Assignment 4
Image Alignment & Stitching

Maartje de Jonge
0194107
University of Amsterdam
maartjedejonge@gmail.com

Lea van den Brink
10909419
University of Amsterdam
leavdb@hotmail.com

Introduction

Image stitching is a technique used to make panorama's of images that cover the same scene. That is, the images are aligned first and then concatenated into a single image. In this fourth assignment we take a deeper look into image alignment and image stitching. We used two important computer vision techniques for image alignment. First, we identify matching keypoints between two images of the same subject using scale-invariant feature transform (SIFT) [4] descriptions. Secondly, we identify the transformation parameters using RANSAC [2], assuming that an affine transformation [3] exists between the two images. To construct the transformed image we use nearest neighbour interpolation. Image alignment is described in Section 1 and image stitching is described in Section 2.

1 Image Alignment

- **Question 1.1**
  As a first step for image alignment we need to find matching keypoints. We implemented keypoint matching using functions of VLFeat, following the guidelines in `http://www.vlfeat.org/overview/sift.html`.

  The keypoint matching involves the following steps: First we convert both images to grey scale. Next we identify interest points and characterize their local appearance using the function *vl_sift*. The interest points are described by their location, their scale and their dominant orientation of the gradient. The local appearances are characterized by 128 gradient orientation frequencies that cover a 16x16 neighbourhood around the interest points, divided into sub-blocks of 4x4 size. The local appearances are used to establish the supposed matches between the interest points using the function *vl_ubcmatch*. The function uses the algorithm suggested by D. Lowe [5] to reject matches that are too ambiguous.
- **Question 1.2**
  The result of key-point matching is shown in Figure 1. The figure shows two images of the same boat with a different camera position. The green circles indicate the interest points, whereby the size of the circle visualizes the scale of the region being used to characterize the local appearance and the direction of the line inside the circle visualizes the dominant direction of the gradient. The yellow lines connect the interest points that are identified as matches. We see mostly good matches that connect corresponding points in the two images, however we can identify some mismatches as well. For example, in the middle we see a line with a completely different direction that connects a point of the mast with a point in the

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

grass. If we check the orientation indications we see that most circles in the second image have an orientation that is slightly turned counter clockwise compared to the orientation of their corresponding circles in the first image. This observation is inline with the visual orientation difference, that is the boat in the second image is turned a bit counter clockwise.

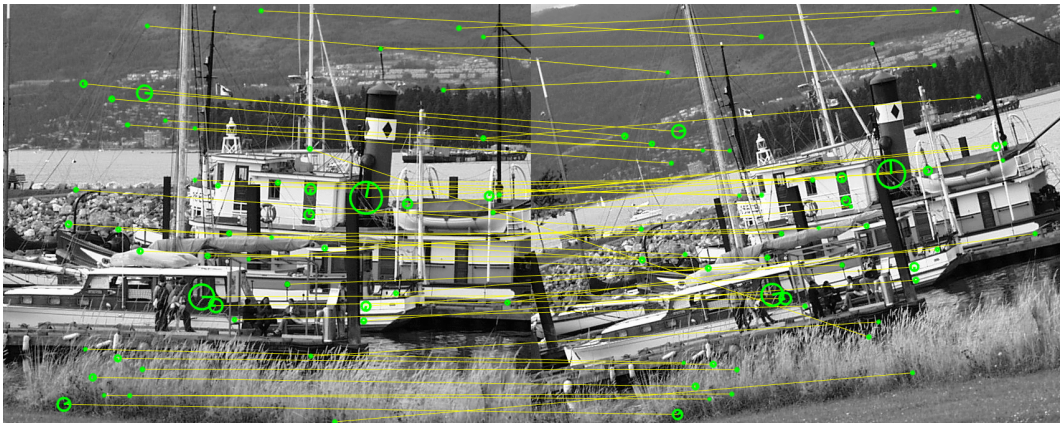

Figure 1: Two images of the same boat. **(First)** taken straight and **(Second)** at an angle. The green circles indicate the keypoints that have been matched, connected by yellow lines.

- **Question 1.3**
  We implemented random sample consensus [2] (RANSAC) to estimate the optimal transformation between the two boat images. The implementation consists of the following steps that are repeated n times: First we select P matches from all matches returned from our keypoint matching function. Next we calculate the transformation parameters based on these selected matches. We do this by solving the equation $x = (A^T A)^{-1} A^T b$, with x the transformation parameters and A and b the matrices build by vertically concatenating the (sub)matrices shown below for the selected keypoints P. We then transform all keypoints from image1 and count the number of inliers. We define an inlier as a transformed point that lies within a radius of 10 pixels from its pair point in the second image. We save the transformation parameters and the inliers if this count exceeds the best total so far. After finishing the n loops we calculate the final transformation parameters by recomputing the least square estimate on all of the inliers.

