Universiteit van Amsterdam

Report

Week 4

May 16, 2017

Students:
Tijmen van Dijk
11336404

Tim de Haan
11029668

Practicumgroep:
Group C

Course:
Beeldverwerken

Course code:
5082COVI6Y

Introduction

With 43000 citations the SIFT method by Lowe is an important contribution to the field of computer vision (Lowe, 2004). For this report we seek to use its image recognition power to build a mosaic out of two images, also using RANSAC to improve the accuracy of the image stitching. First, a theory section will describe the SIFT method in detail, and we will briefly touch upon some theorical considerations for the use of RANSAC. For the implementation of the SIFT method we will make use of the vlfeat library, detailing the necessary steps in the Matlab files accompanying this report.

SIFT

Scale-Invariant-Feature-Transform or SIFT is an algorithm for detecting stable features in an image, which has applications like object recognition, image mosaicing, and motion tracking. Below is a condensed description of the algorithm.

1. Detection of scale space extrema

The first step is locating keypoints at which distinct features may be found. Objects can consist of many different structures at different scales. For example, at a small scale the leaf structure of a tree is distinct, at a larger scale the branch structure becomes more prominent. Therefore, to make the method scale-invariant, it is necessary to detect features at a comprehensive range of scales, otherwise features are lost if the scale of an image changes. This is why SIFT uses 'scale space' and more precisely, linear Gaussian scale space L, which is computed as the convolution of an image with a Gaussian kernel, which has a variable scale parameter. This defines the scale space L at multiple scales, and thus over a range of Gaussian smoothing filters.

Within L, nearby scales are subtracted from each other to obtain the multiple 'difference of Gaussian' (DoG) function, which also contains a scale parameter. Conveniently, this difference between two levels of smoothing is a close approximation of the Laplacian of Gaussian, and even of a scale-normalized version.
The Laplacian of Gaussian is a blob detector whose local maxima and minima correspond

to stable features in an image. Consequently, the local maxima and minima of each scale level of DoG identify the stable features at that scale. These local extrema are computed by comparing a sample point with its 8 neighbours in its own DoG level, and with its 9 neighbours in the nearest DoG scales above and 9 neighbours below its own scale. This results in many potential keypoint locations over a range of scales. Each potential keypoint has a defined spatial location and a scale.

2. Keypoint localization

The locations for keypoints are now identified at a pixel level. To more accurately localize the keypoints, it is required to look at a sub-pixel level. This is done with a second-order Taylor expansion of the DoG function, shifted to the potential keypoint location. The derivative of this Taylor expansion set to 0 estimates the location of the extremum more accurately. Substituting this extremum back into the Taylor expansion gives a value to this extremum, this value signifies the contrast of the keypoint. Low contrast keypoint are unstable features and can thus be discarded.

The DoG function reacts not only to blobs, but also along edges. However, the locations along the edge are often highly sensitive to noise. As such they are not stable features in an image, and should be discarded. Keypoints at edges can be recognized by their curvature gauge. An edge will have a large principal curvature across the edge, and a weak principal curvature along the edge. These principal curvatures are proportional to the the eigenvalues of the Hessian matrix at the keypoint location. Comparing the ratio of these eigenvalues and discarding the keypoints with a ratio above a certain threshold will remove keypoints along edges.

3. Orientation assignment

Apart from the method being scale invariant, it would also be very convenient if the keypoints would not be lost after image rotation. To achieve rotation invariance, each keypoint is assigned an orientation based on its local neighbourhood. At the level of L which is closest to the scale of the keypoint, a gradient magnitude and orientation is calculated from the differences in pixel values of adjacent points, not only for the keypoint itself, but also for each point around it. The size of this neighbourhood depends on the scale.

Orientations of the points within this neighbourhood are collected into a histogram, consisting of 36 bins covering a total of 360 degrees. Before adding, these orientations are weighted by their gradient magnitudes and by a Gaussian filter, so that closer points contribute more to the final orientation than far-away points. The orientation that receives the highest peak in the histogram is considered the keypoint orientation. If there are other peaks that lie within 80% of the value of the highest peak, the keypoint is split into multiple keypoints with the same location and scale, but with different orientations. Now a keypoint can be represented relative to its orientation, so it can be recognized even if rotated. This makes the method invariant under rotation.

