EXAMENSARBETE INOM TEKNIK,
GRUNDNIVÅ, 15 HP

STOCKHOLM, *SVERIGE 2016*

A comparison of object detection algorithms using unmanipulated testing images

Comparing SIFT, KAZE, AKAZE and ORB

OSKAR ANDERSSON

STEFFANY REYNA MARQUEZ

KTH
SKOLAN FÖR DATAVETENSKAP OCH KOMMUNIKATION

En jämförelse av algorithmer för objektigenkänning genom användandet av icke-datormanipulerade bilder

En jämförelse av SIFT, KAZE, AKAZE och ORB

Oskar Andersson and Steffany Reyna Marquez

Degree Project in Computer Science
DD143X

Handledare: Mårten Björkman
Examinator: Örjan Ekeberg
CSC, KTH 2016-05-11

ABSTRACT

While the thought of having computers recognize objects in images have been around for a long time it is only in the last 20 years that this has become a reality. One of the first successful recognition algorithms was called SIFT and to this day it is one of the most used. However in recent years new algorithms have been published claiming to outperform SIFT. It is the goal of this report to investigate if SIFT still is the top performer 17 years after its publication or if the newest generation of algorithms are superior.

By creating a new data-set of over 170 test images with categories such as scale, rotation, illumination and general detection a thorough test has been run comparing four algorithms, SIFT, KAZE, AKAZE and ORB. The result of this study contradicts the claims from the creators of KAZE and show that SIFT has higher score on all tests. It also showed that AKAZE is at least as accurate as KAZE while being significantly faster. Another result was that while SIFT, KAZE and AKAZE were relatively evenly matched when comparing single invariances that changed when performing tests that contained multiple variables. When testing detection in cluttered environments SIFT proved vastly superior to the other algorithms. This led to the conclusion that if the goal is the best possible detection in every-day situations SIFT is still the best algorithm.

SAMMANFATTNING

Att förmå en dator att känna igen objekt i en bild har länge varit en dröm bland forskare men under de senaste 20 åren har denna dröm börjat bli verklighet. En av de första algoritmer som lyckades göra detta kallas för SIFT och är än idag en av de mest använda. Under de senaste åren har dock nya algoritmer plublicerats vilka hävdar att de har bättre träffsäkerhet än SIFT. Det är denna rapports mål att undersöka om SIFT fortfarande är den mest träffsäkra algoritmen 17 år efter dess publikation eller om den nya generationen av algoritmer är bättre.

Genom att skapa ett nytt data-set på över 170 test bilder med kategorier som skala, rotation, illumination och generell igenkänning har ett noggrant test utförts som jämför fyra algoritmer, SIFT, KAZE, AKAZE och ORB. Resultatet av denna studie motsäger påståendet från skaparna av KAZE och visar att SIFT får fler träffar på samtliga test. Resultatet visar även att AKAZE är minst lika träffsäker som KAZE och dessutom betydligt snabbare. Utöver detta så visar studien att även om SIFT, KAZE och AKAZE är relativt jämnbördiga när det kommer till att jämföra enskilda invarianser så ändras detta när man genomför test som innehåller flera variabler. När tester gjordes på igenkänning i stökiga miljöer visade sig SIFT vara betydligt mer träffsäker än de andra algoritmerna. Detta ledde till slutsatsen att om bästa möjliga igenkänning söks i vardagliga förhållanden så är SIFT fortfarande den bästa algoritmen.

CONTENTS

ACRONYMS

SIFT Scale-invariant Feature Transform

DoG Difference of Gaussian

AOS Additive Operator Splitting

PDE Partial Differential Equation

DoH Determinant of the Hessian

AKAZE Accelerated KAZE

ORB Orientated BRIEF Rotated FAST

BRIEF Binary Robust Independent Feautures

FAST Features from Accelerated Segment Test

SURF Speeded Up Robust Features

FED Fast Explicit Diffusion

CV Computer Vision

IC Intensity Centroid

LDB Local Difference Binary

1

INTRODUCTION

1.1 COMPUTER VISION

While a three year old child can easily recognize and detect objects in a picture our smartest computers still struggle. This is because a computer sees a picture only as a matrix of numbers.

Computer vision deals with processing and analyzing visual data and transforming these matrices of pixel values into objects and features. This was initially thought to be an easy task. For instance in 1966 a researcher at MIT expected a group of undergraduate students to be able to solve object recognition during the summer. Predicatively the project lead nowhere [6].

It was not until the late 1990s that modern computer vision started to take form and in the last 15 years there has been an explosion of applications of computer vision, from face-detecting cameras to games and self-driving cars.

Computer vision is divided into several categories such as tracking, classification and recognition. Tracking deals with following an object from frame to frame, algorithms for tracking can make many assumptions such as no changes to illuminance or noise but need to be very fast. Object classification is just starting to become a reality, this deals with the difficult task of deciding what category an object belongs to. For example if an image is displaying a cup or an airplane. Most state of the art methods for classification use deep neural networks which require extensive training and a lot of computing power. The most basic task is recognition which is a task that is both needed for tracking and classification and also field of research on its own.

The focus of this report is on object recognition which will be further described in section 1.2

1.2 OBJECT RECOGNITION

Given an image containing a cup, object recognition aims to answer the question:
"Is this my cup?"

