Assignment 4: Image Alignment and Stitching

Davide Belli
11887532
davide.belli@student.uva.nl

Gabriele Cesa
11887524
gabriele.cesa@student.uva.nl

Introduction

In this Assignment, we are going to experiment with Image Alignment. Given two input images representing some scene, this tasks consists in finding the best affine transformation that transforms one image into the other. To do this, first we will find some interest points and we will match them between the two images using David Lowe's SIFT. Then, we will perform RANSAC algorithm over a subset of matching pairs. This algorithm aims to find the best affine transformation described as the solution of a matrix multiplication where the parameter matrix describes the rotation, translation and scaling components in the image transformation. The output transformation returned by RANSAC can then be used to transform one image in the same coordinate space of the other and to stitch the images together in an optimal way.

1 Image Alignment

1.1

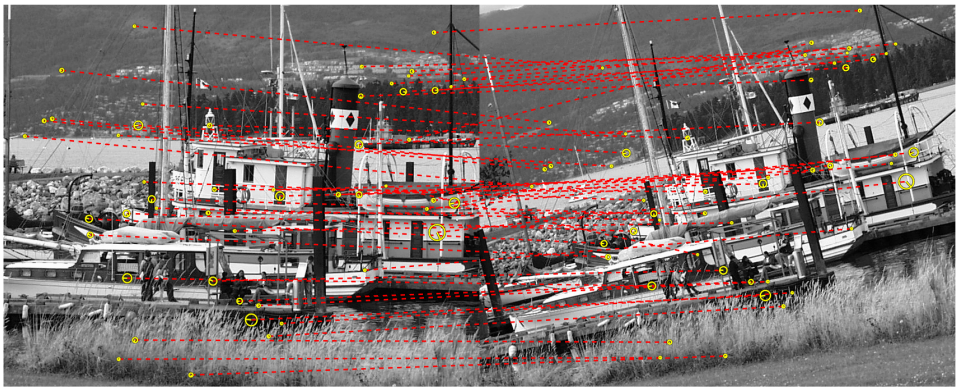

Figure 1: Interest points in pictures and their matchings between the two images

In Fig. 1 we show the matchings found between interest points in the two sample images using the functions from David Lowe's SIFT. Interest points are characterized by center coordinates, scale and orientation.

In Fig. 2 we can see the the best transformation found by the RANSAC algorithm applied to *boat1.pgm* to transform it to *boat2.pgm*. The reverse transformation is then applied to *boat2.pgm* to have it matching *boat1.pgm*, as shown in Fig. 3. In both cases, we decided to keep the transformed image the same size as the other one in order to make them directly overlapping regardless to the zoom factor induced by having higher or lower image size.

Figure 2: Best transformation found by RANSAC applied to transform *boat1.pgm* to *boat2.pgm*

Figure 3: Best transformation found by RANSAC is reversed and then applied to transform *boat2.pgm*

For comparison, we also plot in Fig. 4 the same transformations performed using MATLAB's functions *affine2d* and *imwarp*. Here, by default, image sizes are widened to contain the whole transformed picture. As a result, it is not easy to directly compare the scaling factor in the original image with the one in the transformed image due to a different zoom factor in the preview.

1.2

In order to mathematically solve the affine transformation using the linear least squares method (with the pseudo-inverse), we need at least three matching pairs of interest points. If we tried using fewer matches, we could not correctly capture the scaling, translation or rotation aspects in our transformation. To demonstrate this, we can see that for every matching pair we define two different equations with 3 variables each. Thus, we need at least 3 of those pairs of equations (in total 6 equations in 6 variables) to be able to solve for the 6 variables (so that the system of equations is not under-determined).

Moreover, since we are using the linear least squares method, it is better to have an over-determinate system (i.e. with more equations than variables, and so at least 4 pairs) to make the solution more robust to outliers. Indeed, since RANSAC works by selecting random subsets of matches, choosing only three values may result in computing the wrong transformations when the matching between two images have many errors (outliers). Let's consider the case in which three matches represent a very similar transformation that, however, is not the one we want to find (e.g one object that doesn't appear in the second image is wrongly mapped to a similar one not appearing in the first one). In this situation, if three matches from that wrong transformation are picked as first subset in RANSAC, we may find out that this transformation is optimal since we would find no outliers by applying

(a) MATLAB's *imwarp* applied to *boat1.pgm*

(b) MATLAB's *imwarp* applied to *boat2.pgm*

Figure 4

parameters found with the pseudo-inverse. On the other hand, if we would have chosen a larger set of matches *P* in RANSAC, it would have been less likely to obtain a subset where every point belongs to that wrong transformation. Thus, those parameters wouldn't be selected as optimal ones. It is important, however, to not use a very large subset size *P*, as in that case it would be very likely to include outlier matches most of the times.

The number of iterations in RANSAC to be able to find good parameters depends on two hyper-parameters. At first, by choosing a smaller set of matching pairs *P* for each iteration, it is easier to find a transformation resulting in few outliers. With $P = 10$ and $N = 50$, it usually took from 1 to 5 iterations to converge to no outliers. By increasing the number of P, the number of iterations to "converge" also increases up to 30 or 40. In addition, by specifying a Peak Threshold greater than zero when calling *vl_sift*, only the most relevant interest points are returned, resulting in a fewer number of outliers when running RANSAC.

2 Image Stitching

2.1

(a) Input images *left.jpg* and *right.jpg*

(b) Stitched images after finding best transformation from *right.jpg* to *left.jpg*

Figure 5

For this task, we computed the best transformation from *right.jpg* coordinate space to *left.jpg* space using the algorithm described in Task 1. Afterwards, we applied that transformation to *right.jpg* and combined it with *left.jpg* to obtain a stitched scene including both pictures. The input images and the stitched result can be seen in Fig. 5.

Conclusion

In this assignment, we learned how to define, compute and visualize matching pairs of interest points between similar images. We have seen how an affine transformation can be computed with a matrix multiplication defined by joining the transformation components describing scaling, rotation and translation between coordinate spaces. We found out that, often, few iterations of Linear Least Squares solutions considering subsets of matching pairs are enough to find an accurate transformation. Comparing the transformation of our images obtained with matrix multiplication with the transformation returned using MATLAB's *affine2d* and *imwarp*, we found both results coherent with the expected one. One aspect to point out is that the transformation returns float coordinates which must be rounded to the nearest integer to match the discrete coordinate system of pixels. Finally, we learned how to use this transformation to stitch together a pair of images. An important step to consider in this operation is to compute the combined dimensions of the new stitched image to capture the whole content of it, adding padding before or after, if necessary.