UNIVERSITEIT VAN AMSTERDAM

COMPUTER VISION

ASSIGNMENT 4

IMAGE ALIGNMENT & STITCHING

Ahmet Taskale
Student Number:12344087
ahmet.taskale@student.uva.nl

Andreea Teodora Patra
Student Number:13365169
andreea.patra@student.uva.nl

Jim Wagemans
Student Number: 11912286
jimwagemans@gmail.com

October 5, 2020

1 Introduction

This assignment consists of two parts named image alignment and image stitching, where a function takes two images as input and computes the affine transformation between them and a function that takes two images and stitch them together.

Image alignment usually follows a scheme, which involves the Scale Invariant Feature Transformation (SIFT) and RANdom SAmple Consensus (RANSAC) method. These methods will be discussed in section 2.1. By applying the SIFT method, plotted image pairs are connected by matching points (keypoints). Furthermore, the RANSAC algorithm is performed to calculate the optimal affine transformation between the images after which the resulting transformations from image 1 to image 2 and vice versa, are visualised.

In the second part of the assignment, image stiching will be performed. As mentioned earlier, image stitching takes two images as input and computes a stiched image of the two. Similar to the process in the first part, the optimal transformation between input images needs to be found. Finally, the stiched image is visualised alongside with the image pair.

2 Image alignment

2.1 Question 1

1. To retrieve the keypoints from both the images, the SIFT function from CV2 is used. Keypoints are described as the points that do not change when the image is rotated or scaled. Once the points are found, a descriptor vector is created to be able to match them in both the images. To be able to match them, a brute force method from CV2 is used. SIFT algorithm follows 4 steps to be able to detect the keypoints: scale-space extrema detection, keypoint localisation, orientation assignment and keypoint descriptor. To make the detection scale-invariant, the images are computed are different scales. Therefore, in the scale-space extrema detection, SIFT used Difference of Gaussians to find the local extrema over scale and space. In this way, the same corners can be detected even if the images are scaled.

The scale space is sampled at n values. For some $\alpha > 1$, usually 1.6, it is sampled at

$$0 < i \leq n \ : s_i = \alpha^i s_0.$$

Next the difference of Gaussian function $D(x, y, s)$ is calculated at the sampled scales using

$$D(x, y, s_i) = f_{s_{i+1}}(x, y) - f_{s_i}(x, y)$$

where $f_{s_{i+1}}$ is the image scaled with the factor s_{i+1}.
Determining the extrema is done using the derivative. A different starting scale means a translation in the s coordinate. Translating and then taking the derivative is the same as taking the derivative and then translating. This explains why SIFT is scale invariant. The same goes for rotation. Taking the derivative and then rotating is the same as rotating and then taking the derivative.

Next step involved removing the low-contrast keypoints and edges. The low-contrast are removed by computing the Taylor series expansion of the scale space to get an accurate position of the extrema and a threshold is used to filter out the keypoints that have a lower intensity. The edges are removed using a Hessian matrix to compute the principal curvature.

Next the magnitude and orientation of the gradient vectors of points near the keypoint location is sampled . This is used to construct a histogram of orientations. Each gradient vector is weighted by magnitude and a Gaussian weight of $1.5s$. This makes it scale invariant. We now find the highest peak in this histogram and any peaks with a value of at least 80% of the highest peak. This is rotation invariant because a rotation would only shift the histogram.

The descriptor for the keypoint is created by taking a 16x16 neighbourhood and dividing it into 4x4 grids. In each cell, the histogram of orientations explained before is calculated. The histogram is represented as a vector and the descriptor is therefore, created. The descriptor vector is rotated to correct for the orientation of x and normalize it to correct for illumination.

2. In the image, a sample of 10 keypoint matchings is taken and displayed in both images.

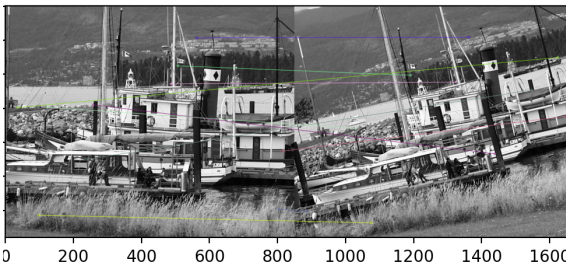

Figure 1. Sample of 10 keypoint matchings displayed in both images

3. To calculate the affine transformation between the images, the RANSAC algorithm is used. Iterating N times, P points are sampled from the total set of matches and the transformation parameters are then calculated using the equation presented in Figure 1 from the assignment. We determine the new shape of the transformed image using a function that looks at the transformed corners of the transformed image and estimated the size by taking the estimate of the difference between the min and max of the x and y coordinate. For better visualisation of the image, we translate the images using a translation of 50 in the x direction and 50 in the y direction.

