R&I Research Note

R&I N 30/2007

Thomas Bakken

An evaluation of the SIFT algorithm for CBIR

R&I Research Note	**N 30/2007**
Title	**An evaluation of the SIFT algorithm for CBIR**
Author(s)	Thomas Bakken
ISBN / ISSN	/0809-1021
Security group	Open
Date	2007.08.16

Abstract

This research note describes SIFT (Scale Invariant Feature Transform) algorithm, and tries to uncover suitable application areas for the algorithm. The algorithm is specifically tested for its feasibility for finding matches between mobile phone camera images of magazine advertisements and the original advertisements. SIFT has a good hit rate for this application, but the algorithm does not scale when considering time used to compare images.

The note also suggests a method to improve the quality of the image match results.

Keywords

SIFT (Scale Invariant Feature Transform), CBIR (Content Based Image Retrieval), NNS (Nearest Neighbour Search), QoM (Quality of Match)

Preface

Telenor R&I is currently involved in several projects related to image search and image retrieval. Essential to many of these projects is the need for a method to compare the similarity of images based on image content. SIFT (Scale Invariant Feature Transform) has been identified as a promising algorithm for this application.

This research note describes SIFT and tries to uncover suitable application areas for the algorithm. The note also suggests a method to improve the quality of the image match results.

The study is a part of the CAIM-project [1] and has been financed through a Telenor-fund dedicated to collaboration activities between Telenor R&I and the University of Tromsø (UiT). The work has been conducted by Thomas Bakken, under the supervision of Sigmund Akselsen and Anders Schürmann.

Contents

1 Introduction

We shall in this paper present the results of examining an algorithm called Scale-Invariant Feature Transform (SIFT) [2]. SIFT is an image processing algorithm which can be used to detect distinct features in an image. Once features have been detected for two different images, one can use these features to answer questions like "are the two images taken of the same object?" and "given an object in the first image, is it present in the second image?"

1.1 Overview

In this first chapter, the introduction, we will start by offering a motivation for the research. Then we will elaborate the goals of the research, before a list of definitions is given.

In the second chapter, we will point out how this research project is linked to other projects, and discuss the contributions the SIFT algorithm might make to a CIBR service.

The third chapter will focus on the SIFT algorithm and image searches. We shall have a look at the four stages of SIFT as well as how SIFT can be used for object detection in images.

Chapter four will present a variety of results and findings gained from testing the algorithm under various circumstances. Furthermore, all results will be discussed in the order they are presented.

In the last chapter, we will review the most important findings and present a conclusion for the research.

1.2 Motivation

David Lowe, at University of British Columbia, is the creator of the SIFT algorithm. Lowe speaks highly of how SIFT can be used to detect similar objects in two different images. The SIFT algorithm is supposedly able to identify two object as similar even when the object is partly concealed in either one of the images, has changed orientation, or the object is viewed at different angles.

Since we are very interested in all algorithms that are able to detect objects in an image, and especially interested in algorithms able to detect obscured and/or blurred objects, a close examination of the SIFT algorithm is very much in place.

In particular, if we find that the SIFT algorithm performs well, it could, for example, be incorporated in the M2S service [3], a part of the MOVE[1] project [4]. We could then take the next step and see how well the algorithm performs in a real-time system.

[1] *MOVE is a project which field of research is primarily the applications of mobile devices. MOVE has been further developed in the CAIM project*

1.3 Goal

The primary goal of this research is to determine if the SIFT algorithm is suitable to use as part of a CBIR service.

The primary goal can be split into several sub-goals. They are as follows:

- Examine the inner workings of SIFT
- Discover important characteristics of the SIFT algorithm
- Discover important characteristics of image matching when using SIFT
- Determine under what circumstances SIFT works
- Determine under what circumstances SIFT fails

1.4 Definitions

Here we list words that have a special meaning in this paper, and their corresponding definition.

Compared image The image currently being compared with the source image.

Difference of Gaussian An algorithm for detecting edges in an image.

DoG see Difference of Gaussian

Edge An edge in an image is an area with great contrast, and the contrast must go in a straight or curved line.

Fine-scale feature A fine-scale feature is a feature of an image only visible only at a certain scale.

kd-Tree A k-dimensional tree, where $k \in \mathrm{N}^+$. The kd-Tree is used when matching two sets of keypoints. It has a faster running time than that of linear search.

Keypoint A point in an image that SIFT has identified as distinct based on certain characteristics of this point.

Keypoint descriptor A 128-dimensional vector that describes all features of a keypoint.

Keypoint match Two keypoints that has been identified as similar.

Scale A scale is the ratio between the original image size and the size an image is being represented at. A scale of 1 is the image itself. A scale of 0.5 or 2 would be the image at half and double the size, respectively. A double scaled image would have the ratio 2:1, meaning 2 units of measure in the scaled image equals 1 unit of measure in the original image.

Scale-Space Theory for identifying image structures using different scales of an image. The goal of scale-space theory is to offer ways to suppress fine-scale features.

SIFT Scale-Invariant Feature Transform, the algorithm we have analyzed and tested.

Source image The query image that is, the image we wish to try find a match to.

Nearest Neighbour Search In our case, given a node n in a kd-tree, a search for the nearest neighbour of n is a search for the node m closest to n.

NNS see Nearest Neighbour Search

Quality of Match A number associated with a set of keypoint matches. The Quality of match reflects the likelihood of the matches being correct.

Repeatability Number of times a keypoint is repeatedly represented across different image scales.

2 Background

Earlier work relating to CBIR technology, in the CAIM and MOVE-projects, include the M2S service [3]. M2S is an acronym for MMS to Search, and is a prototype service for imaged based information retrieval. In this section, we shall identify problems related to object detection algorithms and the M2S service.

2.1 A Mobile Environment Service: M2S

One reason for examining the SIFT algorithm is to find out whether it can be used as part of a context aware mobile environment. In particular, the M2S mobile environment service is of interest to us.

The target users of the M2S service are tourists. That is, a typical scenario where the M2S is thought applicable involves a tourist, an advertisement found in a tourist magazine, and a camera phone. An interested tourist captures an image of the advertisement he or she likes, creates an MMS with the image as content, and sends it to a given four digit phone number. The M2S service must then analyze the incoming data and detect what the image depicts. The idea was then to have M2S reply with, for example, a video stream, textual information, or perhaps more images.

The success or failure of the M2S service relies a great deal on how accurately the identification process is able to detect the content of images. From what the author has learned, the best results that could be produced with the object detecting algorithms used in M2S had a correctness of about 20%. In other words, on average, 1 out of 5 content identifications were successful.

For testing purposes, this might be quite good. However, the results are far below satisfactory to be used for commercial purposes. The 20% correctness of the service implies that four out of five times, the tourist will get the wrong information. And four out of five times they will receive information on a competing tourist firm, rather the one they searched for. Few companies would want to pay money for offering information on a competing company 80% of the time.

It is evident that one needs a good object detection algorithm for image content analysis. If we could obtain an image searching and matching accuracy of 95%, M2S could offer a service which customers would probably find satisfactory.

3 The SIFT algorithm

The SIFT algorithm identifies features of an image that are distinct, and these features can in turn be used to identify similar or identical objects in other images. We will here give an introduction to the SIFT algorithm.

3.1 Introduction to SIFT

SIFT has four computational phases. The reason for this being that some computations performed by SIFT are very expensive. The cost of extracting the keypoints is minimized by the cascading approach of SIFT. The more expensive operations are only applied on locations that pass an initial, cheaper test.

The output of the SIFT algorithm is a set of keypoint descriptors[2]. Once such descriptors have been generated for more than one image, one can begin image matching (see Figure 1).

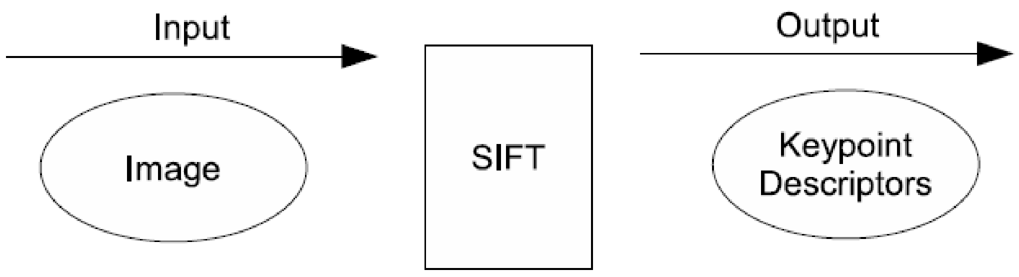


Figure 1: SIFT takes as input an image, and generates a set of keypoint descriptors. The keypoint descriptors may then be stored in a separate file.

The image matching, or object matching, is not part of the SIFT algorithm. For matching we use a nearest neighbour search (NNS), an algorithm that is able to detect similarities between keypoints (see Figure 2). Thus, SIFT only makes matching possible by generating the keypoint descriptors.

Although the matching process is not part of SIFT, we will use results from image matches as an indicator of how well the SIFT algorithm is suited for image matching. However, we shall distinguish between SIFT and matching when, for example, testing the running time of SIFT.

