amsterdam
11617586
`akashrajkn@gmail.com`

1 Introduction

Image alignment and matching are important components of many computer vision tasks like object detection and tracking. In this report, we first experiment with SIFT features to compute image descriptors and match them. Following this, we estimate an affine transformation between the images using RANSAC. Finally, we stitch two images together using the descriptors and resulting transformation. One of the simplest applications of image matching and stitching is *Panography*, where images are transformed and blended together.

2 Image Alignment

Figure 1 shows the image descriptors (features) computed by SIFT algorithm. We used the `vl_sift` function provided in the VLFeat library [1] to compute the Scale Invariant Feature Transform (SIFT) [2] frames of the image.

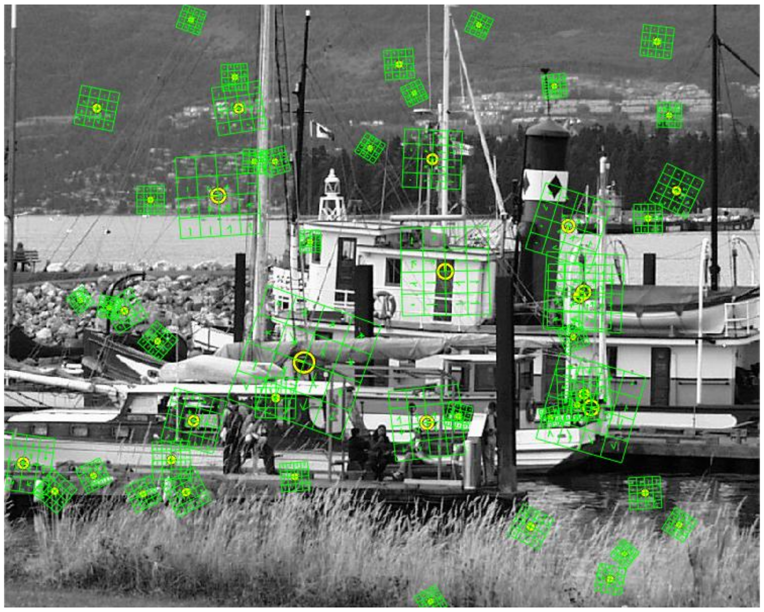

Figure 1: Descriptors for `boat1` image.

SIFT features are invariant to scale and rotation, and are robust to changes in illumination and noise. It provides a reliable method of matching between different views of a scene. The following steps are used to compute SIFT features:

1. **Scale space extrema detection**: Various locations and scale settings are explored to identify the interest points that possess orientation and scale invariance.
2. **Keypoint localization**: Scale and location are determined by fitting a model at the interest points.
3. **Orientation assignment**: Based on the local gradient direction, each location is assigned an orientation (provides invariance). The transformed image is used for further operations.
4. **Keypoint descriptor**: Gradients are measured around each interest point and it is transformed into a representation that is stable to small changes of illumination and noise.

For aligning the two `boat` images, first we computed the SIFT features (descriptors) for the two images. The descriptors of the second image are "matched" to the first based on the Euclidean distance (or L^2 norm). We used the `vl_ubcmatch` function to match the descriptors between the two images. Each descriptor of frame is a 128-dimensional vector of integers which makes it highly distinctive. This allows it to be matched with a high probability to the correct descriptor in the second image. Figure 2 shows a random sample of 50 matched descriptors. We observe a small fraction of erroneous descriptor matchings (lines with slope substantially different from others).

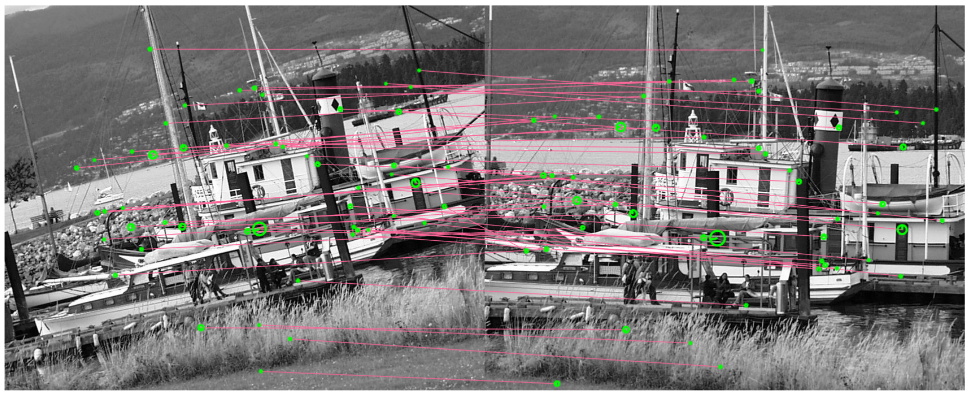

Figure 2: 50 random keypoint matchings between `boat1` and `boat2` images.