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

  We used $n = 10$ and $P = 4$ (as suggested in the lecture slides) as parameters for generating the figures in this report. We discuss the choice of these parameters when answering question question 2.1 and question 2.2. We then also visualize the sampling process by showing the result of a good sample and the result of a bad sample.

  The RANSAC algorithm gives us the parameters for the matrix M and the matrix T. We can use these matrices to transform the image using the equation:

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

  Transforming the pixel coordinates does not directly gives us the new, aligned image. The problem is that the transformed points in generally do not lay on perfect pixel coordinate, that is, they are given by decimal numbers. Simply rounding these numbers is not sufficient since they may be separated by a distance that is smaller or larger than one. In the former case we have multiple values for a single pixel, in the latter case we have 'black holes' in our image. We calculated the pixel values of the transformed image by using nearest neighbour

interpolation. That is, we gave each pixel in the new image the value determined by the transformed (x', y') at the smallest euclidean distance.

Figure 3 (left) and 4 (left) show the results of applying the transformation estimated with RANSAC on the original boat images shown in Figure 2. We see that the poses of the boat in the transformed images correspond to the pose in the original target image.

The left images in Figure 3 and 4 show the result of our own transformation using nearest neighbour interpolation, the middle and right image show the result of applying the matlab functions $affine2d$ and $imwarp$ with respectively nearest neighbour and linear interpolation. When zooming in, we see, as expected, that the result of our own transformation looks very similar to the result of the matlab transformation using nearest neighbour interpolation. At the other hand, the matlab result using linear interpolation looks less blurry and jagged, especially at the edges of objects.

Figure 2: Original images

Figure 3: Left image of Figure 2 aligned with the right image using RANSAC. The images show the result of our own transformation using nearest neighbour interpolation (first) and the results of using the matlab functions $affine2d$ and $imwarp$ with respectively nearest neighbour interpolation (second) and linear interpolation (third).

Figure 4: Right image of Figure 2 aligned with the left image using RANSAC. The images show the result of our own transformation using nearest neighbour interpolation (first) and the results of using the matlab functions $affine2d$ and $imwarp$ with respectively nearest neighbour interpolation (second) and linear interpolation (third).

- **Question 2.1**

An two dimensional affine transformation [3] is a geometric transformation that composes the effects of translation, rotation, isotropic scaling and shear. The transformation has the property that parallel lines remain parallel, but corners can be distorted.

Since the general affine transformation is defined by six parameters, we can determine the transformation by specifying three matching points consisting of an x and y coordinate. The six resulting equations are sufficient to determine the solution of the linear system with six unknowns (assumed that the points are not on the same line). We can also explain this intuitively:

Suppose we have a rectangle defined by the following cartesian coordinates: $A = (0, 0)$, $B = (0, 1)$, $C = (1, 1)$, $D = (1, 0)$.

We first show why two matches is not sufficient to determine the full affine transformation. Assume that we have the following matches: $A' = (0, 0)$ and $B' = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$, i.e. A is mapped to itself while B is rotated slightly with respect to A. Now we can not be sure what happened to the points C and D. For example, the actual transformation could be a pure shear transformation in which case the point D stays at $(1, 0)$ and C moves to $(1\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. At the other hand, the actual transformation could also be a rotation (plus translation) in which case both C and D move. We conclude that the actual transformation is not fully determined by the positions of A' and B'.

We next show why three matches is sufficient. Suppose that A, B and C move to A', B' and C'. Then the line through C' and D' must be parallel to the line through A' and B'. Furthermore, the line through B' and D' must be parallel to the line through A' and C'. These two lines uniquely determine the position of D'. The same reasoning can be applied to other points in the coordinate frame.

We experimentally verified that picking 3 points instead of 4 in the boat image transformation also gave good results. However, we kept the number of selected matches on four in the rest of the experiments since it was the recommended number in the lecture slides.

- **Question 2.2**

Th RANSAC algorithm implements an iterative process whereby for each iteration a transformation is calculated based on a small set of selected matches. If the selected matches are good matches, the resulting transformation is good as well. An example is shown in Figure 5. The red crosses and lines represent the selected matches, the yellow lines visualize the transformation on a sample of 20 points. We see that all points in the left image are transformed to their corresponding points in the right image. In contrast, if the selected matches are poor, the resulting transformation will also be poor. This is illustrated in Figure 6. The upper-left red point is not matched properly, which has the effect that almost all points of the left image are not transformed to their corresponding positions in the right image.

We will now estimate the average number of iterations needed to find a good transformation. We assume that a sample containing only good matches results in a good transformation, while a sample that contains one or more poor matches results in a poor transformation (given a small sample size, 4 in our case). For simplicity we assume that we have much more matches than selected matches so that we can ignore the effect of drawing without replacement and use an binomial approximation.