The SIFT implementation used for all tests presented in this paper is a C# implementation by Sebastian Nowozin. See [6] for where to find the implementation.

[2] Keypoints and their respective descriptors are discussed further in the next section.

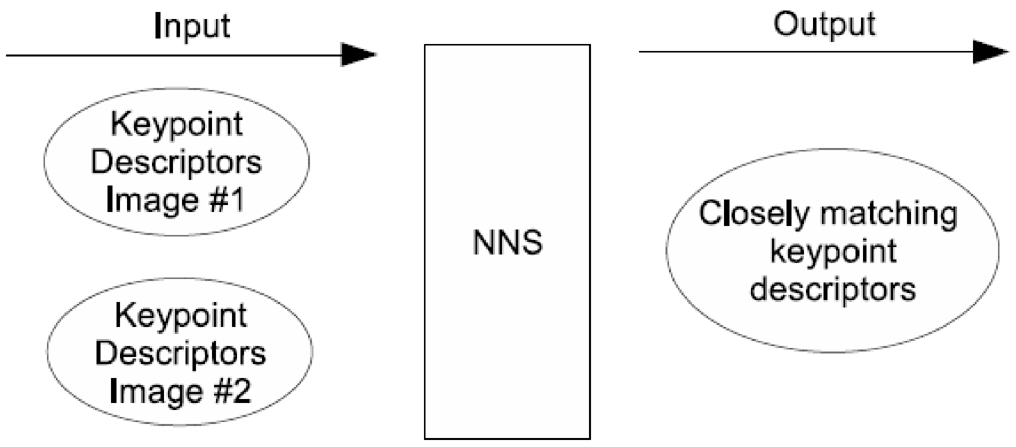

Input
Keypoint Descriptors Image #1
Keypoint Descriptors Image #2
NNS
Output
Closely matching keypoint descriptors

Figure 2: When we check for an image match, the two sets of keypoint descriptors are given as input to a nearest neighbour search algorithm. The output of the algorithm is a set of keypoint descriptors found to be very similar.

3.2 Keypoints and keypoint Descriptors

As already mentioned, the SIFT algorithm produces keypoint descriptors. A keypoint is an image feature which is so distinct that image scaling, noise, or rotation does not, or rather should not, distort the keypoint. That is, given a keypoint in an image, if one scales the image to half the size, or double the size, the keypoint would still be identifiable. The same goes for image rotation and noise. If an image is, for example, rotated clockwise, the keypoint would still persist.

A keypoint descriptor is a 128-dimensional vector that describes a keypoint. The reason for this high dimension is that each keypoint descriptor contains a lot of information about the point it describes. We shall in the next section have a closer look at what information the keypoint descriptors hold when we discuss the four phases of SIFT. To illustrate what a keypoint is, let us have a look at some images where keypoint descriptors have been detected. In Figure 3, we see a blurry image in which SIFT has detected 240 keypoints (The image can also be found in the lower right hand corner of Figure 4). The blue circles represent a keypoint found, along with its scale. Namely, the size of the circle represents the size, or scale, of the 128-dimensional vector.

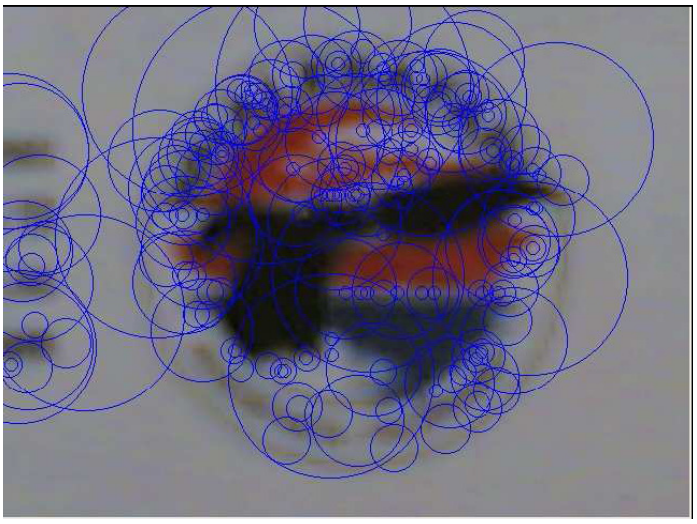


Figure 3: An image processed by the SIFT algorithm. 240 keypoints have been identified.

The 240 keypoints is a fairly small amount, for the SIFT algorithm to detect. The resolution of the image is 320x480, and such an image usually produce about 1500-2000 keypoints. The reason for the small amount of keypoints is the quality of the image. Since the image is very out of focus, SIFT cannot find too many points in the image that are resistant to scale, rotation and distortion. One can see, in Figure 5, three more examples of images and the corresponding set of keypoints drawn on top of them. The resolutions are 160x240, 80x120, 40x60 for Figure 5 (a), (b), and (c), respectively. Even though the largest of these three pictures is only half the size of the blurry image, SIFT identified ~4.6 times more keypoints[3].

MØT
Moby Dick
Helt siden starten i 1988 har Hvalsafari, Andenes fått besøk av et utall journalister og TV-team. The Whale & Dolphin Conservation Society (WDCS), som har spesialisert seg på å vurdere tilbud om hvalsafari, gir oss denne attesten:
«Hvalsafari AS på Andenes – verdens største og beste operatør av arktisk hvalsafari».
Produktet
Kvalifiserte guider
Sesong
Avganger
NO. 1 IN WHALEWATCHING
Whalesafari Andenes
THE ORIGINAL AND LEADING SINCE 1988

Figure 4: Original image that figure 3 is part of. Figure 3 can be found in the bottom right section of this image.

[3] *1100 / 240 _ 4.6*

SIFT identified substantially more keypoints for the three latter images because they are in focus, and thus have greater contrast. When these conditions are present in an image, SIFT will be able to identify similar amounts of keypoints.

We will in the next chapter present many more results and findings regarding SIFT, its keypoint descriptors, and matching, but before that we need to have a closer look at SIFT's four computational phases.

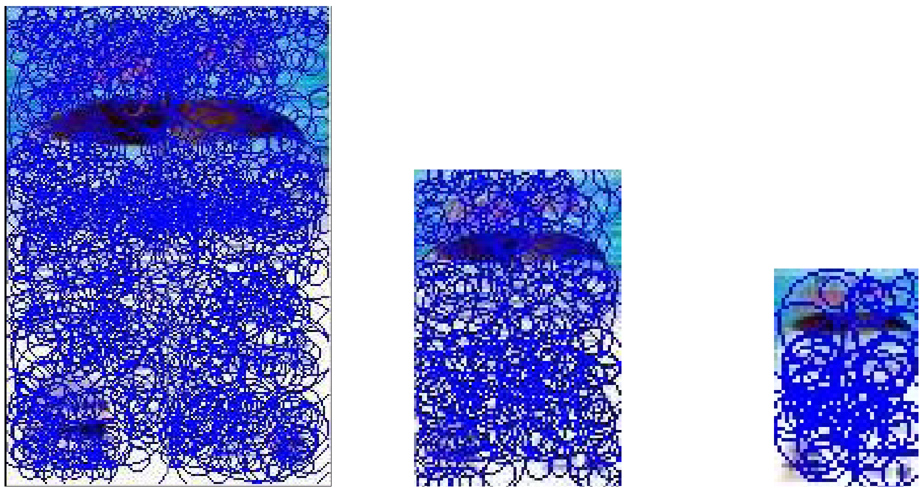

(a) 1000 kp (b) 258 kp (c) 68 kp

Figure 5: Example of the difference in keypoints detected by SIFT when an image is scaled. Keypoints have been abbreviated kp.

3.3 Four computational phases

Since we want to determine whether SIFT is a suitable algorithm to use as part of a CIBR service (see section 1.3), we should know the basics of how the algorithm works. A better understanding of the algorithm should lead us towards a better understanding of the behaviour of the algorithm. Furthermore, it will make us able to analyze the results in the next section in greater detail. The following discussion will be based on, and sometimes directly copied from Lowe's paper [2].

3.3.1 Phase 1: Scale-space Extrema Detection

The first phase of the computation seeks to identify potential interest points. It searches over all scales and image locations. The computation is accomplished by using a difference-of-Gaussian (*DoG*) function. The resulting interest points are invariant to scale and rotation, meaning that they are persistent across image scales and rotation.

3.3.2 Phase 2: Keypoint localization

For all interest points found in phase 1, a detailed model is created to determine location and scale. Keypoints are selected based on their stability. A stable keypoint is thus a keypoint resistant to image distortion.

3.3.3 Phase 3: Orientation assignment

For each of the keypoints identified in phase 2, SIFT computes the direction of gradients around. One or more orientations are assigned to each keypoint based on local image gradient directions.

3.3.4 Phase 4: Keypoint descriptor

The local image gradients are measured in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

There are nine parameters one can assign to adjust what criteria SIFT uses on its four-step way to identify keypoints. However, the author has only evaluated the performance of a SIFT implementation, and not looked into exactly how these parameters affect keypoint generation. Still, the parameters used are the ones Lowe set forth as optimal, and his recommendations are based on empirical studies.

