Beeldverwerken | Yorick de Boer, 10786015
Mosaicing by SIFT | Lucas van Berkel, 10747958

Contents

1 Introduction **2**

2 Theory **2**
2.1 SIFT . . . 2
2.2 RANSAC . . . 4

3 Algorithm **5**

4 Experiments **5**

5 Conclusions **5**

6 Appendix **5**
6.1 Theory Questions . . . 5
6.1.3 . . . 5
6.1.4 . . . 6
6.1.5 . . . 6
6.1.6 . . . 6
6.1.7 . . . 6
6.1.8 . . . 7
6.1.9 . . . 7

1 Introduction

This article is made in to clarify the way algorithms such as SIFT (*Scale Invariant Feature Transform*) and RANSAC (*Random Sample Consensus*) perform. For those unkown with either of these two algorithms, we will shortly discuss the functionality of the two, after which we will discuss the framework behind the algorithms.

Scale Invariant Feature Transform (*SIFT*) is an algorithm capable of recognising objects in a picture (only after given a training-set). It can also match different characteristics of two pictures, in order to combine them. The latter functionality is the one we will use in order to show the power of the SIFT. In our problem, we are given two pictures of one greater picture, in our example 'De Nachtwacht' from Rembrandt van Rijn. The algorithm will calculate 'key-points' in both pictures and match them, in order to calculate the relation between the two pictures. Key-points are locations in a picture which are of special interest to us, such as corners and planes. When two pairs of key-points are matched the difference in rotation and translation of the two pictures is calculated, after which the greater picture can be formed.

Random Sample Consensus (*RANSAC*) is a optimization algorithm which is able to detect patterns in a data-set with a significant amount of *outliers*. The idea behind the algorithm is quite simple. If the problem is to create a reasonable estimation of a line in a two-dimensional field with a lot of outliers, the algorithm will select n random points. Next these points are fitted on a line. This has a certain range and which creates a field of inliers around it. This process of picking random points and estimating inliers is done repeatedly until finally the line with the most inliers is selected as the final model. The RANSAC algorithm will be used in our problem in order to find good pairs of matches, so the rotation and translation can be calculated. Through this article we will both handle our problem and explain why the script does what it does. That way we hope you will understand the framework.

2 Theory

This chapter is meant to clarify the two algorithms.

2.1 SIFT

The SIFT algorithm changed the subject of computer vision ever since. The algorithm was found by David G. Lowe, explained by the same person in an article of 2004. Important to know is that the SIFT algorithm identifies and classifies key-points in a picture. After the classification of those key-points, different pictures can be compared on similarity. The SIFT algorithm can be divided into four parts.

1. **Scale-space extrema detection**
2. **Key-point localization**
3. **Orientation assignment**
4. **Key-point descriptor**

Scale-space extrema detection

The detection of extrema in a picture is an art in computer vision which is not very difficult. The picture is blurred by a Gaussian filter, after which the derivative of the picture is calculated. Spots that are a minimum of maximum compared to its neighbours, are a local extreme. But this operation is dependent on the size of the scale of the Gaussian filter. Making the scale a variable in a Gaussian,$G(x, y, \sigma)$, this problem may be handled. Earlier research has shown that the scale-normalized laplacian of a Gaussian, $\sigma^2\nabla^2 G$, is needed for true scale-invariance for key-point detection. Lowe has shown that the scale-normalized laplacian of a Gaussian can be calculated by taking the difference-in-Gaussian. Lowe uses this fact to calculate the difference-in-Gaussian with a constant multiplicative factor k: $D(x, y, \sigma)$(Difference-in-Gaussian)= $G(x, y, k\sigma) - G(x, y, \sigma)$, and therefore the scale-normalized laplacian of a Gaussian. This is why SIFT is scale invariant. f