4. The local image descriptor

To further improve the stability of the keypoints as distinct and easily recognizable image features, each keypoint is transformed into a feature vector. This is once again done at the level of L which is closest to the keypoint scale. In a 16 x 16 pixel neighbourhood around the keypoint is further divided into 16 4x4 regions. An orientation histogram is computed for each of these regions much like the keypoint orientation assignment, however, this time each histogram is divided into 8 bins. The values in each bin for 16 regions is combined into a feature vector, or image descriptor, consisting of $8 \times 16 = 128$ elements.

Some adjustments are still required. Firstly, the feature vector is normalized. Secondly, to ensure the feature vector has no abrupt changes at the borders the regions, trilinear interpolation is used to smooth transitions. Thirdly, to improve invariance under minor changes in illumination, large gradient magnitudes are thresholded. This decreases their importance relative to gradient orientations, which are unaffected by changes in linear illumination.

This finalizes the SIFT method, which as transformed an image into highly distinctive feature vectors. These feature vectors are invariant to changes in image scale and rotation, and are stable enough to be recognizable under considerable degrees of noise, occlusion, and changes in perspective and illumination.

RANSAC

The steps in our implementation of the RANSAC method have been detailed in the Matlab code accompanying this report. There is one theoretical consideration to discuss here:

Amount of iterations

To compute an estimate of the required amount of iterations N for a good result we apply this formula:

$$N = \frac{\log(1-p)}{\log(1-(1-e)^s)}$$

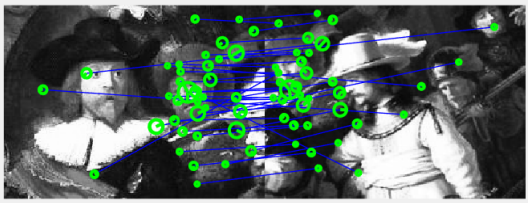
With e being the probability that a point is an outlier, p being the desired probability for a good sample, and s being the minimum number of points in a sample for our model. For a projective transformation, $s = 4$. A visual estimation from the plotted matching points in our implementation gave us 7 outliers for 41 matches, so $e = \frac{8}{41} \approx 0.2$.

Applying the formula with $p = 0.99$ and these values results in $N = 8.7 \approx 9$.

Algorithm

The mosaic construction has been implemented in Matlab and the heavily commented code has been provided with this report. The *main.m* file contains the segmented code so it can be run separately for assessment by the reader.

Experiments



This is a visualization of the matches provided by vl_feat for the Nachtwacht images.

Below is the answer to the 5 points assignment from the SIFT section.

The estimation of the projective matrix based on 4 matches computes these locations for the remaining matches (without RANSAC). Bad matches are also plotted and there are some outliers visible on the far right or bottom. Many points do not seem to lie on any visually salient image feature.

Estimating a projection matrix on the basis of handpicked 'bad' points gives this result. Surprisingly, the points end up in a region where there should be matches, but they have formed a strange sort of line pattern which seems to contain few good matches.

This is the end result of our mosaic construction, detailed in Matlab, using the values for RANSAC as described in the RANSAC section. It seems quite accurate.

Conclusions

The SIFT method's estimation of distinct image features has shown to be easily powerful enough for correctly mosaicing two images together. A small percentage of matches by the SIFT method were bad, but the RANSAC method is capable of designating them as outliers, and succeeds in fitting a projective transformation that is reasonably correct with a small number of iterations.

Appendix

Some of the theory questions in this Appendix have been touched upon in the description of the SIFT, but as this report remained within the length limits, we restate the entire answer here. In some cases these answers are more specific.

Theory Questions

1.

2.

3. 'Scale space theory' concerns itself with handling signals at various scales. In our context of image processing, 'scale space' refers to the linear Gaussian scale space L, which is computed as the convolution of an image with a Gaussian kernel. This Gaussian kernel has a variable scale parameter, so L is defined at multiple scales, and thus over a range of smoothing filters.