It is now time to present image matching, and some results obtained when testing for object recognition in images.

3.4 Image matching

When one has generated sets of keypoint descriptors for two or more images, one can begin matching images. One can at the outset think of three possible image matches and they are as follows:

1. A match where the whole of one image matches the whole of another image.
2. Part of one image matching the whole of another image.
3. Part of one image matching part of another image.

The different matches will have different characteristics. That is, when we have a match given in case 1, we can expect a fairly large percentage of all keypoints matching.

In case 2, we would expect a large percentage match in the image with a whole match, and a small percentage match in the image partly matching.

In the last case, case 3, we would expect a fairly low percentage of keypoints matching in both images. We will now present some test results, before we examine the matching algorithm used.

The author wishes to stress that the purpose of the matching and keypoint tests presented in this paper is to establish how well the SIFT algorithm can recognize the contents of pictures taken with a mobile phone. The content of an image is expected to be an advertisement from a tourist magazine, and we wish to find out whether SIFT can reliably detect what advertisement one has taken a picture of. The test set-up will be explained in the next chapter, but we anticipate it here since we have already begun discussing matching. Furthermore, it should serve as a reminder for the reason why we have done some special image tests, also presented in the next chapter. Let us now get back to the keypoint matching.

Lines between matching keypoints have been drawn on top of the images for ease of understanding. The matches are not always correct, so some lines will point to non-matching locations. We shall call a correct match a hit, and a false match a miss.

In figure 6, we see a matching of two images that have a lot of similarity. As expected, a large number of keypoints were identified as matching. The image on top has been taken with a mobile phone camera, and the image below has been scanned.

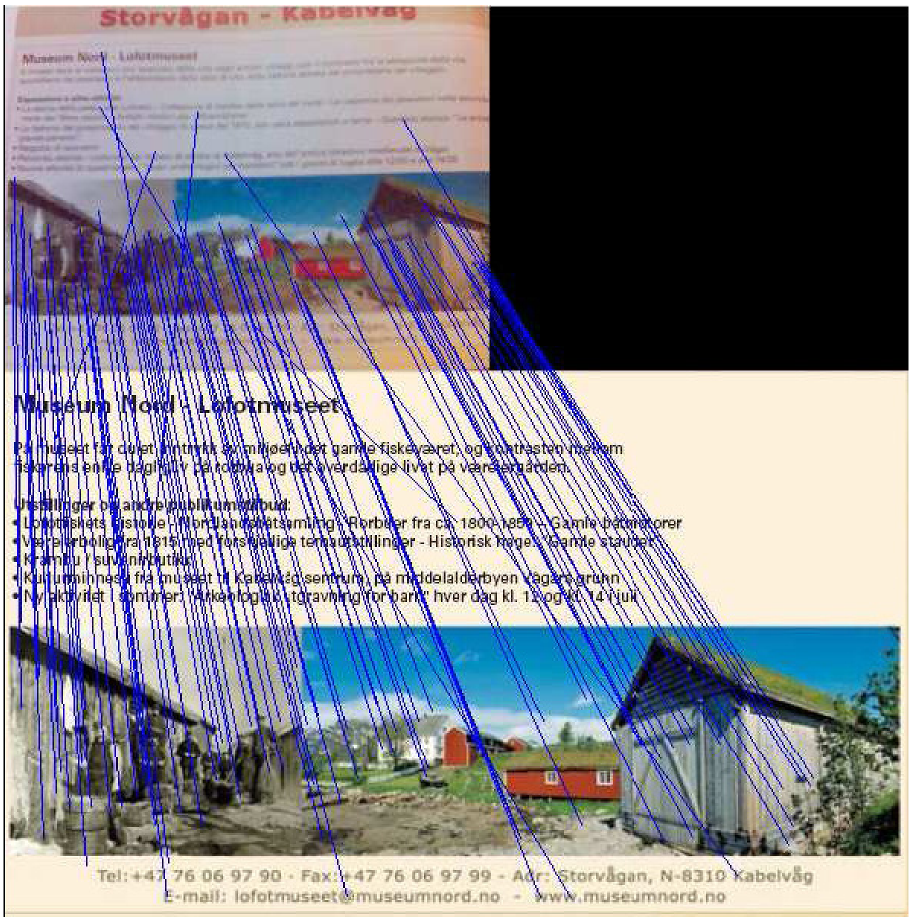


Figure 6: Two images with a large match percent. A total of 100 keypoints were identified as matching.

We can see an example of case 2 in figure 7. Again, the image on the left is taken with a mobile camera, and the image on the right has been extracted from PDF. Note that, although there are many hits, there is also one obvious miss. That is, the almost vertical line going from the grass in the picture on the left, and to the hills in the image on the right is falsely identified as a match.

To determine whether two images are similar, or contain a similar object, David Lowe, the inventor SIFT, suggests that one performs a nearest neighbour search (*NNS*) to identify similar key point descriptors. A key point descriptor is considered a neighbour to another if they have many characteristics in common. Remember that a keypoint descriptor is a 128-dimensional vector. Thus, if two such vectors are similar, they are likely to be a description of two similar objects.

A common way to perform an NNS is to first create a kd-Tree, and then traverse it. We will not go into the details on how this tree is constructed, but it is worth mentioning that the running time of such a search is O(n log n). This is a lot better than the running time of a linear search, which is $O(n^2)$.

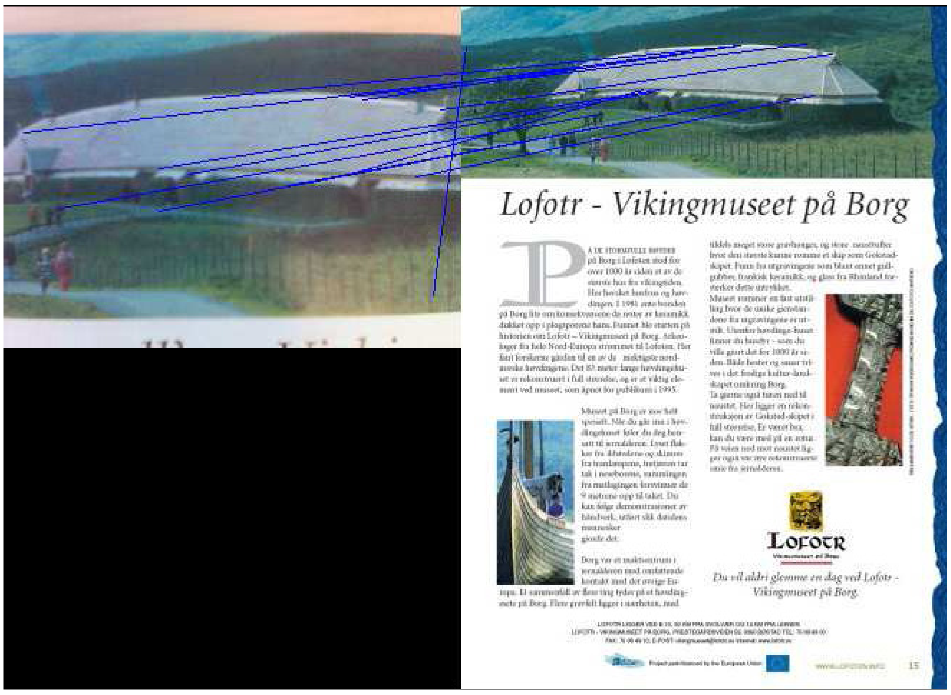

Lofotr - Vikingmuseet på Borg
Lofotr
Du vil aldri glemme en dag ved Lofotr - Vikingmuseet på Borg.
15

Figure 7: Two images with a large match percent. A total of 15 keypoints were identified as matching.

After all neighbouring keypoint descriptors are identified, Lowe advocate to perform a Hough transform on the set of matching keypoints. The purpose of the Hough-transform is to filter out false matches. However, we have not used a Hough-transform in our tests, and the reason being that we have run tests with images of very poor quality. In our case, only a few keypoint matches can be sufficient for identifying two images as a match, but this would be impossible had the few keypoints been removed by a Hough-transform.

We shall now move on to the next chapter, where we will present test results such as running time of SIFT keypoint generation, the time it takes to match images, special purpose tests, and more.

4 Results and Discussion

The SIFT algorithm has gone through extensive testing, and here we will present all results and findings. We will discuss the results on the fly, meaning we do not separate the results and the discussion of the results into two different sections. And the author believes the reader will get a better understanding of the SIFT algorithm by presenting the results and discussing them at once, to point out what is well, and what is not.

Another reason to combine the results and the discussion is the Quality of Match, which will not be introduced before halfway through the testing. The reader needs to be aware of some issues we encountered when testing before being introduced to the QoM.

The test results can be divided roughly into three categories.

The first category consists of image tests done for the reason of curiosity. For example, we have asked questions like "How does SIFT handle an all blue image?" and "How does blurring an image affect the number of keypoints generated?"

The second category has to do with timing and keypoint generation; here we try to find answers to questions like "How long does it take to generate keypoints?" and "How many more, or less, keypoints will be generated when we reduce the size of an image?"