Let M be the total number of matches, let G be the total number of good matches, and let P be the number of selected matches per iteration. The change of success, i.e. the probability of selecting only good matches in an iteration is approximated by: $(\frac{G}{M})^P$. The average number of trials that lead to the first success [1] is now given by $\frac{1}{(\frac{G}{M})^P} = (\frac{M}{G})^P$.

We estimate the numbers for the transformation that aligns the left image with the right image in Figure 2. We take the number of inliers of the best transformation over 20 iterations as a good approximation of the number of good matches. This gives us the following numbers: $P = 4$, $M = 947$ and $G = 905$. From these values we calculate the average number of iterations required as: $(\frac{M}{G})^P = (\frac{947}{905})^4 = 1.20$.

We verified experimentally that indeed a good transformation is found quickly in most cases. We kept the number of loops on 10 since the sampling steps are fast.

Figure 5: RANSAC iteration performed on a good selection. The red crosses and lines represent the selected matches, the yellow lines visualize the transformation on a sample of 20 points.

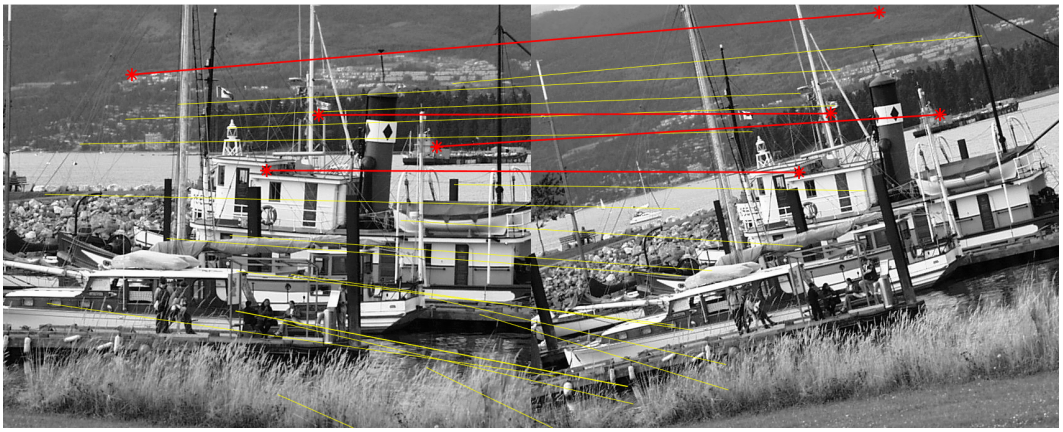

Figure 6: RANSAC iteration performed on a bad selection. The upper-left red point is not matched properly, which results in a poor transformation visualized by the yellow lines.

2 Image Stitching

- **Question 1.1**
  We implemented a function that takes an image pair as input and returns a single image that stitches the two images together. The stitching function involves the following steps: First, we transform one of the images using keypoint matching and RANSAC, as discussed in Section 1. We then calculate the coordinate frame that contains both the transformed image and the other original image. We translate the frame so that the minimum x and minimum y values are both 0, i.e. we wants the stitched image to fit in an image frame with $(0, 0)$ as the coordinate of the top left corner. Finally, we fill the image frame with the values from the transformed image and the other original image. In case a point is in the overlapping region we give preference to the pixel values of the non-transformed image. The reasoning for this is that the transformed image has lower quality since it has been constructed by interpolation.

- **Question 1.2**

  Figure 8 illustrates the result of image stitching, taking the images of Figure 7 as input. The figure shows the stitching result after transforming respectively the right image (first) and the left image (second) of Figure 7. To handle the RGB color values, we transformed the color channels separately and stacked the results together.

Figure 7: Original images.

Figure 8: Stitched images based on transforming respectively the right image (First) or the left image (Second) of Figure 7

3 Conclusion

Image stitching is a technique that is used in modern photo cameras to built a panorama. That is, two (or more) images are stitched together side-by-side to form a single composed image.

In this assignment we experimented with the techniques that are used to built such panoramas. First, we implemented keypoint matching between two images of the same subject based on SIFT features. We then used the RANSAC algorithm to calculate the alignment transformation that results in the highest number of inliers given the keypoint matches. We implemented the actual image transformation using nearest neighbour interpolation. Finally, we stitched the transformed image together with the other original image to construct a 'panorama'. Using these techniques we successfully stitched to images of the same tram together into a 'panorama' view of the tram.

References

[1] A. Bogomolny. Number of trials to first success, 2018.

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

[3] Perkins S. Walker A. Fisher, R. and E. Wolfart. Affine transformation, 2003.

[4] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

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