Computer Vision 1, Assignment 4

Erik Stammes Max Bos Lars Rigter Jan Zuiderveld

March 13, 2019

Introduction

In this assignment the focus is on the RANSAC algorithm [Fischler and Bolles (1981)]. Using SIFT feature descriptors and keypoint matching between these points, we first align two images and later stitch two images [Lowe (1999)].

1 Image Alignment

Question 1
A demo function to run our code is in `main.m`, it uses the functions `keypoint_matching.m` and `RANSAC.m`. An example of keypoint matches is plotted in Figure 1. Figure 2 shows the the transformation from `boat1.pgm` to `boat2.pgm`. Figure 3 shows the transformation from `boat2.pgm` to `boat1.pgm`. In both images, the results when using our custom nearest-neighbor interpolation and the built-in `imwarp` is shown. Effectively, the results of using `imwarp` and our own interpolation function yields the same result, though the `imwarp` function calculates the dimension of the transformed image to show the fully transformed original image, which is something we do not calculate in our custom nearest-neighbor interpolation method. We use the same dimensions as the original image.

Question 2
You need a bare minimum of 6 points to find a unique solution for the affine transformation below, because of the six unknowns m_1, m_2, m_3, m_4, t_1 and t_2. Every match consist of a pair of points and yields two equations e.g. from the upper and lower part of A. In order to solve this system of equations we need 6/2=3 matches.

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

On average, the RANSAC algorithm needs 36.548 iterations to find the best transformation parameters between `boat1.pgm` and `boat2.pgm`. The last section of `main.m` is code that calculates this number for an arbitrary number of iterations. We ran RANSAC 10.000 times to find the above value.

2 Image Stitching

`stitch.m` is a function that takes two images as input and stitches them together. To see the function in action, `stitch_demo.m` can be used, this also plots the resulting image. See figure 4.

3 Conclusion

In this assignments we have seen how to find interesting points and map them to a transformed e.g. affine or linear, version of the image.

Figure 1: 10 randomly selected keypoint matches between boat1.pgm and poat2.pgm images

(a) Using nearest-neighbor interpolation

(b) Using built-in imwarp

Figure 2: Transformation from 'boat1.pgm' to 'boat2.pgm' using nearest-neighbor interpolation and the built-in imwarp function

(a) Using nearest-neighbor interpolation

(b) Using built-in imwarp

Figure 3: Transformation from 'boat2.pgm' to 'boat1.pgm' using nearest-neighbor interpolation and the built-in imwarp function

Figure 4: Result of two images stitched together

4 Contribution

Everyone contributed equally. While the questions were divided amongst the team members, we cooperated on most questions and all came together to discuss the final result and gave feedback where needed. We worked on this together in the same room at the same time at the university. We all agree with the final submission.

References

Fischler, M. A. and R. C. Bolles
1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Lowe, D. G.
1999. Object recognition from local scale-invariant features. In *iccv*, P. 1150. Ieee.