Keypoint localization

From the many candidates for key-points derived from the scale-space extremes, a selection has to be made in order to obtain the key-points relevant to the description of the picture. Lowe solves this problem by calculating a

central maximum by interpolating after a 3D quadratic function. This method was initiated by Brown and it uses a second-order Taylor expression. If the offset of the vector $\hat{x}$ is larger than 0.5 in any dimension, another sample point is closer to the extreme. The new sample point is interpolated and the final offset is added to the location of the extreme. Lowe also eliminates key-points which have greater ratio in principal curvature than a certain threshold. The ratio in curvature is calculated by using the hessian of a candidate. The hessian of the candidate is:
$\begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$
The eigenvalues are proportional to the principal curvatures of D. The trace is the same as the sum of eigenvalues and the determinant shows the product of the eigenvalues. So the ratio of principal curvatures is calculated via:
$\frac{Tr(H)^2}{Det(H)} = \frac{(r+1)^2}{r}$
if a keypoint has a principal curvature ratio larger than r, the keypoint is eliminated.

Orientation assignment

Orientation assignment is important in order to make the key-point detector invariant to image rotation. Lowe solves this quite easy by calculating the gradient magnitude and orientation using the pixel differences. An orientation histogram is made of 36 bins, covering the 360 degree range of orientations (steps of 10 degrees). Peaks in the orientation histogram correspond to the dominant directions of the local gradients. If second largest peak is 80% of the largest peak, a new key-point is made with the second largest peak as dominant in the direction, so a key-point on the same location only with a different orientation. This way the key-point detector is, besides scale invariant, also rotation invariant.

Key-point descriptor

The final step in order to compare local key-points in an image is to find a sufficient representation to save different key-points in to a database. Lowe was inspired by a approach found by Edelman, Intrator and Poggio, who was able to recognise 3D-objects rotated 20 degrees in depth with 94%, instead of the 35% for correlation of gradients. The descriptor representation of Lowe uses the image gradient magnitudes and orientation of sampled points around the key-point, which are combined into feature vector which determines the key-point. A grid of a 4x4 array of histograms with 8 orientation bins each is used so not too much information about the neighbourhood of the key-point is lost. The feature vector of the key-point is also normalized so change of illumination does not interfere with the description. The results of this description are very promising, since the key-points are still recognised 50% of the time if the 3D-object is rotated 50 degrees in depth. The overall recognition of objects of the SIFT method lies around 80%.

Using this technique, recognising and matching objects is much easier. In our problem, we need to obtain the key-points. The figure below describes 40 random key-points in both pictures of *De Nachtwacht*.

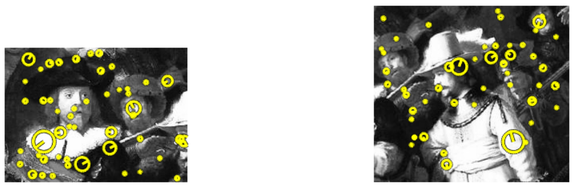

Figure 1: 40 Random keypoints in both pictures

Luckily for us, the SIFT algorithm also matches key-points, which results in a more hopeful result.

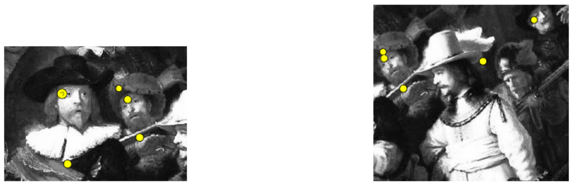

Figure 2: 5 Best matches of keypoints in both pictures

Now, given the code to construct a combined mosaic if only given the projectionmatrix to transform one picture into the other, we should be able to construct our mosaic. Unfortunately, the matches given by SIFT are not perfect. Choosing the four best match-pairs to construct the projectionmatrix, gives us a rather unsatisfiable result. But choosing the 27th best till 31st best, the result is quite pretty, but still not perfect (and we defined the best match ourselves). Maybe RANSAC will be able to help us with this problem.