Instead of trying to describe the object geometrically modern object recognition methods focuses on small patches and try to detect and piece them together. You often start with a set of training images that display an object from different angles and let the computer analyze these images and create a database of features it believes to be distinctive. When you then show the computer another image containing this same object it will calculate the features of the image and try to see if any match the features from the training set. If several features match, we have detected the object.

Extracting the features involve many operations such as convolutions and transformations which can make them computationally heavy, therefore speed is of the essence when comparing object recognition algorithms. The speed of recognition is also important in many applications such as motion tracking.

1.2.1 *Invariances*

In order to identify the same object from different images one has to create invariances. This means to perform operations ensuring that the object can be identified in different amounts of light, illumination, rotation and even in different scale. Creating these invariances is therefore one of the most difficult tasks of the object recognition algorithms.

1.3 PROBLEM STATEMENT

With its publication in 1999 the SIFT algorithm by David Lowe was a catalyst of modern object recognition and it has become one of the most implemented object recognition algorithms. Its strength lies in the ability to create and match features independent of rotation, illumination and most importantly, scale.

Recently however a new algorithm called KAZE has been published with claims to outdo SIFT in both precision and speed. While the invariants of SIFT sits on a solid theoretical base, the scale invariance of KAZE depends on a new type of non-linear diffusion which has not been studied as much. In addition to this the data sets used to prove KAZE's superiority were based mainly on computer manipulated images. Therefore doubts have been raised as to whether KAZE can outperform SIFT in real world situations. An accelerated version of KAZE has also been created called AKAZE which the creators claim has comparable accuracy to KAZE while beeing significantly faster.

Since run-time is an important factor in many applications of computer vision a fourth algorithm, ORB, has been included in this study. ORB was developed by the creators of the OpenCV library in 2011 and while it does not claim to be able to outperform SIFT in raw matching capability it is built for speed.

The goal of this report is firstly to compare KAZE to SIFT to see if KAZE is truly a better algorithm for general detection than SIFT. Secondly the report aims to put these four algorithms to the test in real world situations and compare their matching capabilities and speed in order to be able to give recommendations on which algorithm to use for different tasks.

2

BACKGROUND

This chapter contains an introduction to feature detection and explains the definitions and methods used for detecting features in an image. The main idea behind each algorithm is later presented followed by a description of how each algorithm detects an object, the steps that the algorithm follows and the differences between the algorithms.

2.1 FEATURE DETECTION

The algorithms tested in this report use different methods to find and extract interesting points in an image. Such interesting points are detected using a point detector which aims to find points of the image that change in, for example, colour, brightness or direction. In computer vision these points are seen as the most interesting parts of an image and they are called features. The goal is to detect features that have a high repeatability. This means that the feature is likely to be detected again whenever we look at this object.

Extracting features is the most fundamental computation in each algorithm and therefore the first computation performed before starting with the creation of invariances. There are three methods used for detecting features: edge detection, corner detection and blob detection but the algorithms compared in this report use either corner detection, blob detection or a combination of both.

2.1.1 *Corner Detection*

Corner detection was introduced in 1977 when the first corner detector was created [7]. Since then many corner detectors have been developed, the most popular being the Harris corner detector, but none of them manages to exclusively detect corners.

Corners are points formed from the intersection of two or more edges (showed in figure 1). Edges usually describe the boundaries of different parts of an object or two objects and through looking at the direction of those edges corners can be detected. Since a corner detector does not only detect corners but other interesting features such as line endings, other methods have to be used to make sure that the points detected are real corners. On the other hand, corner detectors have the ability to find the same corner in different images which provides a high detection repeatability.

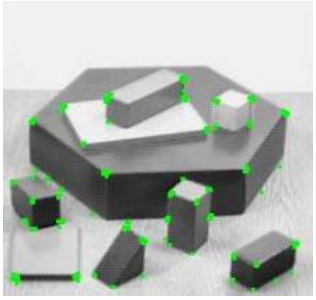


Figure 1: The green circles show the corners detected in the image [11]

2.1.2 *Blob Detection*

Blob detection started around the 1990's when attempts where made to detect the same corners in images of an object in different scales.[8] In 1993 professor Tony Lindeberg at KTH published a paper using a mathematical method to handle the variation of features in different scales. This paper became the foundation of multi-scale detection.

Corner detectors could not detect, with accuracy, features in multi-scaled images (see figure 3) therefore a new method had to be created to deal with changes in image scale. This was an important matter to overcome since poor feature detection could lead to inaccurate detection of an object.

Blobs are regions in which a group of pixels share the same properties. Each region has different properties in comparison to its neighbour regions, thus making each blob different from every other. A blob is represented, mathematically, as a pair consisting of a saddle point and one extremum point making it look like a peak in the frequency domain.

Blob detectors are classified based on the mathematical methods that they employ. The detectors based on differential methods such as the Laplacian of the Gaussian, detect blobs by deriving a function with respect to its position. The ones based on local extrema methods find the local maximum or local minimun of a function.