   The difference between subsequent levels of the scale space L, the Difference of Gaussians (DoGs), are a close approximation of the scale-normalized Laplacian of Gaussian. The Laplacian has its extrema at strong transitions in brightness or possibly color, so 'scale space extrema' are blobs, corners and edges. These make up the strong, stable features which characterize an image.

4. The missing step is substituting the scale-normalized Laplacian of $\sigma^2\nabla^2 G$, multiplied with constant factor $(k-1)$ back into the original definition of the difference of Gaussians D, where it gets convolved with the original image i.

$$D(x, y, \sigma) = (k-1)\sigma^2\nabla^2 G * I(x, y)$$

5. It is evident that the Hessian matrix (H) is equal to its own transpose, thus the Hessian matrix is symmetric. If it is symmetric, then it is orthogonally diagonalizable, so there exists an orthogonal matrix Q such that

$$H = QDQ^T$$

   where D is a diagonal matrix with the eigenvalues of H on its main diagonal. Subsequently, we use that Q is an orthogonal matrix, $Q^{-1} = Q^T$, and thus

$$\begin{aligned} \text{trace}(H) &= \text{trace}(QDQ^T) \\ &= \text{trace}(QQ^T D) \\ &= \text{trace}(D) \end{aligned}$$

   By the aforementioned properties of D, this means that the trace of H is equal to the sum of its eigenvalues.

6. - 5 Octaves
   - 4 DoG
   - 128 elements in the feature vector
   - Smoothing scales at multiple occasions.

7.

8. Rewriting the notation of Lowe into a more familiar notation using ∇_D to denote the gradient of D and H_D to denote the Hessian matrix of D gives

$$D(\mathbf{x}) = D + \mathbf{x}^T \nabla_D + \frac{1}{2}\mathbf{x}^T H_D \mathbf{x}$$

for equation 1 and

$$\hat{\mathbf{x}} = -{H_D}^{-1}\nabla_D$$

for equation 2. Substitution gives

$$D(\hat{\mathbf{x}}) = D + (-{H_D}^{-1}\nabla_D)^T\nabla_D + \frac{1}{2}(-{H_D}^{-1}\nabla_D)^T H_D(-{H_D}^{-1}\nabla_D) \quad (1)$$

$$D(\hat{\mathbf{x}}) = D - ({H_D}^{-1}\nabla_D)^T\nabla_D + \frac{1}{2}({H_D}^{-1}\nabla_D)^T(H_D{H_D}^{-1})\nabla_D \quad (2)$$

$$D(\hat{\mathbf{x}}) = D - {H_D}^{-1}{\nabla_D}^T\nabla_D + \frac{1}{2}{H_D}^{-1}{\nabla_D}^T\nabla_D \quad (3)$$

$$D(\hat{\mathbf{x}}) = D - \frac{1}{2}{H_D}^{-1}{\nabla_D}^T\nabla_D \quad (4)$$

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}(-{H_D}^{-1}\nabla_D)^T\nabla_D \quad (5)$$

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\hat{\mathbf{x}}^T\nabla_D \quad (6)$$

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}{\nabla_D}^T\hat{\mathbf{x}} \quad (7)$$

Step 2 uses that ${H_D}{H_D}^{-1} = I$. Step 3 uses the fact that the Hessian is a symmetric matrix. The inverse of a symmetric matrix is also a symmetric matrix, so $({H_D}^{-1})^T = {H_D}^{-1}$.

Step 7 uses the commutative property of the dot product. ${\nabla_D}^T\hat{\mathbf{x}} = \nabla_D \cdot \hat{\mathbf{x}} = \hat{\mathbf{x}} \cdot \nabla_D = \hat{\mathbf{x}}^T\nabla_D$

9. We are in '3D' because every keypoint, aside from having spatial coordinates, has also been assigned an orientation (θ). Therefore we need trilinear interpolation with the dimensions being (x, y, θ).

10. He means changes in illumination that are linear. Changes in illumination that result in all pixels having their brightness increased do not alter gradient values, so the features are resilient under those. He contrasts these with more complex changes in illumination in the next line, which have a considerable effect on large gradient magnitudes, which is why these are thresholded to provide some stability.

Bibliography

- Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.
- www.vlfeat.org