Affine transformation

Affine transformation maps one space to another by preserving the straight lines (points) and planes, and preserving the ratio of distances between two points in a straight line. Let **X** and **Y** be affine spaces; then an affine transformation is of the form $x' \to Mx + b$, where M is a linear transformation. For a point (x, y) in an image, we can define an affine transformation as,

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

where (x', y') is the transformed point. We can rewrite the above equation to the form, $Ac = b$, which can be solved using a pseudo-inverse, $c = (A^T A)^{-1} A^T b$, where

$$A = \begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \end{bmatrix}, \; c = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix}, \; b = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

RANSAC and Nearest neighbor interpolation

Random sample consensus (RANSAC) [3] is a method to fit a model to the observed data. Model parameters are estimated by an iterative method. We use RANSAC to estimate the parameter vector

c of the affine transformation. Given a dataset containing outliers (erroneous descriptors), RANSAC uses a voting scheme to get the optimal parameters. It is a non-deterministic approach, and the probability of obtaining the optimal transformation parameters increases with the number of iterations. In each iteration:

1. P matched descriptors are chosen at random. The parameters of the affine transformation model are determined using the P matchings.
2. Using the above parameters, the entire set of matchings is transformed. We count the number of inliers that are within $\epsilon = 10px$ of their actual location, i.e., $||r_i|| \leq \epsilon$. (We define a transformed point as "inlier" if the Euclidean distance to the original point is within 10 pixels.)

The best transformation is the one with the largest number of inliers.

The coordinates of the pixels in the transformed image are not perfectly lined up due to coordinates being rounded to integer values. The remaining missing pixels ("black dots") in the transformed image were filled with the values of the nearest neighbor pixels in the original image, to get which we performed a transformation of the missing pixel coordinates into the space of the original image: $[x, y] = ([x', y'] - [t_1, t_2]) \times M^{-1}$.

Figure 3 shows a transformed `boat2`, when RANSAC was performed with $N = 50$ iterations and $P = 3$. For comparison, we also used MATLAB's built-in functions `imtransform` and `maketform`. The result from our transform function is very similar to that of MATLAB's.

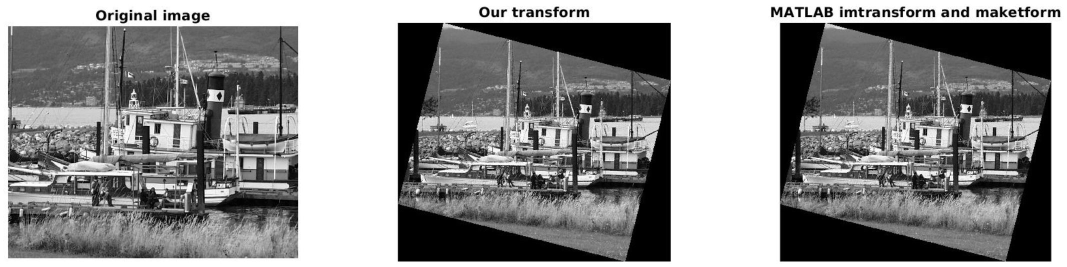

Figure 3: The leftmost figure is the original `boat1` image. The second figure shows the transformed `boat2` image. The rightmost figure is the transformed `boat2` using the built-in MATLAB functions.

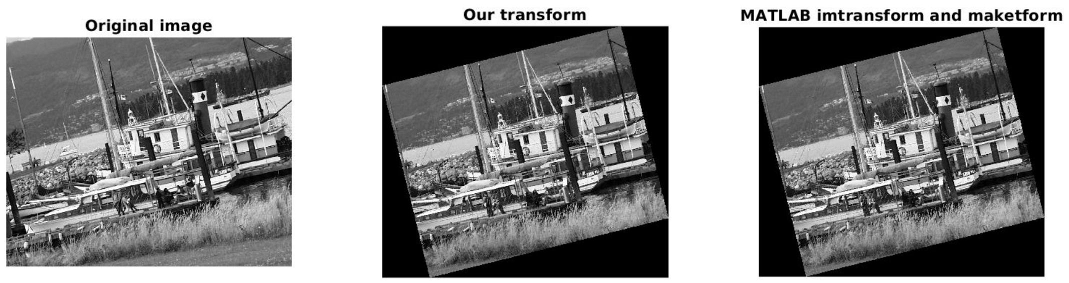

Figure 4: The leftmost figure is the original `boat2` image. The second figure shows the transformed `boat1` image. The rightmost figure is the transformed `boat1` using the built-in MATLAB functions.