In the third category we put all image matching done on transformed images. An image transformation can be, for example, to remove colour, scale the image up or down, and so on.

We will present the categories in the chronological order they are stated. That is, we begin with the first category, then proceed with the second, and finish off with the third.

4.1 Special purpose tests

We will now present some keypoint generation tests performed mainly to see what SIFT would do with images with very peculiar characteristics.

One test we have done is of pictures with one or two colours only. The images can be seen in Figure 8. Figure 8 (a) is an image filled with solid blue. Figure 8 (b) is the same image as Figure 8(a), except for that vertical red bars have been added.

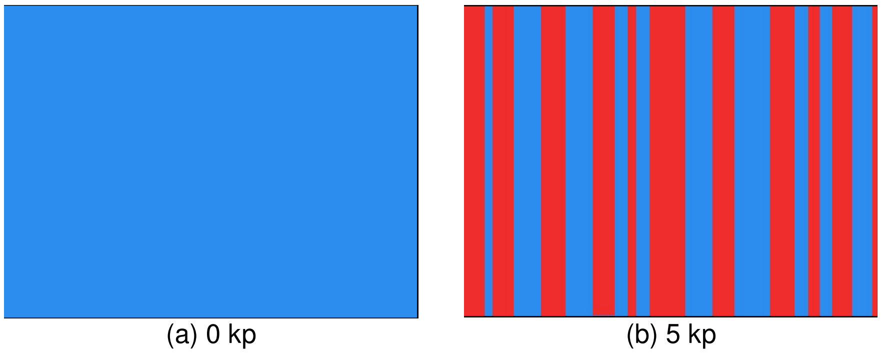
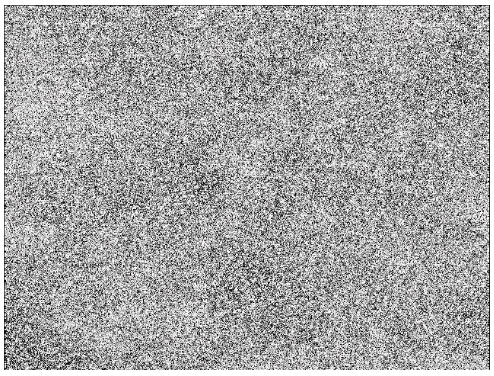


Figure 8: Two special purpose images. We wanted to find out what the output was when SIFT was presented with images with almost no variation.

The purpose of this test was to see how many keypoints SIFT would generate when presented with images that contain very little variation. The all blue image yielded no keypoints at all. This is just as expected, since SIFT detects areas with high contrast and low repeatability. An image consisting entirely of one single colour should therefore generate no keypoints.

As with the first image, one could expect that the second picture would not have any keypoints either. And had the author paid more attention when drawing the image, this might have been so. However, SIFT detected five keypoints. These five keypoints are located at the very bottom of the third red stripe from the left. The reason for these five keypoints is that the third red stripe does not go all the way to the bottom. There are one or two rows of blue pixels at the very bottom. Since blue stands in great contrast to red, and this area was like no other in the picture, SIFT was able to detect keypoints.

Now, let us see how SIFT handles images consisting almost entirely of areas of extreme contrast. The images can be seen in Figure 9.


(a) 5526 kp

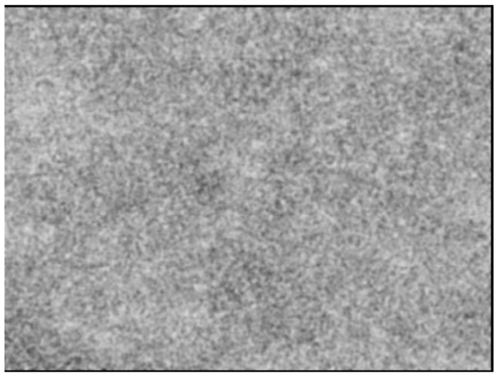

(b) 9891 kp

Figure 9: Another pair of special purpose images. We wanted to find out what the output was when SIFT was presented with images consisting entirely of areas with huge variation.

SIFT found 5526 keypoints in Figure 9(a), and, surprisingly, 9891 keypoints in Figure 9(b). The huge amount of keypoints is not what is surprising; that was rather expected. What is surprising is the effect of blurring an image that has areas with immense contrast. When blurring, one could expect less keypoints to be generated. However, in this case, the number of keypoints is doubled.

One possible explanation for this abnormal behaviour of SIFT lies within the characteristics of the images. It is likely that Figure 9(b) has more variety than Figure 9(a). That is, the blurred image has more unique areas with high contrast, than the original image.

Because of the previous results, we wanted to test the impact of change in contrast and variation had on keypoint generation. Instead of blurring an image with sharp contrast, we wanted to try sharpening a blurry image. Further, we wanted to see if we could find any connection between the two images. The images can be seen in Figure 10.

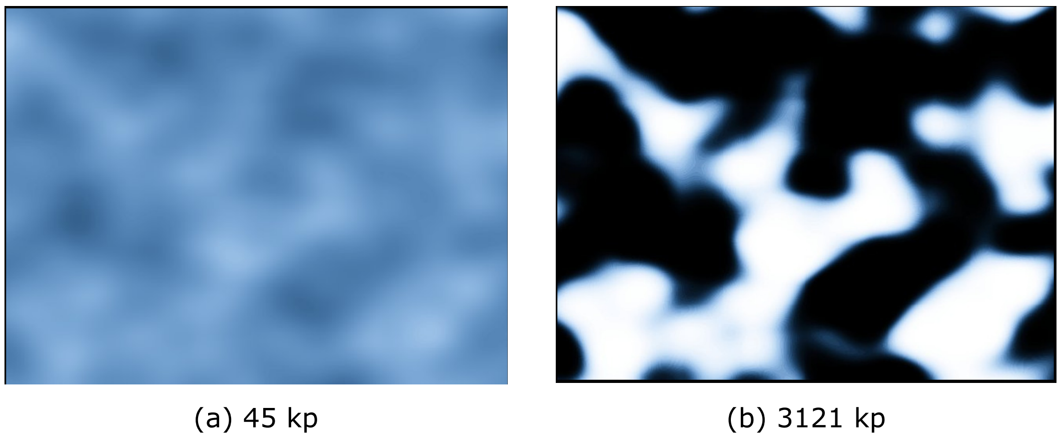


Figure 10: The images used to find out what effect increasing the contrast in an image had on keypoint generation.

As expected of SIFT, the algorithm identifies fairly few keypoints for Figure 10(a). Notice that the image does contain some descent amount of variation, which speaks for a higher amount of keypoints. However, because of the very low contrast between the pixels, SIFT still detects very few keypoints.

Figure 10 illustrates something very important about the SIFT algorithm. Whereas we are able to see structure in a very blurry image, SIFT cannot. If we see a blurry picture of some people in front of a house, we can make a fairly accurate guess of what is in the image. However, if the image is sufficiently blurry, SIFT would be able to locate close to no keypoints. Thus, we can conclude that a prerequisite for SIFT to perform well is sharp and clear images.

Figure 10(b) has 70 times more keypoints than Figure 10(b). Although this is a very, very large increase in number of keypoints, it is not an anomaly. Since the original image had very few keypoints, we would expect the number of additional keypoints to be generated, when contrast was increased, to be very high.

Let us now see where the keypoints generated are located. This is of interest to us since, if an increase in contrast would increase the precision of SIFT, this finding could be used to increase the search precision of SIFT when faced with blurry images taken with a mobile phone. Figure 11 shows the images with their respective keypoints drawn on top of the images, and as before, represented by blue circles.

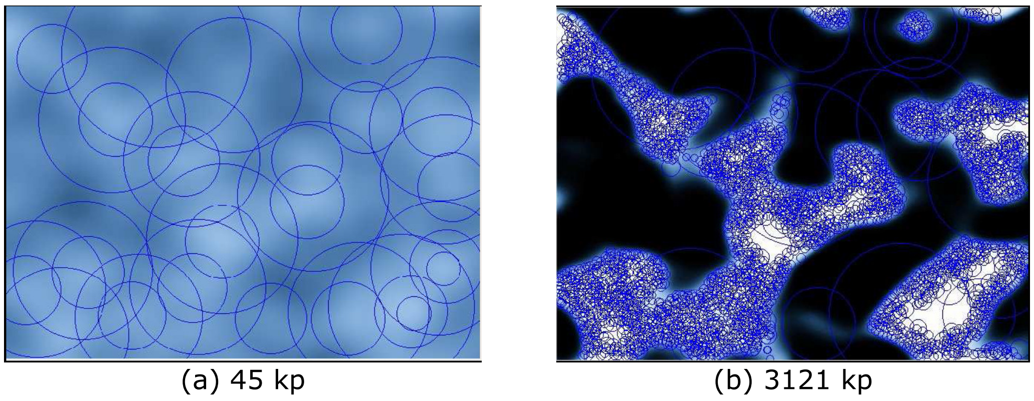


Figure 11:Keypoints, illustrated by circles, have been drawn on top of the images to show their location and scale.