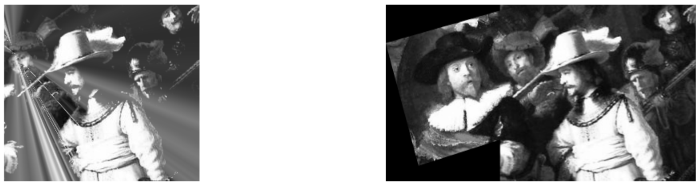

Figure 3: Left: 1st till 4th best match. Right: 27th till 31st best match.

2.2 RANSAC

As stated earlier, RANSAC (*Random Sample Consensus*) is a optimization algorithm in order to detect a pattern in a data-set with rather much *outliers*. Before we start implementing the algorithm, it is important to understand the pseudo-code. Since every data-set derived from a experiment is not perfect, using least squares will give you a estimate which will not suffice, it will take outliers into account. RANSAC filters this data-set, calculates an estimate using least squares and tests the estimate afterwords with the entire data-set. Some variables are needed in this process.

First you decide how large your filtered data-set must be. For a linear line in a 2D-problem, this number is usually 2(first variable). A line is fitted over these two numbers and every data-point in the original data-set is compared to the line with a error-rate. If the error-rate is lower than a certain threshold(second variable), the data-point is considered an inlier of the filtered data-set. After all data-points in the original data-set are tested, the number of inliers must exceed a second threshold(third variable) in order to determine if the estimate is a reasonable model. The estimate is optimised for the current inliers and the total error-rate is calculated. If the new model is better than the current best model, the new model will replace the best model. After a certain number(fourth variable) of iterations the best fitted model is returned.

Four variables have to be filled in. The first variable(let's call it n) is the amount of data-points a model has to be fitted in order to obtain a reasonable model, luckily for us, the number for our problem(rotation, scaling, translation) is 2. The second variable(h) is the amount of iterations before stopping to search. That variable is usually calculated by $(1-q)^h \leq \epsilon$, where q is the probability a data-set produces an accurate model(how big the chance is that a data-point is a inlier to the power of n, so P_{inlier}^n) and ϵ a certain alarm-rate, which may be chosen and is usually very small. The third variable(d) is the amount of inliers in a filtered set, for it to be a good model. This variable is calculated by $d = Size_{data-set} \cdot P_{inlier}$, since this threshold is only reached with a perfect model, usually fraction of it is taken. The last variable(t) is the threshold for which data-points are considered inliers over outliers. This variable may be guessed, such as the alarm-rate may be guessed. With these information of the pseudo-code of the RANSAC algorithm, we can now start implementing it.

3 Algorithm

There have been two versions of RANSAC designed. The main difference between the two is the way in which they handle the error. And therefore how they determine what an inlier is and what is not.

The first algorithm, implemented in Matlab calculates the error from the projection matrix. It transforms the matched points from one image to the other and checks how close they are to the actual match in the second image. The algorithm is fully described as code with comments so no further pseudo code is made.

The second algorithm is one we were not able to implement because of lack of time. It needs two random points where it creates a line between. Next it calculates the distance between all points and the line in both images. The error is then the relative difference of distance between the distance in one image compared to the distance in the other image. If the distance is passed a certain threshold it is added to the list of inliers. Where the projection matrix is calculated from. Finally after all iterations the best projection matrix is returned.

```
function [projectionMatrix] = ransac(matchCoor1, matchCoor2, thresholdLine)

bestError = inf
for iterations {
    2randomCoordinates1 = rand(matchCoor1)
    line = drawLine(2randomCoordinates)
    inliers = empty
    if matchCoor2 distance < tresholdLine {
        inliers add matchCoor1
    }
    tempError = sumAllErrors()
    if totalError < tempError && inlierCount > bestInlierCount {
        projectionMatrix = createProjectionMatrix(inliers)
        bestInlierCount = inlierCount
        bestError = tempError
    }
}

end function
```

4 Experiments

A number of experiments would have been performed with SIFT and RANSAC on different pairs of pictures if our RANSAC algorithm had worked correctly. Unfortunately this was not the case. So the experiments could not be executed.
As stated before we did construct a pretty good mosaic without the help of RANSAC, this can be seen in figure 3.