One of the algorithms that uses a blob detector based on a differential method is SIFT which will be further described in section 2.2

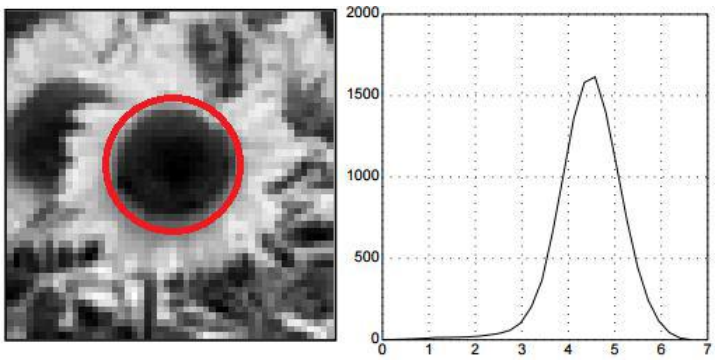


Figure 2: Blob detection [15]

2.2 SIFT

The Scale-invariant feature transform (SIFT) algorithm, was developed in 1999 by David Lowe[1] and has since become one of the most popular algorithms for object recognition. It was later copyrighted by David Lowe.

Before SIFT, most object recognition algorithms were based on Harris corner detection. These algorithms were invariant to rotation which implied the possibility of recognizing an object regardless of its image being rotated or not. However they were not invariant to scale, because when the scale of an image changes the same corner may not be a corner anymore as shown in Figure 3.

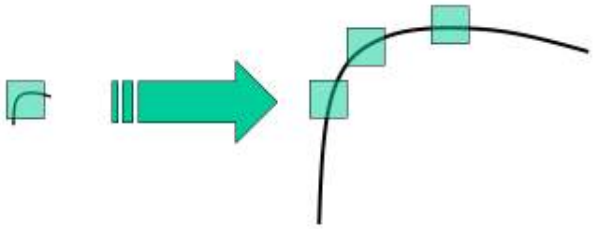


Figure 3: Scale invariance [9]

SIFT transforms image data into linear scale-invariant coordinates that are relative to local features. These features are invariant to image translation, scaling and rotation and robust to change in illumination and viewpoint. The algorithm generates a large number of features that cover the image[1] using blob detection.

For object matches to be verified there has to be at least 3 features that perfectly agree on an object.[1]

The SIFT algorithm has four major steps for object recognition: Scale-space extrema detection, Keypoint localization, Orientation Assignment and Keypoint description.

2.2.1 *Scale-Space Extrema Detection*

The algorithm begins by searching over all image locations and scales for possible interest points (pixels) that can be invariant to scale and orientation.

To choose these points, different blurred versions of the original image are produced using the Gaussian scale-space kernel:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{1}$$

The Gaussian scale-space kernel which convolved with an Image produces a blurred version of the original image.

After creating the blurred images, the algorithm uses linear diffusion (see equation 2) to create difference-of-gaussian (DoG) images and performs down sampling. The DoG is an approximation of the Laplacian of the Gaussian method which has been thoroughly studied over the last few decades. Each sample point (pixel) in the DoG images is then compared to its eight neighbours in the current image and then to its nine neighbours in the scale above and below. This means that each pixel will be investigated

in several different scales and the candidate keypoint will be taken from the scale that gave the highest measurement in the frequency scale, causing scale invariance.

$$DoG(x, y, \sigma) = L_1(x, y, k_1\sigma) - L_2(x, y, k_2\sigma) \tag{2}$$

Difference of Gaussians or Linear Diffusion is an aproximation of Laplacian of the Gaussian method which involves the subtraction of one blurred version of an original image (L_1) from another, less blurred version of the original (L_2).

2.2.2 *Keypoint Localization*

Since too many keypoints are produced in the previous step, it is necessary to reject those that are deemed unstable. In this case instability means that the keypoint has low contrast or is caused by noise. Secondly edges that have a poorly detected location have to be rejected since these edges might not be robust to noise.

The rejection is done by investigating the potential keypoint's surroundings through the use of the Harris corner detector, which detects large gradients (derivatives) in all directions.

2.2.3 *Orientation Assignment*

In this step, orientations are assigned to each keypoint so that invariance to image rotation is achieved. The assignment is done based on local image gradient directions and magnitude. This can be seen in figure 4

An orientation histogram is created with all gradients around the region of each keypoint. The peak with the dominant direction in the histogram will be set as the orientation for that keypoint. If several peaks of the same magnitude are found then multiple keypoints with different orientations, but same location and scale, will be created at that peak.

2.2.4 *Keypoint Description*

Finally, a description of each keypoint is computed to create invariance to illumination. These descriptions are local image gradients around a keypoint that are measured at a selected scale. Then an orientation histogram is created and represented as a vector to form the keypoint's descriptor.

Several measures are taken and the vectors are normalized to values between one and zero. This normalization eliminates the constant that is multiplied by the gradients when there is a change of contrast in the image. This achieves robustness against distortion, contrast and change in illumination.

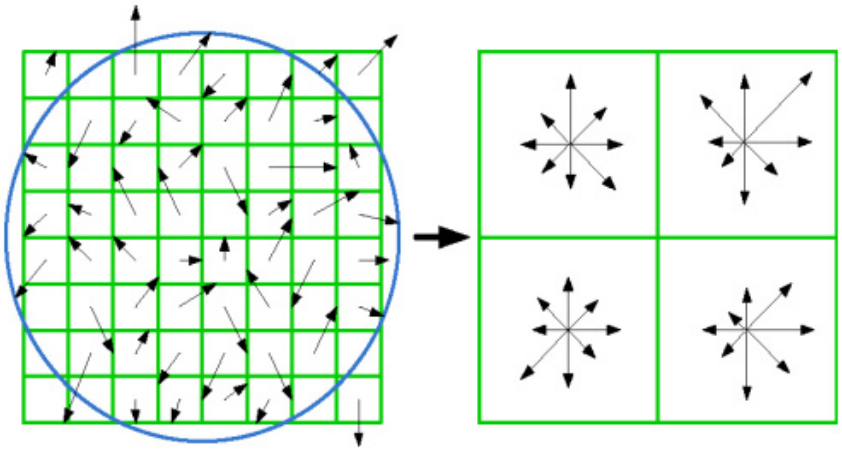


Figure 4: To the left: the representation of the image gradients created at the Orientation Assignment step. The grid split into four parts and for each part the gradients in each direction are summed up.
To the right: the descriptors obtained from the gradients. [1]