As can be readily seen, the keypoints identified in the contrasted image, on the right, are in the exact same location, or area, as the keypoints detected in the original image. That is, the areas dense with keypoints in Figure 11(b) are areas covered by one circle or another in Figure 11(a). What one should note about this is that the slightly lighter areas in the original image have given rise to huge amounts of keypoints when the contrast has been increased.

To illustrate how closely the keypoints of the contrasted image follow the keypoints in the original image, we draw, in the low contrasted image, red lines from the center of each of the larger circles and to the other overlapping large circles. Next, put the high contrasted image on top of the low contrasted image (with the red lines). The result can be seen in Figure 12.

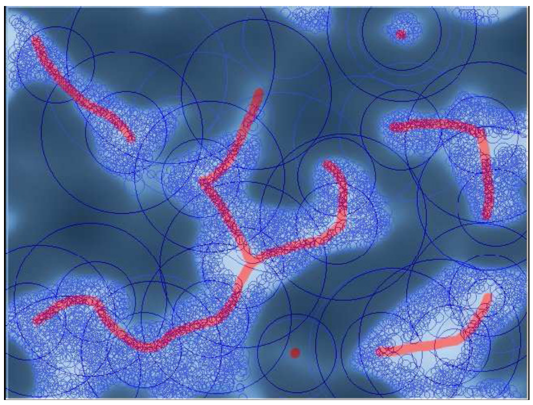


Figure 12: The low contrasted and high contrasted images have been combined to one image.

There has been performed tests involving change in contrast on a blurry image, to see if this could increase the accuracy of SIFT. However, when the contrast was changed, the keypoint descriptors also changed. Therefore, the keypoints in the higher contrasted, mobile phone image could not be matched with keypoints from a real image of an advertisement. Still, this needs to be tested more before one can draw any conclusion on whether adjusting contrast can be useful for blurry images or not.

And with that we conclude the presentation of the first category of image tests. The next category we will present is timing results.

4.2 Running time

We have measured the running time of keypoint generation and of keypoint comparison. We have tested both under various circumstances, for example by varying the image size.

4.2.1 Keypoint generation

Let us begin by having a look at keypoint generation time. In Figure 13 we have plotted a graph illustrating the average keypoint generation time of keypoint generation. The graph data was gathered from generating keypoints from a total of 45 images with varying resolution. The resolution of the images varied from 640x480 to 40x30.

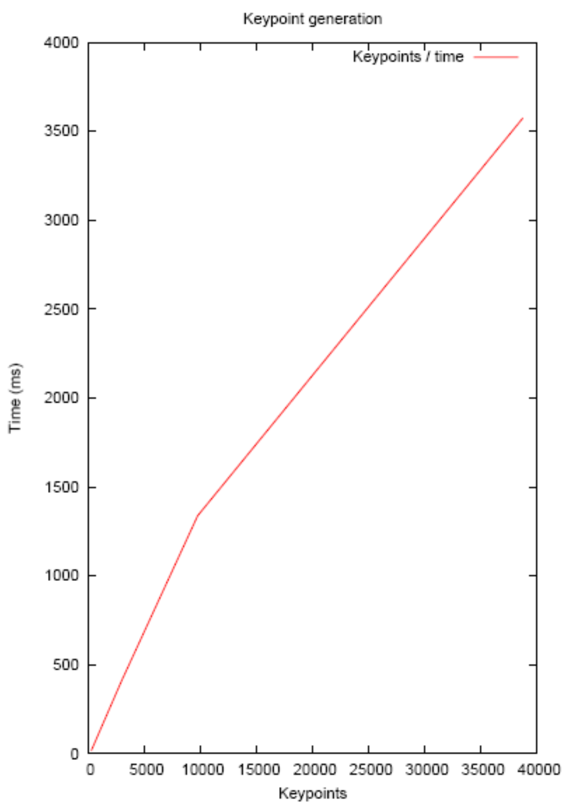


Figure 13: Keypoint generation time

The keypoint generation time is almost a linear function. Had we performed even more tests, it is likely that we could derive a linear function for the keypoint generation.

The running time of keypoint generation is only dependent on the number of keypoints detected, and *not* the resolution of an image. This is illustrated in figure 14. The large images with long running time are images for which SIFT detected many keypoints. Notice that there is a line below the 500 mark on the y-axis which has almost no change in running time, even though the image is scaled down to 1/4 the size. This particular image is the blurry image presented in the previous chapter. It is labeled Figure 3, and can be found on page 5.

Why the running time of keypoint generation is not affected by the image size is because of the four phases of SIFT. During the first phase, all points in an image have to pass some initial test, which is very cheap in terms of complexity to perform. Thus, in a large, but blurry image, one could expect that very few points made it past the first phase of the algorithm, and hence the low running time. The opposite is true for a very small, but clear and contrasted image. It would have a high running time, since expensive calculations had to be performed for many points in the image.

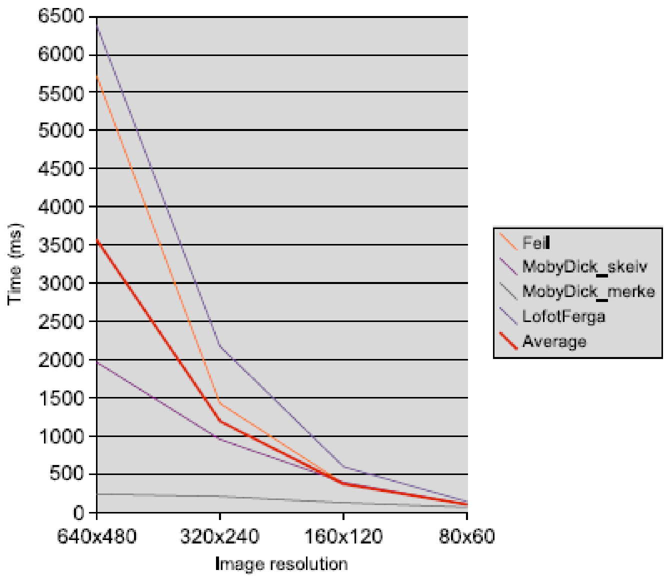


Figure 14: Keypoint generation time for images with an initial large resolution

To further illustrate this, we have included a table which shows the average number of keypoints generated per millisecond (Figure 15). Though the graph is not a horizontal line, which would indicate constant keypoint generation time, the keypoints per millisecond ratio varies very little. That is, it goes from a low value of 0.07 keypoints/millisecond to a high value of 0.14 keypoints/millisecond. The variation is caused by the small amount of tests we have performed. SIFT is likely able to identify about 1.1 keypoints/millisecond, on average.

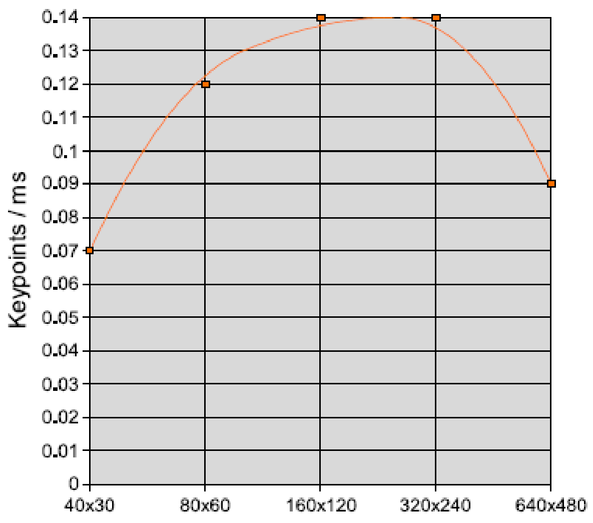


Figure 15: Average keypoint generation per millisecond

We will now leave the topic of keypoint generation timing, and will move on to keypoint matching timing.

4.2.2 Keypoint matching

We have tested for the time it takes to compare two sets of keypoint descriptors. As mentioned earlier, this is accomplished by a NNS in a *k*d-Tree.

However, though the matching utilizes a NNS, the criterion for a match is not to be a nearest neighbour; then all nodes would have a match. Note, as stated before, the comparison is done from the source image, to the compared image. Thus, in the following outline, to "select a node" means to select a node from the keypoints of the source image. The procedure is as follows:

1. Select a node from the set of all nodes not yet selected.
2. Mark the node as selected.
3. Locate the two nearest neighbours of the selected node.
4. If the distance between the two neighbours are less than or equal to a given distance, we have a match. Mark the keypoints as match.
5. Perform step 1-4 for all nodes in the source image.

The key step of the matching algorithm is step 4. It is here it is decided whether the source node (or keypoint) has a match in the compared set of keypoints, or not. Figure 16 illustrates this step of the matching procedure.