In this figure, the warpAffine function from CV2 is used to visualise the transformation of the first image in the second one and the reverse as well. The inverse transformation was done by specifying a flag - WARP_INVERSE_MAP into the warpAffine function

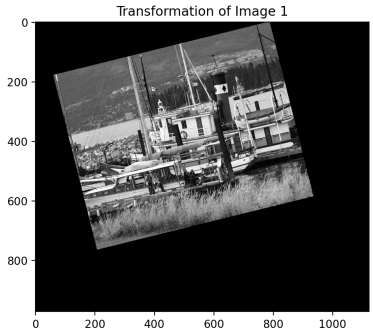

Figure 2. Transformation of first image of boat

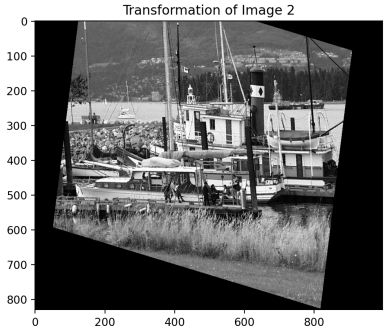

Figure 3. Transformation of second image of boat - inverse affine transformation

For the visualisation of the transformation of image 1, the translation of 50 in both directions was applied, however for the transformation of image 2 into image 1, it was not necessary. We can also visualise the connection of the keypoint matchings that have resulted using the RANSAC algorithm and can conclude that they lie close to each other in the images and it has a better outcome than using the brute force algorithm of matching.

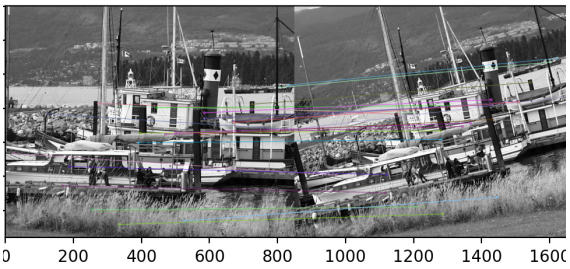

Figure 4. RANSAC connecting the keypoint matchings of transformed pixels coordinates

We also implement the nearest neighbour interpolation to visualise the transformations. The algorithms rounds the transformed coordinates to be able to get integer locations. We are using the Forward Warping method which starts from the source image and applies the affine transformation to get their position in the target domain. The pose of the target image is the same as the pose of the transformed image. However, using our own function of nearest neighbour interpolation, we can see that if we want to display the full transformed image, we will get values on the upper border with the same intensities as the source image. The Inverse Warping method was not implemented as the Forward Warping was performing as expected already.

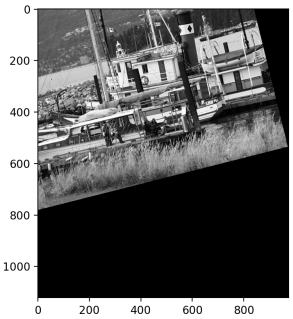

Figure 5. Transformation of first image of boat

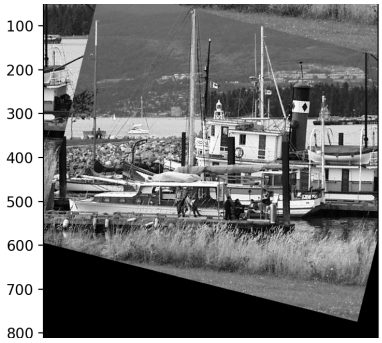

Figure 6. Transformation of second image of boat - inverse affine transformation

2.2 Question 2

1. To be able to get an affine transformation as formulated in the assignment, we need 3 matchings. Instead of using the 2D coordinates, we are using the homogeneous coordinates to be able to calculate the affine transformation. We can see the affine matrix is of the form 2x3 and therefore, we have 6 degrees of freedom. For each row, we can see that we have 3 unknowns and to be able to solve system, we need an equation for each row. Therefore, 3 points are needed to calculate the affine transformation.