The affine transformation is a system of equations with six unknowns. To get a unique solution, we would need at least $P = 3$ matches, since each match generates two equations. However, it is better to have more matches since one could accidentally sample e. g. 3 points on one line which would result in infinitely many solutions to the system, which is not at all useful.

The minimum number of iterations, N is computed using the method given in [4]. Let q be the probability that a point is an inlier. The likelihood that all P samples are inliers is q^P (in one iteration). Let Q be the total probability of success after N iterations.

$$\text{Likelihood that all } N \text{ iterations will fail, } 1 - Q = (1 - q^P)^N$$

$$\text{Minimum number of iterations, } N = \frac{\log(1 - Q)}{\log(1 - q^P)}$$

To get a probability of success, $Q = 0.95$, with $P = 3$ and $q = 0.5$, we get,

$$N = \frac{\log(1 - 0.95)}{\log(1 - 0.5^3)} \sim 23$$

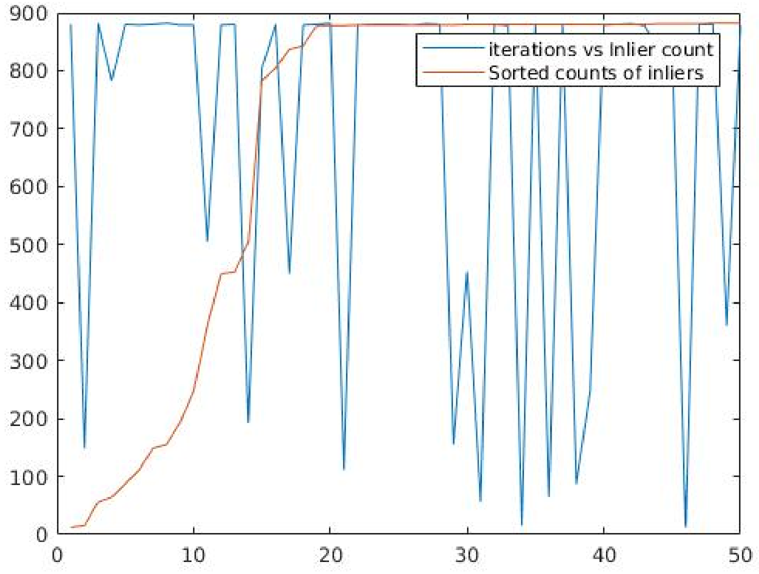

Figure 5: Plot shows the number of inliers per RANSAC iteration.

3 Image Stitching

With the help of image alignment functions described in the previous section we have implemented a simple algorithm that stitches together two images with overlapping scenes. Throughout this assignment we are working with two images of a tram in the street, `left.jpg` and `right.jpg` (Figure 6), and are transforming `right.jpg` and stitching it with `left.jpg`, which remains unchanged.

First, as in the previous section, we find an affine transformation c that transforms `right.jpg` to correspond to the pose of the objects in `left.jpg` using keypoint matching and RANSAC. Hereafter we rotate `right.jpg` using $[m_1, m_2; m_3, m_4]$ from the transformation c and do nearest neighbor interpolation for the missing pixels in the transformed image. Finally, we compute the dimensions of the resulting stitched image, and position `left.jpg` and the transformed `right.jpg` in the stitched image using the $[t_1, t_2]$ components of c and various image sizes. The pixel values of the intersecting part of the two images are taken from the untransformed `left.jpg` since this results in a higher quality of the stitched image (the transformation reduces image quality).

The result of the stitching procedure is shown at Figure 7. We observe that stitching works well: the wire and the various line on the road are aligned well. However, the result of stitching depends on the results of RANSAC and thus varies from time to time: for example, sometimes the vertical alignment of the parts of the stitched image would be off by 1-2 pixels. Figure 8 demonstrates that the image quality is indeed reduced if the pixel values of the transformed `right.jpg` replace the values `left.jpg` in the overlapping image region.