5 Conclusions

A complete overview of the RANSAC algorithm has been given. Unfortunately we were unable to complete our RANSAC algorithm which resulted in the fact that we were unable to to do the experiments.

6 Appendix

6.1 Theory Questions

6.1.3

For 2-dimensional images a *scale space* is the convolution of a image by a Gaussian kernel. For every octave (the image scaled down) there is a separate scale space. The size of the scale is the variance of the Gaussian filter. If the value of a point is the minimum or maximum compared to its neighbours it is a extrema. By using *scale space*, different visual operations become *invariant*, such as calculating the extrema.

6.1.4

$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma) \end{aligned}$$

Lowe offers the heat diffusion equation as the relationship between D and $\sigma^2\nabla^2 G$, the heat diffusion equation is the following with the usual t replaced by σ^2:
$\frac{\partial G}{\partial \sigma} = \sigma\nabla^2 G$
The normal derivation formula follows:
$\frac{\partial G}{\partial \sigma} = \sigma\nabla^2 G \approx \frac{G(x,y,k\sigma)-G(x,y,\sigma)}{k\sigma-\sigma}$
By transferring the $k\sigma - \sigma$ to the other side of the equation, the same formula of $D(x,y,\sigma)$ appears. That is the relationship between D and $\sigma^2\nabla^2 G$.

6.1.5

Since the hessian matrix is symmetric, the matrix must be diagonalizable. Let the hessian be matrix H. The diagonalized matrix is a NxN matrix with the eigenvaulues over the diagonal.

$$PHP^{-1} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$$

The trace of the left part of the equation is equal to the right part of the equation. So

$$\begin{aligned} Trace(D) &= Trace(PHP^{-1}) \\ &= Trace((PH)P^{-1}) \\ &= Trace(H(PP^{-1})) \\ &= Trace(HI) \\ &= Trace(H) \end{aligned}$$

Since the trace of the diagonalized matrix is the same as the sum of the eigenvalues, you may conclude that the trace of the hessian is equal to the sum of the eigenvalues.

6.1.6

Page 10:
However, there is a cost to using a large σ in terms of efficiency, so we have chosen to use σ = 1.6, which provides close to optimal repeatability.
We will not have to change this value, because it's optimal like Lowe said.
Page 12:
The experiments in this paper use a value of r = 10, which eliminates keypoints that have a ratio between the principal curvatures greater than 10. We would only need to change this if the ratio between the principal curvature is very much different.
Page 16:
Therefore, we reduce the influence of large gradient magnitudes by thresholding the values in the unit feature vector to each be no larger than 0.2, and then renormalizing to unit length.
No need to change this, Lowe determined this value expirimentally.
Page 20:
For our object recognition implementation, we reject all matches in which the distance ratio is greater than 0.8,
Might need to change this for different types of images.

6.1.7

$D(x) = D + \nabla D x + \frac{1}{2}x^T\nabla^2 D x$
$x = -\nabla^2 D^{-1}\nabla D$
Substituting (3) into (2)
$D(x) = D + \nabla D - \nabla^2 D^{-1}\nabla D + \frac{1}{2}x^T\nabla^2 D - \nabla^2 D^{-1}\nabla D$
$D(x) = D + \nabla^2 D - \nabla^2 D^{-1} + \frac{1}{2}x^T\nabla^2 D - \nabla^2 D^{-1}\nabla D$

$D(x) = D + \frac{1}{2}x^T \nabla D$
$D(x) = D + \frac{1}{2}\nabla D x$

6.1.8

The meanings of the three dimensions are:

1. orientation
2. length
3. position

With these three dimensions we can describe the key point descriptor best.

6.1.9

A brightness change by a constant value does not influence the gradient values, because they are based on pixel differences. The descriptor is based on the gradient values and will not change either.