2.3 KAZE

The KAZE algorithm was developed in 2012 and it is in the public domain. The name comes from the Japanese word *kaze* which means wind and makes reference to the flow of air ruled by nonlinear processes on a large scale.[2]

The idea behind the creation of this algorithm was to detect and describe 2D features in a nonlinear scale-space extrema so as to obtain a better localization accuracy and distinctiveness.[2] The Gaussian blurring used in other object recognition algorithms, such as SIFT, does not respect the natural boundaries of objects since image details and noise are smoothed to the same degree at all scale levels.

To make blurring adaptive to image features, KAZE makes use of nonlinear diffusion filtering alongside the AOS (Additive Operator Splitting) method. With this filtering the image noise is reduced but the object boundaries are kept, as seen in Figure 5.

For object recognition KAZE follows mainly the same four steps as SIFT but with some differences in each step. These differences are described in the following subsections.

Figure 5: Blur with Difference of Gaussian (left), Blur with non-linear diffusion filtering (right) [2]

2.3.1 *Nonlinear Scale-Space Extrema*

This algorithm uses nonlinear diffusion filtering combined with a conductivity function instead of the Gaussian scale-space kernel used in SIFT. This conductivity function was proposed by Perona and Malik and takes the gradient of a Gaussian smoothed version of the original image (∇L_σ) as a function of time. The aim of this is to obtain features that have higher repeatability and distinctiveness than SIFT.[2]

$$c(x, y, t) = g(|\nabla L_\sigma(x, y, t)|) \tag{3}$$

Perona and Malik's conductivity function where t is time and L an original image.

Contrary to SIFT, which uses the Difference of Gaussians (DoG) to process the blurred images, KAZE uses AOS Schemes to process them since the images build a system of partial differential equations (PDEs). KAZE also uses the original image to create the blurred images and does not perform down-sampling as done in SIFT.

2.3.2 *Normalized Keypoint Localization*

KAZE computes the response of the scale-normalized determinant of the Hessian (DoH) matrix (4) since the DoH is a blob detection method with automatic scale selection. The responses with the maximum value make for the possible keypoints.

$$L_{Hessian} = \sigma^2(L_{xx}L_{yy} - Lxy^2) \tag{4}$$

Scale-normalized Determinant of the Hessian (DoH) where: L_{xx} the horizontal derivative, L_{yy} the vertical derivative and L_{xy} the cross derivative at scale σ

2.3.3 *Vector Orientation Assignment*

The assignment is performed in the same way as SIFT but instead of representing the gradients in an orientation histogram, the gradients are represented as points in a vector space.[2] Then the longest vector with the dominant orientation is assigned as the orientation of that keypoint, as seen in figure 6

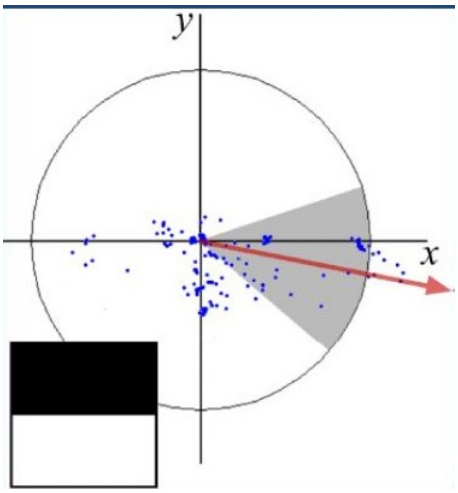


Figure 6: The keypoint surroundings represented as a circle, the orientation represented as a red arrow and the keypoint represented as the center of the circle.[5] The black/white square is the representation of the derivative in $y - axis$ in the frequency domain

2.3.4 *Modified SURF Keypoint Descriptor*

An adapted version of the Speeded Up Robust Features (SURF) descriptor is used since the descriptors have to work in a nonlinear scale-space model. The derivative responses are calculated and summed into a keypoint descriptor vector [2] and then the vector is centered at the keypoint. Finally the descriptor is normalized into a unit vector.

Since KAZE computes multi-scale derivatives (gradients) for each pixel it is more expensive to compute than SURF, though still comparable to SIFT, but it saves computational efforts of keypoint description because the same set of derivatives are used for the description of a keypoint.

2.4 AKAZE

KAZE's method of using non-linear diffusion filtering requires it to solve a series of PDEs. This can not be done analytically forcing KAZE to use a numerical method called an AOS scheme to solve the PDEs. However this process is computationally costly and therefore an accelerated version of KAZE was created. This version is called Accelerated KAZE or AKAZE.

