Computer Vision Assignment 4

Frederik Harder
10986847
frederikharder@gmail.com

Maartje ter Hoeve
10190015
maartje.terhoeve@student.uva.nl

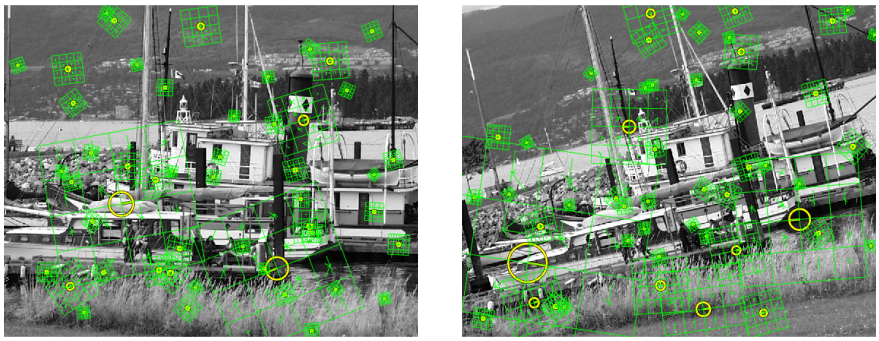
March 2016

For this assignment we implement a function to do image alignment. Matching points between two images are defined, after which the transformation matrix for going from the one image to the other image is calculated. We can use a similar approach when we want to stitch two images together.

1 Image alignment

In this part we perform the image alignment, following the algorithm that is given in the assignment description. We are given two images, see figure 1. In order to compute the transformation from the one image to the second image, we first use a MATLAB implementation of David Lowes SIFT to find the interest points in both images and to characterize the local appearance of the regions around the interest points. The MATLAB function to do this is `[frames, descriptors] = vl sift(im)`. Now we use `vl ubcmatch(descriptorsIm1, descriptorsIm2)` to get the key point matches. Figure 2 shows the two pictures with these matches. Now we can perform the RANSAC algorithm in order to get the best transformation matrix, as is explained in the assignment description. The final transformation matrix contains six unknowns at this stage. For this reason we need to get at least three matching points (i.e. six points in total, three in both images) in order to solve the equation. From this point, the more matching points we take, the more accurate the solution becomes. We use $N = 1000$ (i.e. 1000 repetitions) in order to get stable results as a standard value for the number of iterations. Note that for this particular image, this many iterations is not needed, as the matching points that are found are very accurate. So, in all cases we have tried, we only needed one iteration in order to get a good transformation matrix. At each points in the loop we do the transformation with the transformation matrix we have found at this point.

Figure 1: Boats



(a) subset of SIFT-features in image 1 (b) subset of SIFT-features in image 2

Figure 2

As we do many iterations, we show only one of these transformations in figure 3. After we have found the best transformation matrix, we transform the first image in two manners. First of all we use MATLAB's built-in functions and `imtransform`. Next to that we build our own function to do the transformation. For this we compute the expected size of the transformed image, based on the corner points. Then we use the inverted transformation on all pixels in the target image to map back to the original and find the closest neighbouring pixel by rounding. Pixels which don't map onto the image dimensions are left black. The results of both approaches can be compared in figures 4 and 5. In the latter, we perform the same for the transformation the other way around, i.e. from the second image to the first image.

2 Image Stitching

In this part we need to stitch the images shown in figure 6 together. For the image stitching we use a very similar approach to the approach taken in the previous part of the assignment. First of all we compute the transformation

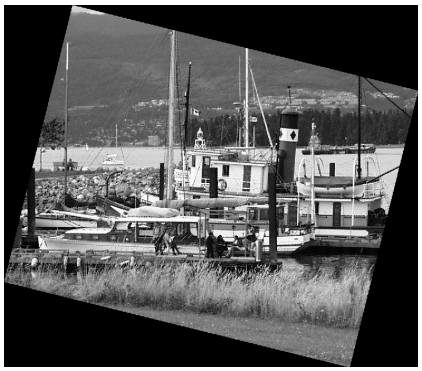
Figure 3: Subset of feature matches

(a) matlab functions

(b) own function

Figure 4: Transforming image 1 to match 2

(a) matlab functions



(b) own function

Figure 5: Transforming image 2 to match 1

Figure 6: Images to be stitched

from the one image to the second image. In this case we have less clear matches, as the images are less similar than the images we used in the previous part. For this reason we do need to use the thousand repetition set as a standard in the previous question. By doing this we get a proper transformation matrix. Now again, we need to calculate the size of the stitched image and put the the pixel values of both images at the correct point in the matrix, using the same approach as in part 1. However, in contradiction to the the first part of the assignment, now we have two coloured images. So, after doing all the transformation in gray scale (which is required by the MATLAB functions) we transform the pixels in the coloured image. The nearest neighbour interpolation is now done by taking the average for the r, g and b values of the surrounding pixels. The result is shown in figure 7.

Figure 7: stitched image