2. In figure 1 from the assignment, we see that the vector x is of length 6. For n different matches, the matrix A is of size $2n \times 6$ and b is of size $2n$. In order to solve this system, A need to be at least 6×6 thus $n = 3$. This assumes that the points are linearly independent, otherwise A would not be invertible.

Next we discuss how many iterations of RANSAC we need to find a good transformation. We make this estimate a function of f, the fraction of matches that are inliers and α the probability that we will not find a good match. For the boat we estimate $f = 0.5$ and for the bus $f = 0.2$. We allow a failure probability of $\alpha = 0.01$. Now we discuss how to find n the minimum number of tests as a function of f, α. Since we need 3 points to estimate a affine transformation, the probability that we get at least one outlier is $1 - f^3$. The probability that we get at least one outlier in each of our n tests is $(1 - f^3)^n$. This means the probability of getting 3 inliers at least once after n tests is

$$1 - (1 - f^3)^n$$

This should be equal to $1 - \alpha$ (probability we find a good transformation). We now solve for n

$$
\begin{aligned}
1 - (1 - f^3)^n &= 1 - \alpha \\
(1 - f^3)^n &= \alpha \\
n \log\left(1 - f^3\right) &= \log(\alpha) \\
n &= \frac{\log(\alpha)}{\log(1 - f^3)}
\end{aligned}
$$

3 Image stitching

3.1 Question 1 and 2

As done in the previous question, first the optimal transformation between the input images needs to be found. For this part, two image pairs, including a boat (boat1.pgm and boat2.pgm) and a bus (left.jpg and right.jpg) are used for stitching. After estimation of the size of the stitched image, the image pairs are combined and computed as a single image.

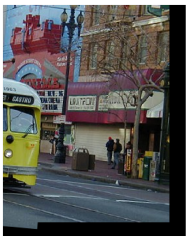

Figure 7. Resulting stitched image of left.jpg and right.jpg with estimated size of the image.

Figure 8. Resulting stitched image of left.jpg and right.jpg with a pre defined size (600, 700).

Figures 7 and 8 were obtained by setting the following parameters for the number of iterations:

- Fraction of inliers: estimated fraction of matches that are inliers
- Error margin: probability of not having three inliers when picking random matches (determines accuracy of perspective transform matrix)
- Shape of the method: basically the code has two ways of dealing with the size of the final stitched image, where 'function' is used for using the algorithm to determine the size of the resulting stitched image and 'precalculated' which pre-defines the size for the stitched image. In this case the size is set to (600, 700) for bus images.

Figures 7 and 8 were obtained by setting the fraction of inliers and error margin to 0.2 and 0.02, respectively. As for the shape of the method, Figure 7 was set to 'function' and Figure 8 to 'precalculated'. The algorithm used to estimate the size of the stitched image is based on the transformed corners of the right image which is the target. That is why, when using the algorithm we get a cropped version of the left image as right image contains only the front of the bus. The size (600,700) was estimated by taking into consideration the size of both transformed images which is around (300,400). For Figure 8, for better visualisation of the stitched image we use a translation of 300 in the x direction and 100 in the y direction.

As for the images with the boat, Figures 9 the fraction of inliers and error margin to 0.5 and 0.01, respectively.

Figure 9. Resulting stitched image of boat1.pgm and boat2.pgm with fraction of inliers and error margin set to 0.5 and 0.01, respectively.

Both images of the boat had similar landscapes, therefore the algorithm for estimating the size of the stitched image worked better in this case. The bus images required more number of iterations, a smaller number of fraction of inliers, because the intersection of the bus images was smaller compared to the boat ones. The error margin was set low for both cases.

4 Conclusion

For image alignment, based on the obtained results, it is concluded that the images shown in Figure 2 and 3 represent the optimal transformations for visualization. Furthermore, it was shown that at least three matchings are needed to get an affine tranformation. As for the discussion how many iterations is needed to solve an affine transformation, the reader is referred to section 2.2, where the equation shows the probability of getting at least three inliers once after n tests.

For image stitching, we have concluded that the parameters shape of the method, fraction of inliers and the error margin contribute to the optimal resulting stitched image. Figures 7 and 8 were obtained by setting the fraction of inliers and error margin to 0.05 and 0.01, respectively. As forthe shape of the method, Figure 7 was set to 'function' and Figure 8 to 'precalculated'. Figures 9 and 10 were obtained by setting the fraction of inliers and error margin to 0.05 and 0.01, respectively.