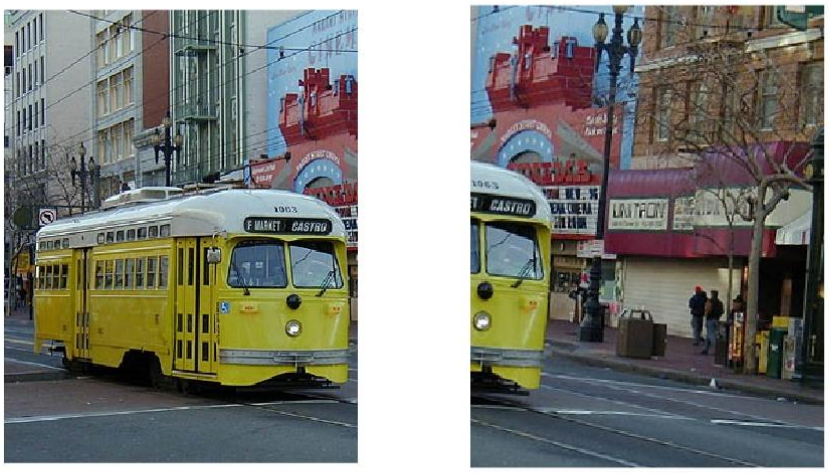

Figure 6: The original images `left.jpg` and `right.jpg`.

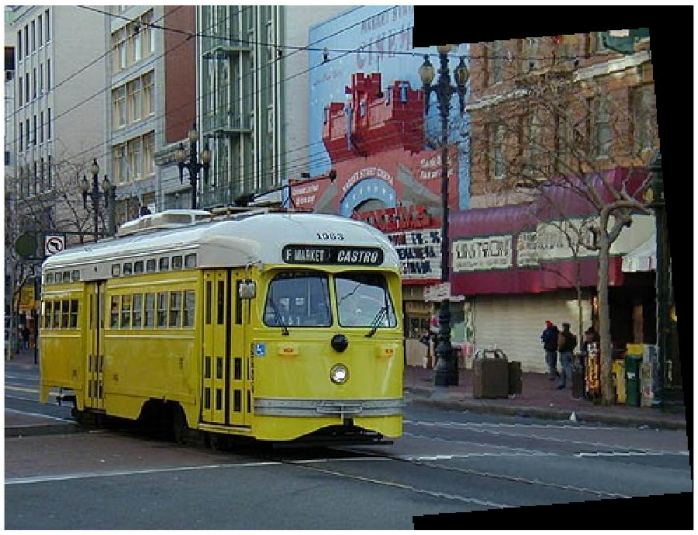

Figure 7: `right.jpg` transformed to correspond to the pose of the objects in `left.jpg` and stitched with `left.jpg`, such that `left.jpg` is “on top” of `right.jpg`.

4 Conclusion

In this report we explored the basics of image alignment and stitching. For the Image matching task, we first studied the steps involved in computing the SIFT features. Using these scale invariant features, the descriptors in the two images were matched to find an affine transformation whose parameters were estimated by RANSAC algorithm with the nearest neighbor interpolation. After matching the image descriptors, the two images were stitched together by transforming one image

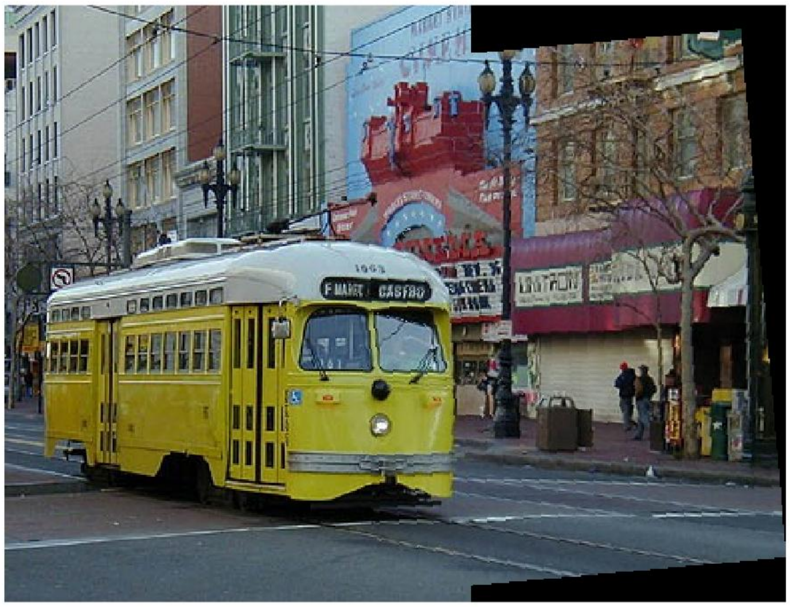

Figure 8: `right.jpg` transformed to correspond to the pose of the objects in `left.jpg` and stitched with `left.jpg`, such that `right.jpg` is "on top" of `left.jpg`.

into the coordinate space of the other. Future work may include using a more robust method of interpolation (e.g. linear) instead of the nearest neighbors variant.

References

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

[2] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2(60):91–110, 2004.

[3] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[4] Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2(1):1–104, 2006.