The accelerated KAZE algorithm works in the same way as KAZE but in contrast to KAZE, it uses a faster method to create the non-linear scale-space caled the Fast Explicit Diffusion (FED). The FED is a decomposition of box filters and uses the original image for the first cycle, then the resulting image is used for the following cycle.

In addition to the use of FEDs, AKAZE uses a binary descriptor to further increase speed. To create the keypoint description, AKAZE makes use of a modified version of the Local Difference Binary (LDB) descriptor. The LDB follows the same principles as the BRIEF algorithm but computes binary tests in the region surrounding the keypoint.

2.5 ORB

The Oriented Fast and Rotated Brief (ORB) algorithm is based on the BRIEF keypoint descriptor and the FAST keypoint detector since both algorithms are computationally fast. It was presented in 2011 to provide a fast and efficient alternative to SIFT.[4] Contrary to SIFT, it is free from licensing restrictions and due to its fast computation it can be used on low-power devices.

2.5.1 *The FAST Keypoint Detector*

ORB starts with using FAST to detect keypoints in the image and Harris corner detector to order the detected keypoints and exclude non-corner keypoints. Then a scale pyramid is used to produce multi-scale features in order to obtain scale invariance.

Since FAST does not produce orientations, the Intensity Centroid (IC) technique is used. The IC constructs a vector from the corner's center to a centroid and then the orientation is computed.

2.5.2 *The BRIEF Keypoint Descriptor*

The BRIEF keypoint descriptor produces binary descriptions of keypoints. This is a quick and efficient way of storing and comparing keypoints however it has poor orientation performance. In order to improve this, a steered version of BRIEF (sBRIEF) is created by multiplying the matrix of binary descriptions with their angle of orientation.

To transform the sBRIEF into rotated BRIEF (rBRIEF) a search using a greedy algorithm is performed. This process is described in detail in [4].

3

METHOD

In order to compare the algorithms three things are needed, implementations of the algorithms, a program that runs the tests and a data-set of training and test images. For the implementations the choice was made to use the open source software library OpenCV.

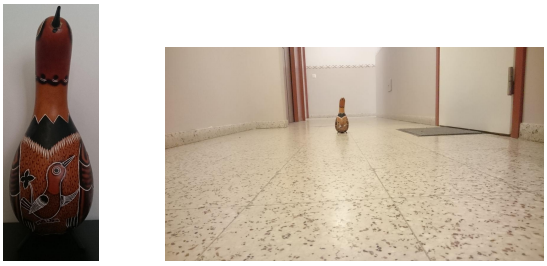
OpenCV is a software library aimed at real-time computer vision. It was created by the Russian researcher Gary Bradsky while working at Intel in 1999. Since it is cross platform and free under the BSD-licence it is used by both hobbyist and researchers. Among the users of OpenCV are companies such as Google, IBM, Toyota and Honda. OpenCV is written in C++ but has bindings for several other languages such as Python, Java and MATLAB.[10]

For the program running the tests Python was used. Since Python is a high level language development is quick and easy. The problem with choosing a high level interpreted language such as Python is the loss of speed compared to low level compiled languages such as C or C++. However this loss of speed is greatly diminished since OpenCV for Python contains bindings to the algorithms implemented in C++. To further decrease the loss of speed from using Python the NumPy extension[13] is used. Numpy is an optimized library for numerical operations and is recommended when using OpenCV with Python.

The usual way to create data-sets for comparing object detection is using computer manipulation. This means that you take a very limited number of images but manipulate these using a computer to change the scale, rotation, brightness and noise to create a large amount of test images. A comparison between SIFT and KAZE has already been done using the computer manipulated test data created by Mikolajczyk with KAZE proving superior performance [2]. However the next step is to make a comparison using data-sets of real-world images with actual differences of scale, rotation and illumination. It is important to not only rely on tests using computer manipulated images since the goal of these algorithms are to detect objects in real photos. Due to the high amount of work needed to create real world data-sets few are to be found online and it proved necessary to create a new data-set to suit the goals of this report.

3.1 DATA-SET CREATION

The goal was to create a data-set testing the algorithms invariances to rotation, illumination and scale without using computer manipulation. The data-set was divided into four different categories, a tests of rotational invariance, a test of invariance to illumination, a test of scale invariance and finally a test of general detection capabilities.



(a) (b)

Figure 7: Training image (a) and test image (b) of an object side by side

All tests consisted of a training image of the object taken with simple background followed by a series of test images of the object in different settings with increasing difficulty. The goal of each test was for the different object recognition algorithms to detect the object in each training image using the features extracted from the test image. The training images were cropped in order to remove features that were not part of the object. An example of a training image and test image is shown in figure 7.

When choosing objects to use for testing, blank objects were rejected due to potential issues with glare from differing light sources. Objects with deformable surfaces were also avoided except for two stuffed animals since early testing proved that all algorithms had significant issues with these. In total ten different objects were chosen ranging from books to remote controls to stuffed animals all containing an abundance of features to match. With ten different objects the risk of the results being influenced by one object favouring a particular algorithm was also minimized.

3.1.1 *Test of rotational invariance*

The goal of this test was to show invariance to rotation. To do this, the tested object was put on the floor to avoid difference of angle. Then four pictures of the object were taken with object rotated roughly 90 degrees each time.

Figure 8: The images above show how a test of rotational invariance was designed.