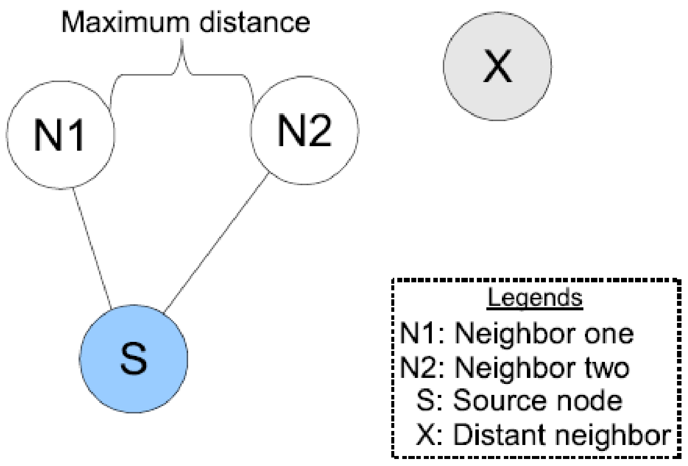


Figure 16: The image matching algorithm verifies the distance between the two closest neighbours of a source node. If the distance is within a given boundary, the source node is marked as a match

We shall now present some results on the running time of keypoint matching. In Figure 17, we see the average number of keypoints compared per millisecond. In this diagram, we can see what impact the ratio of keypoints between the source image and the compared image has on the matching procedure.

For instance, the first column represents a comparison between an image with a small amount of keypoints with other images with a small amount of keypoints[4]. The S stands for small, and the L for large. Thus, S:L means the source image has few keypoints, while the compared images has many, and L:S means the source image has a large set of keypoints, while the compared images has a small set.

[4]*A set of keypoints ≤ 500 is considered small*

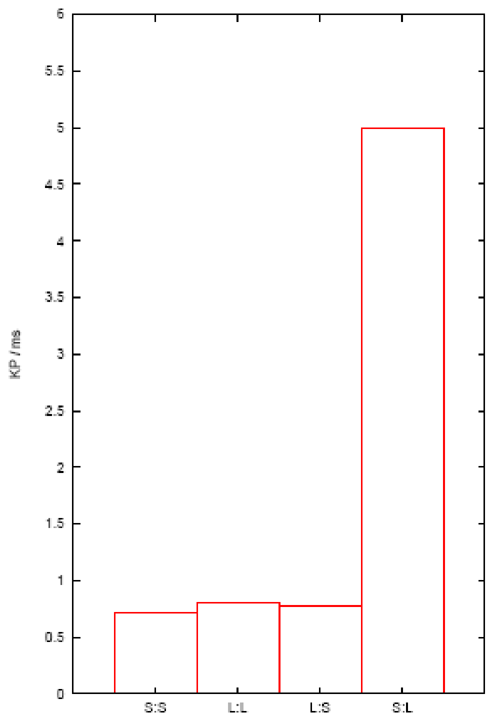


Figure 17: Keypoint matching results, taking the ratio of keypoints between the source and compared images into account.

The number of keypoints processed per millisecond is almost constant for all cases, with one exception. The exception is the fourth column, and it represents the case when the source image has few keypoints, and the compared images has many keypoints. The author can only guess at the reasons for this. It seems likely that the answer lies within the structure of the *k*d-Tree, and how it is traversed by the NNS algorithm.

Remember that the NNS search is only an approximate search, which means it offers no guarantee of finding the closest neighbour, but only "a pretty close one". If the NNS search is able to identify a match between a source node and two neighbours without examining all nodes in the compared image, it would be able to terminate fast. And this is probably what causes the many keypoints per millisecond ratio. Namely, the fact that it runs 5 times faster than the other three tests, implies that 80%, or 4 out of 5 keypoints, is never examined. Again, the reason being that a satisfactory match has already been detected.

Before we present matching results, let us have a look at the running time of the NNS when searching through a collection of images (see Table 1). Because of the usually high number of keypoints that need to be compared, the total running time of an image search is usually very high. In particular, notice the very high running time when comparing the image ”Feil‟. The higher resolution of 640x480 pixels yields exponentially higher running time.

The total running time of comparing 35 images up against a collection of 58 images was 2 hours, 12 minutes and 22 seconds. That is an average of 3 minute and 27 seconds per comparison. Such high running time is unacceptable for a real-time image matching system. However, notice that the larger images have a much higher running time than the smaller ones, as expected from previous results presented in this section. We shall later see how well the matching performs when we scale down the source images in order to decrease the running time. The trade-off when scaling down images is that we get less key points. In other words, we trade quantity of keypoints for speed.

Table 1: Time to match a single image with a collection of 58 images.

Source Image	Image Dimensions	Time (minutes:seconds)
LofotFerga	320x240	07:54
Nusjford	384x288	09:58
Svinøya	96x72	01:58
Feil	640x480	18:42

It is now time to leave the second category of timing results, and move on to the next and last category. We shall now examine many search results and establish how accurately SIFT is able to identify two matching images.

4.3 SIFT matching accuracy

We shall test the accuracy of SIFT under various circumstances. We will begin by, for all source images, search through a collection of 9 images for matches. We will, for now, let the compared image with the most keypoint matches be the best match. Thus, if one compared image has 4 matches, and another 7, the latter would be the image best matching the source image.

All source images are taken with a mobile camera, and are of an advertisement, or part of one. Furthermore, all images in the collection, that a source image is compared to, are scanned images of an actual advertisement. Thus, the source images are usually of poor to medium quality, while the compared images offer high quality.

Both the source images and the compared images are highly diversified in order to best model a real test environment. We shall later present results from 56 source images, where each one has been compared with a collection of 122 scanned images. But, for now, we shall stick to a single source image at a time, and the results obtained when comparing it with a collection of 9 scanned images.

To begin with, we can refer back to Figure 6 and Figure 7on page 10 and 11, respectively. They are both results from searching through the collection of 9 images, and are the best matches. The similarity between the source image and the one identified as best matching by SIFT is very high, so let us now look at a test where the amount of similarity is not so great.

In Table 2 is listed all matches for the image "Mobil_MobyDick_merke.jpg". In Figure 18 we have illustrated this best match, for the reader to verify that it is correct.

So, even though the source image is blurry, and is only a small part of the advertisement, SIFT managed to detect many matching keypoints. Note, however, that there is a few keypoints identified as matches, but are not correct.

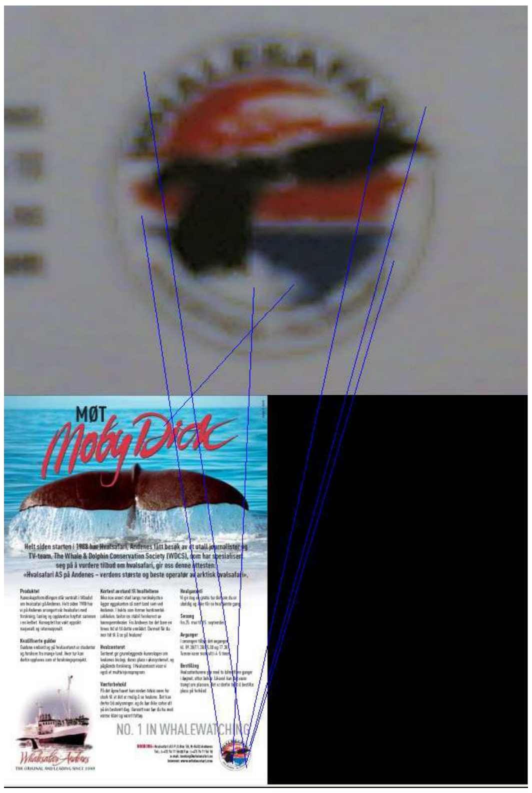


Figure 18: Best match illustrated for Mobil_MobyDick_merke.jpg. SIFT has correctly identified the source image to be part of the MobyDick advertisement.

Table 2: Image matches for Mobil_MobyDick_merke.jpg

Filename	Number of matches
Lofotmuseet.jpg	0
Lofotdykk.jpg	1
Storvaagan.jpg	1
MobyDick.jpg	21
Lofotferga.jpg	0
Rorbuhotell.jpg	0
Borg.jpg	0
LofotenDesign.jpg	0

Next, we have tested how well SIFT is able to match a rotated picture.

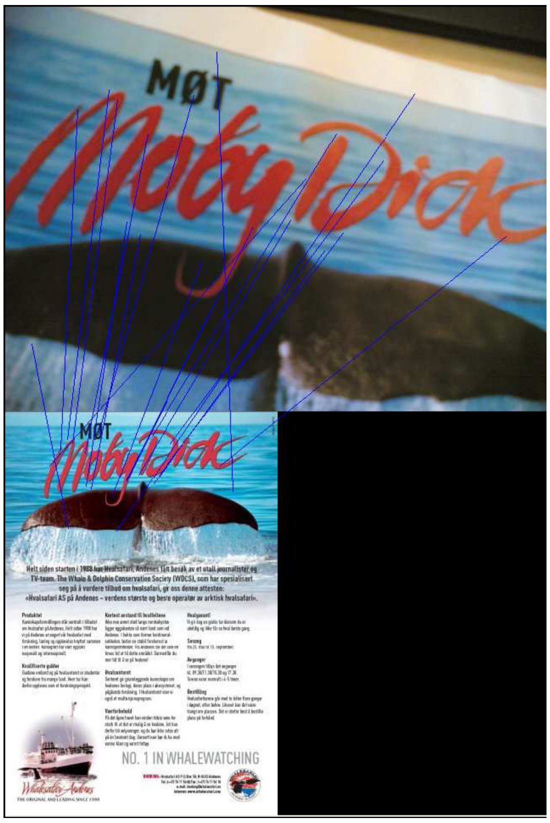


Figure 19: Best match illustrated for Mobil_MobyDick_skevt.jpg. Again, SIFT has correctly identified the source image to be part of the MobyDick advertisement.