3.1.2 *Test of scale invariance*

One of the major differences between the algorithms compared is how they cause scale invariance. Since scale can be simulated by distance the test of scale invariance consist of increasing the distance between the camera and the object to be detected. By using an empty hallway the lighting were kept at a roughly constant level and the tiles on the floor helped in keeping the object at a constant angle. An example of this can be shown in figure 9.

Figure 9: The four images above show how a test of scale invariance was designed.

3.1.3 *Test of illumination invariance*

The main goal of testing illumination invariance was to ensure that the algorithms can handle everyday changes in light such as the change between being outside and indoors. This was accomplished by placing the object in a brightly lit indoor scene, a sunny outdoor scene, a shadowy outdoor scene and lastly a dark indoor scene.

3.1.4 *General detection tests*

A number of general detection tests of objects in cluttered scenes were also made. The goal of this was to test the algorithms combined capabilities and to test more realistic scenarios. The tests involved placing the object in scenes with different amounts of cluttering while varying distance, light and rotation.

3.2 THE TESTING SOFTWARE

The testing software used was a Python program using the algorithms from their OpenCV implementations. Since ORB was developed by OpenCV staff it is included in the OpenCV library by default. KAZE and AKAZE too are included in the standard

OpenCV library. SIFT is however copyrighted and is no longer part of the standard OpenCV library but could be found in a contributions package.

The program was designed to first load a training image of an object followed by a series of testing images. Secondly, four sets of keypoints using the four chosen algorithms were extracted using a call to their OpenCV implementations and the time it took to extract the features were measured for each algorithm. Thirdly, the set of keypoints for each algorithm were put through a matcher. The program used a brute-force matcher to find the two best matches for each keypoint followed by a ratio test, as suggested by David Lowe[1], rejecting matches with a relative distance greater than 75%. SIFT and KAZE used the euclidean distance while ORB and AKAZE used the hamming distance since they use a binary descriptor.

The final step consisted of using OpenCV's findHomography and perspectiveTransform methods in order to take the matches and show the position of the object in the training image. For this at least 10 matches were required. An example of the output is shown in figure 10.

All tests were ran on a machine equipped with an Intel Core i5-4670K CPU clocked at 3.40GHz and 8GBs of RAM. It used the latest version of OpenCV (3.1) and a 64 bit version of Ubuntu 15.10.

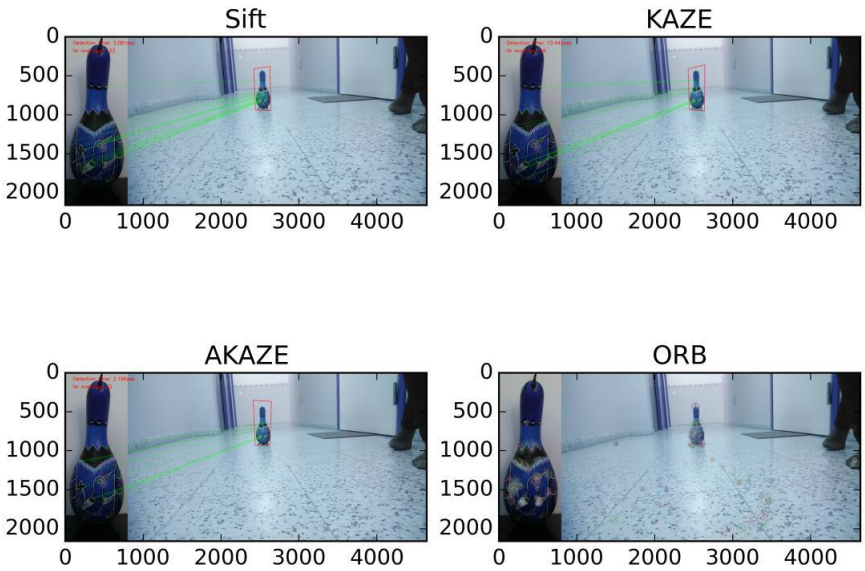


Figure 10: The result from one of the test images in a scale test of the wooden bird. Here SIFT, KAZE and AKAZE managed to detect the object and have marked it's location with a red square and the matches with green lines.

4

RESULTS

In this section the results for each invariance test, rotation, illumination and scale are presented. The results of the general detection test are also presented along with the combined score from all tests. In total 172 tests were performed over eleven different objects. For the tests of scale invariance and general detection, test images where none of the algorithms were able to detect the object were excluded in order to focus on their relative capabilities. The reason for excluding images in these tests were that while rotation and illumination only varies by a fixed amount in normal situations, scale and general detection has a potential for unlimited variation and this was simply a way to limit the difficulty of these tests.

4.1 TEST OF ROTATIONAL INVARIANCE

All algorithms claim to be able to recognize the object regardless of rotation. This claim was pretty much proven true for all algorithms except for ORB. SIFT, KAZE and AKAZE were all able to detect the object in more than 85% of the tests while ORB only had a detection rate of 62%. It is important to note however that the objects that ORB had issues with was mainly the two stuffed animals which had slightly deformable surfaces and were susceptive to small changes in non-planar rotation due to their unstable base. Also notable is that AKAZE outperformed KAZE slightly. 26 tests were performed on seven objects, the results can be seen in figure 11 and table 1 below.

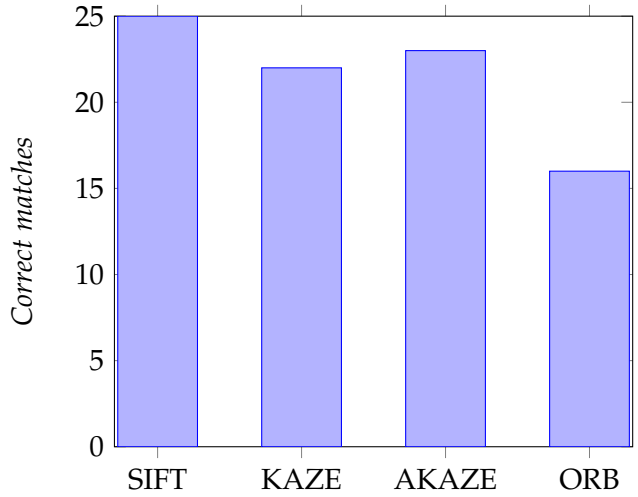


Figure 11: Showing the number of correct matches for the test of rotational invariance

Table 1: Rotational invariance

	SIFT	KAZE	AKAZE	ORB
Matches	25	22	23	16
Misses	1	4	3	10
Matches (%)	96	85	88	62

4.2 TEST OF ILLUMINATION INVARIANCE

For illumination invariance, five objects were used to perform 20 tests. All algorithms proved to have a high invariance to changes in illumination, however SIFT, KAZE and AKAZE pulled ahead of ORB again. The results are shown in figure 12 and table 2.

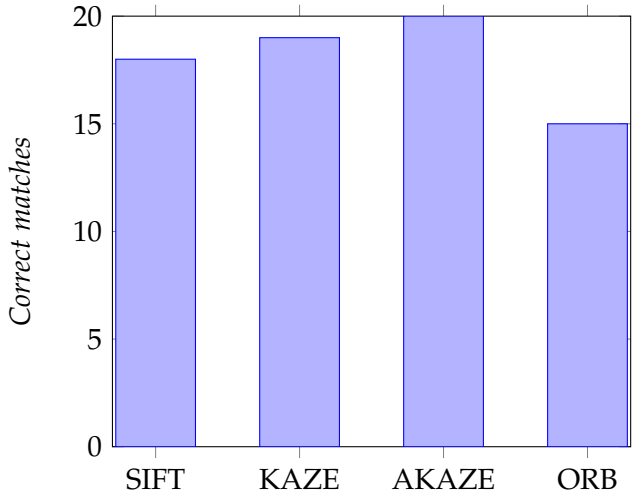


Figure 12: Showing the number of correct matches for the test of illumination invariance

Table 2: Illumination invariance

	SIFT	KAZE	AKAZE	ORB
Matches	18	19	20	15
Misses	2	1	0	5
Matches (%)	90	95	100	75

4.3 TEST OF SCALE OF INVARIANCE

Ten objects were used to perform 108 tests of scale invariance. However since the goal was to compare the algorithms relative capabilities the fourteen images where none of the algorithms were able to detect the object were excluded from the result. This left 94 tests.

While SIFT proved to be superior in detecting objects that were far away, AKAZE came very close and again was able to outperform KAZE. ORB however, appeared to have difficulties in detecting objects that were far away and only managed to detect the object in 40% of the tests. The results can be seen in table 13 and figure 3 below.

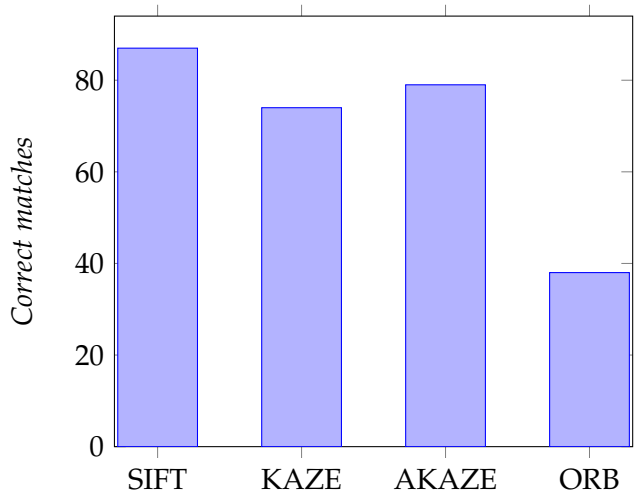


Figure 13: Showing the number of correct matches for the test of scale invariance

Table 3: Scale invariance

	SIFT	KAZE	AKAZE	ORB
Matches	87	74	79	38
Misses	7	20	15	56
Matches (%)	93	78	84	40

4.4 GENERAL DETECTION TEST

The test of general detection was composed of 47 test images over seven objects, five test images were excluded since none of the algorithms were able to detect the object in them leaving a 42 images to base the result on. In this test SIFT proved vastly superior to the other algorithms with KAZE and AKAZE sharing a distant second place. The results can be seen in figure 14 and table 4 below.