Table 3: Image matches for Mobil_MobyDick_skevt.jpg

Filename	Number of matches
Lofotmuseet.jpg	2
Lofotdykk.jpg	4
Storvaagan.jpg	3
MobyDick.jpg	21
Lofotferga.jpg	3
Rorbuhotell.jpg	2
Borg.jpg	3
LofotenDesign.jpg	0

Again, similarities between keypoint descriptors produced by SIFT was used to correctly locate what advertisement the source image was taken of. Let us now see if we can match images when we remove the colour from the source image, the compared image, or both.

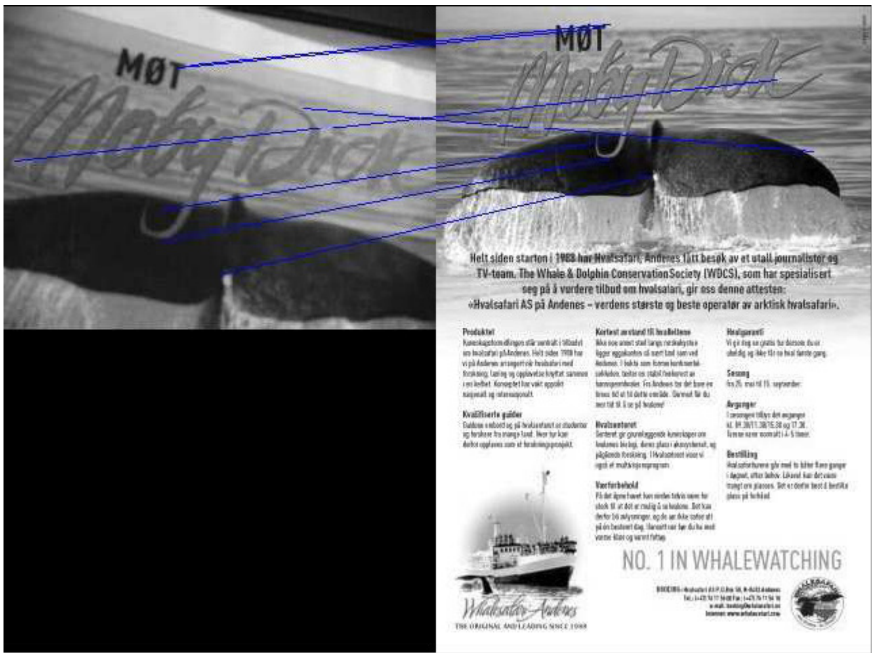


Figure 20: Although we removed the colour from both images, the match found by the NNS search was correct.

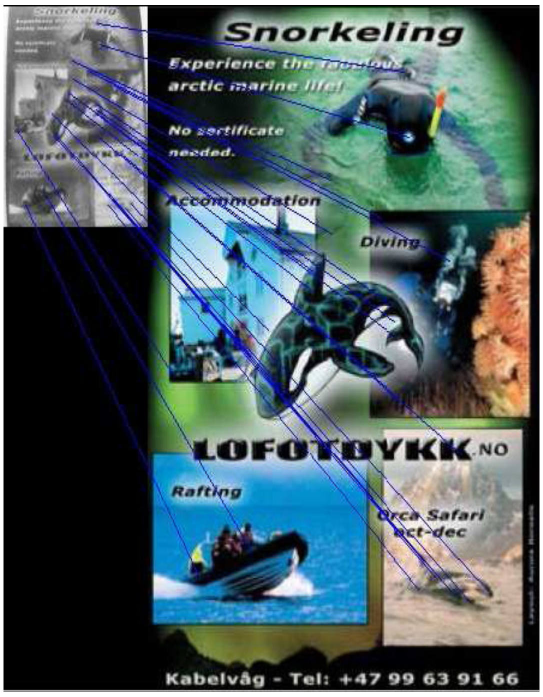


Figure 21: We have removed the colour from the search image, but the keypoint descriptors generated by SIFT are not affected. The match is correct.

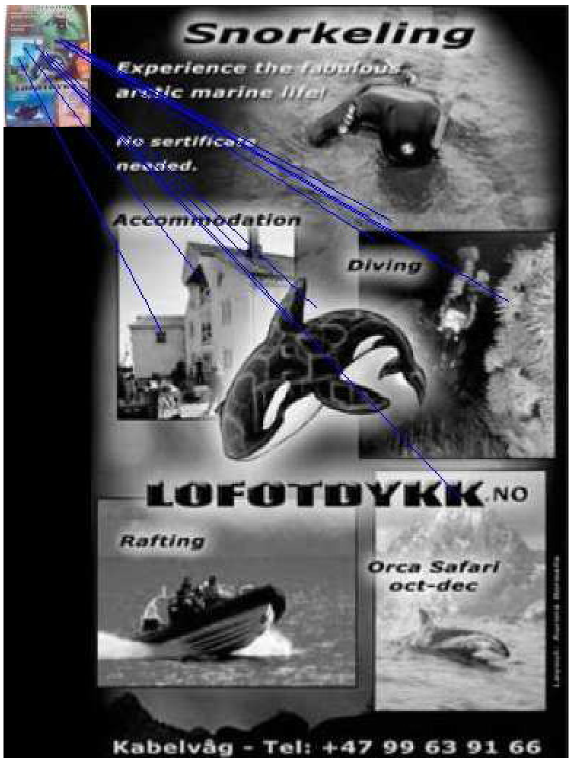


Figure 22: We removed the colour from the compared images. NNS still managed to locate the correct image.

Thus far, everything has gone well. The previous all have a high number of keypoints. This is of interest to us, since, as we saw earlier, if we reduce the amount of keypoints in the source image, detecting matches is very fast. But, first, let us test for the accuracy of matches when we compare an image with a large set of keypoints with an image with a small set.

The source image has a large amount of keypoints, as in the previous tests. The collection of compared images has been downscaled by a factor of 8, which ensures a set of very few keypoints for each compared image.

Table 4: Image matches for MobyDick_rett.jpg, having 2404 keypoints. Only the four best results are shown here.

Filename	Number of matches	Number of keypoints
Borg-42x60.jpg	36	53
Lofotferga-40x60.jpg	12	62
LofotenDesignAS-33x51.jpg	10	41
MobyDick-40x60.jpg	9	68

As can be seen in Table 4, the matching failed badly. The correct image, ”MobyDick-40x60.jpg‟, is only the fourth best match. Furthermore, it has only 9 matches, whereas the best matching image has a total of 36 matching keypoints. Many more tests where the source image has many more keypoints than the compared images have been performed, and the result is *always* the same.

What we can conclude from this is that, whenever we want to find a best match for an image, we absolutely must take the ratio between the number of keypoints in the source and the compared image into account. If the number of keypoints in the source greatly exceeds the number of keypoints in the compared image, the match is very, very unreliable. That is, the correctness of the match goes *down* as the ratio goes up.

Now, to handle this unfortunate behaviour of NNS, a normalized rating will now be proposed. It is called a *Quality of Match*.

4.4 Quality of Match

First, we shall make some observations about the results preceding this section. Second, we will set up an initial formula based on the observations, and modify it using basic algebraic manipulations. Lastly, we will feed some of the preceding results to the formula, and see how well its output describes the quality of the match.

4.4.1 Observations

We start with a few variable assignments.

Let K_s be the number of keypoints in the source image, K_c be the number of keypoints in the compared image, and K_m be the number of matching keypoints.

Now, one might be tempted to argue that the percentage $QoM = \frac{K_m}{K_s} \times 100$ is a good estimate for how well two images match, or the alternative $QoM = \frac{K_m}{K_c} \times 100$. In some cases this is correct. Namely, when the number of keypoints in both images is about the same, we could use either of the two formulas to compute a percentage of matching keypoints.

However, we saw earlier that when the number of keypoints in a source image greatly exceeds the number of keypoints in the compared image, the matching keypoints are very unreliable. And when the number of keypoints in the source image is much lower than the number of keypoints in the compared image, the matching keypoints are very, very reliable.

From this we note that

1. $K_s \gg K_c$ = very unreliable matching
2. $K_s \ll K_c$ = very reliable matching

Note that we are most interested in the number of matching keypoints from the source, to the compared image. Thus the second formula, $QoM = \frac{K_m}{K_c} \times 100$, will be not be taken into further consideration.

Our formula must take the ratio between the number of keypoints in the source image and the number of keypoints in the compared image into account, lest it be very inaccurate in many cases.

Let the ratio be represented with $R = \frac{K_s}{K_c}$. Observe that if R > 1, then there are more keypoints in the source image than in the compared image. In order to reflect the unreliable matches, a ratio R > 1 should lower the importance we give our keypoint matches.

The opposite is also true. A ratio R < 1 should give more weight to a match, since the likeliness of a match being a proper match grows.

Let us now use the ratio to derive a formula for calculating the QoM.

4.4.2 Quality of Match formula

As noted in the previous section, a low ratio should increase the QoM, and a high ratio should decrease the QoM. Furthermore, a ratio of about 1 should not affect the QoM. Thus, let $QoM = \frac{K_m}{K_s} \times 100 \times \frac{1}{R}$.

Note that if R < 1, the importance of a match will increase. If R > 1 the importance of a match will decrease. Since $R = \frac{K_s}{K_c}$, we can rewrite the QoM formula to

$$QoM = \frac{K_m}{K_s} \times 100 \times \frac{1}{\frac{K_s}{K_c}} = \frac{K_m \times K_c}{K_s^2} \times 100 \tag{1}$$

This formula now takes into account the ratio as well as the percentage of matches in the source image.

4.4.3 Normalized rating

We obtain a normalized rating for the search results with the following algorithm.

1. Locate the best QoM
2. For every match, set $Rating = \frac{QoM}{BestQoM} \times MatchPercent \times 100$

Thus, we ensure that the rating is a number between 0 and 100, 0 being a complete miss and 100 being a perfect match.

4.4.4 Testing the QoM formula

We will now present the most comprehensive search result in this paper. We have used 56 different images as source images, and compared each one with a collection of 122 images. The 122 images contain images of varying resolution. There are 61 unique images, and each image is represented in 2 different scales (original size, and 1/4 size).

Before commenting on the results, it should be in place to list the meaning of each column in the result tables. They are as follows:

Image The filenames have been exchanged for numbers. They only serve to distinguish between the images in the search result.

Matches The number of keypoints in the source image matching keypoints in the compared image.

Match % The percentage of matching keypoints in the source. This is the percentage we have in all previous tests rate a search with, meaning that the compared image with the highest number of keypoint matches was deemed the best match.

QoM The Quality of Match, calculated using the formula recently described.

Rating The rating of the image, based on the QoM value and the match percent. This was also recently described.

Correct? A "Yes" in this column means that the image identified is a correct match. That is, the source image was correctly matched with an advertisement.

Let us examine the results displayed in Table 5. This result is of particular interest because of the third row. It shows an image match of 26.9%, however, it is only given a rating of 1.4. The reason for this is, as should now be clear, that the ratio of keypoints between the source image and the image at row number 3 is very high.

Thus, since the source image has many, many more keypoints than the compared image, the likelihood of making a false match is high. This is taken into account when the rating is calculated, and the image ends up as only a third best match.

Table 5

Image	Matches	Match %	QoM	Rating	Correct?
1	34	11.89	216.1	11.9	Yes
2	25	8.74	77.3	3.1	Yes
3	**77**	**26.92**	**13.5**	**1.4**	**No**
4	11	3.85	43.1	0.8	No

Another result is presented in Table 6. The scenario is much the same as with the results shown in Table 5, and perhaps even more revealing. The second best match has a much lower rating than the one ranked as first, even though it has over twice as many keypoint matches. Note also the low rating of the third image compared to the match percentage.

Table 6

Image	Matches	Match %	QoM	Rating	Correct?
1	189	8.37	15.1	8.37	Yes
2	**418**	**18.51**	**1.2**	**1.4**	**No**
3	**387**	**17.14**	**0.3**	**0.37**	**No**

Let us now compare search results with and without QoM. The purpose of this test is to see how well the image matching process performs, on average, by only comparing keypoints, as well as how much better it performs when one takes the QoM into account. This test also serves to disclose whether the SIFT algorithm is reliable enough to be included in a project such as M2S.

We have compared 54 source images with a collection of 122 images (see Table 7). That is, each of the 54 source images has been compared to each of the 122 images in the collection. For each source image, a best match has been recorded. The correctness percentage represents how often the best matching image was correctly identified as similar to the source image.

Table 7

# of Source Images	Images in Collection	Correctness	QoM?
54	122	52%	No
54	122	92%	Yes

Table 7 needs some explanation. One reason that the matching performed almost twice as good when one takes QoM into consideration is that a certain image in the collection gave consistently high matching percentage, regardless of what image it was compared to. The size of this image was small[5], and that is why the QoM could filter this image out in almost all cases, and thus obtain a much higher correctness percentage. However, without the QoM, the image was cause of many mismatches for the regular NNS.

It is also worth mentioning that "out of the box"-SIFT is not meant for this type of image matching. It is designed to detect similar or equal objects in two high resolution images, and not for comparing downscaled, low resolution images. Furthermore, it was because of this characteristic the QoM was developed in the first place.

In summary, for our special purpose test of comparing low resolution images, a regular NNS search yields below average results with 5 out of 10 images correctly matched. However, when taking the QoM into account, the matching procedure becomes much, much more reliable with 9 out of 10 images correctly matched.

This concludes the SIFT and matching discussion. We will now move on to the conclusion.

[5] *Only 40x30 pixels*

5 Summary and Conclusion

5.1 Summary

The SIFT algorithm has been shown to be a slow, but robust algorithm for image comparison. Through extensive testing, we have discovered under what circumstances SIFT image matching works well, and when it does not.

For our special purpose image matching, namely that of tourists taking pictures of magazine advertisements, we have seen that by calculating a quality of match, and a rating, we can significantly increase the correctness of an image match.

We also discovered the reason for why we needed a quality of match; the NNS search will incorrectly identify many keypoints as matching when the following requirements are met:

1. The compared image is of low resolution (typically around 40x30)
2. The source image is of fairly high resolution (at least 3 times the resolution of the compared image)

We have also seen that SIFT is able to detect similarities between images, even though the images has gone through a transformation. The transformations we tested were:

Saturation We removed colour from the source image, the destination image, and both.

Scale We scaled down the source image as well as the destination image.

Rotation We had pictures taken at a skewed angle.

We have seen that the big drawback of using SIFT is the complexity of matching sets of keypoint descriptors. The time it takes to compare a single image with 50 other images takes, on average, over 3 minutes.

5.2 Evaluation

SIFT has been demonstrated to be very suitable for object detection in images with high resolution. However, SIFT performs poorly when it is faced with images of poor resolution.

With the addition of a quality of match and rating, the matching results become much more reliable, and especially so for images with low resolution.

For a real time system like M2S to use SIFT, one would have to strictly limit the number of images one compares a source image with. One possible way to do this would be to use geospatial data of where the source image is taken as a filter. The geospatial data would enable the system to detect where the user is located, and thus what objects he or she might have taken a picture of. However, this could be hard to accomplish if the source image is of an advertisement, because an advertisement has no a priori connection to the location an image is taken.

5.3 Future Work

More work needs to be done on possible ways to select a small set of pictures for NNS to process. Geospatial data was mentioned in the previous section as one way to limit the number of pictures a source image is compared to, but it has not been tested yet.

More ways to filter out images include utilizing other content based image retrieval (CBIR) algorithms. Most CBIR algorithms are able to detect low-level features like colour and lighting, and they can be very fast. Such algorithms could thus be applied to an image to find a small set of similar images, and probably reduce the number of images processed by SIFT. That is, given an incoming image X, if a fast CBIR algorithm was able to pick out five similar images, SIFT could be used to pick out the best matching image of those five.

The SIFT algorithm should be tested in a distributed environment. A single computer would not have enough processing power to handle incoming image comparison requests in a system like M2S. Thus, how one could effectively distribute the image comparison process of NNS needs work.

5.4 Conclusion

SIFT does what it is designed to do, and it does it well. The most obvious drawback with SIFT is the time it takes to compare two images. The running-time of an NNS search is so large that it effectively renders SIFT useless for a system like M2S. However, with modifications like quality of match and the utilization of other metadata, SIFT could be an extremely robust resource for object detection and image matching.

References

[1] http://caim.uib.no/

[2] Lowe, D G, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, 60, 2, pp. 91-110, 2004.

[3] Schürmann, A, Aarbakke, A S, Egeland, E, Evjemo, B and Akselsen, S. 2006. Let a picture initiate the dialog! A mobile multimedia service for tourists (pdf). Fornebu, Telenor Research & Innovation. Report R&I N 33/2006.

[4] Akselsen, S, Bjørnvold, T A, Egeland, E, Evjemo, B, Grøttum, K J, Hansen, A A, Høseggen, S, Munkvold, B E, Pedersen, P E, Schürmann, A, Steen T, Stikbakke, H, Viken, A and Ytterstad, P. 2006. *MOVE - a summary of project activities and results (2004-2006).* Fornebu, Telenor Research & Innovation. Report R&I R 17/2006.

[5] Anne Staurland Aarbakke. 2007. *M2S and CAIR: Image based information retrieval in mobile environments.* Masteroppgave i informatikk. Mai 2007. Institutt for Informatikk, Det matematisk-naturvitenskapelige fakultet, Universitetet i Tromsø.

[6] http://user.cs.tu-berlin.de/ nowozin/libsift/