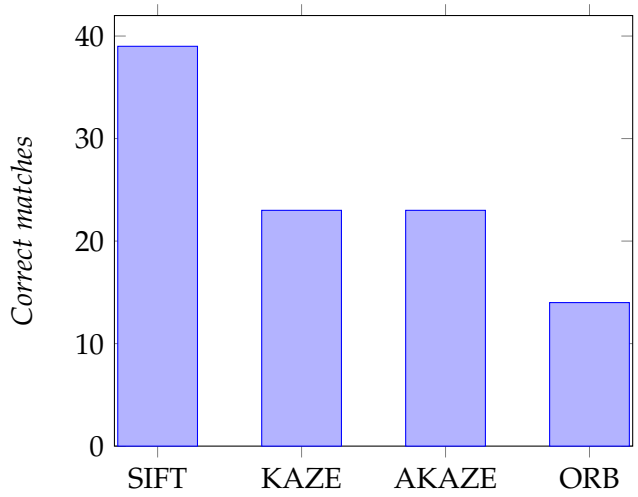


Figure 14: Showing the number of correct matches for the general detection test

Table 4: General detection

	SIFT	KAZE	AKAZE	ORB
Matches	39	23	23	14
Misses	3	19	19	28
Matches (%)	93	55	55	33

4.5 COMBINED RESULTS

The result of each test as a percentage is shown in figure 15. Also shown is the results gained from adding all tests performed (figure 17)

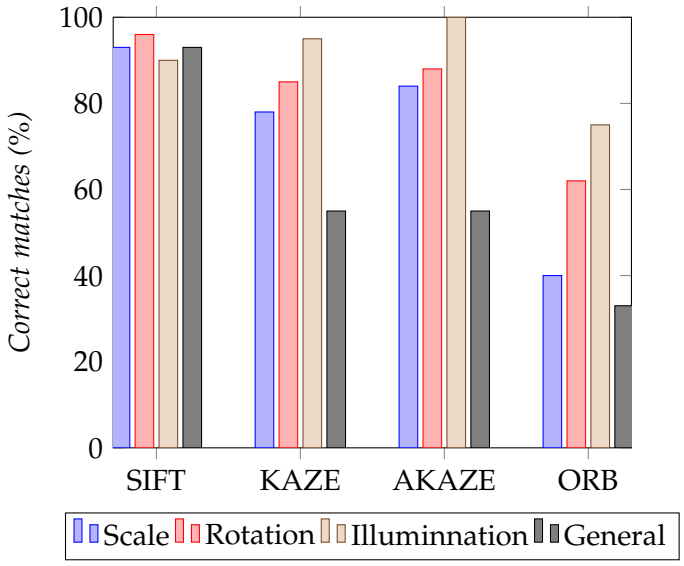


Figure 15: Showing the results of all tests performed

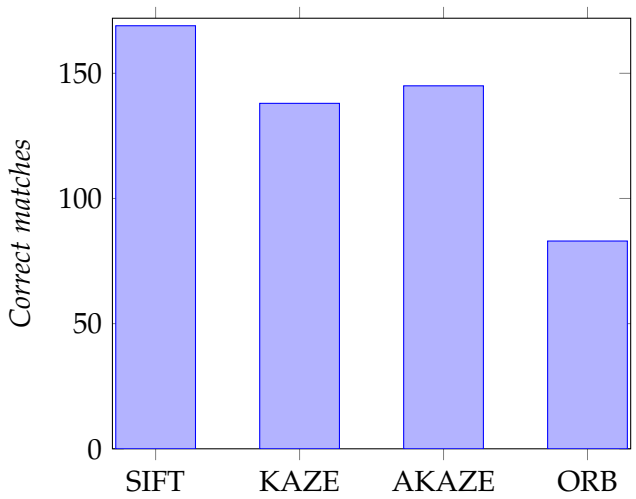


Figure 16: Showing the total number of correct matches for each algorithm on the total amount of tests

4.6 RUN-TIME

The run-time of the algorithms differed greatly depending on the size of the image. However the ratio between the algorithms remained the same. KAZE was about three times slower than SIFT while AKAZE was slightly faster than SIFT. ORB however was more than ten times faster than all other algorithms and about 100 times faster than KAZE.

Table 5: Run-time results

Algorithm	Uncompressed image (around 3MB)	Compressed image (around 100kb)
SIFT	2.8s	0.5s
KAZE	9.4s	1.0s
AKAZE	1.8s	0.25s
ORB	0.1s	0.02s

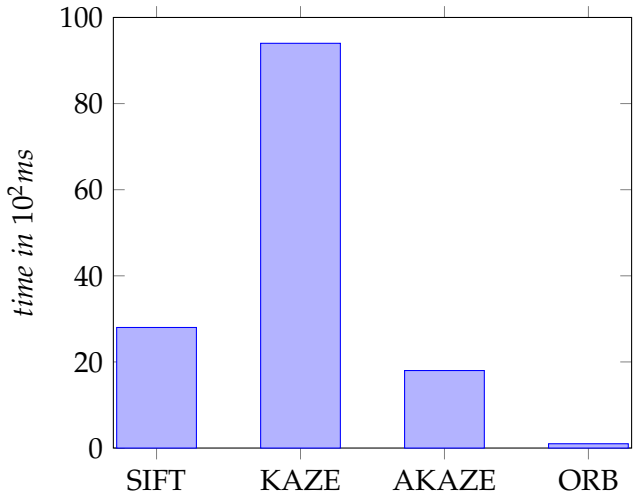


Figure 17: Showing the runtime (in $10^2 ms$) for the algorithms on uncompressed images

5

DISCUSSION AND CONCLUSION

5.1 DISCUSSION

Object recognition in real-world images remains a difficult task. Even for the simpler tests of invariance to illumination and rotation the algorithms were unable to obtain a 100% detection rate. It is however fair to say that the three slower algorithms, SIFT, KAZE and AKAZE all proved that they can handle normal changes in rotation and illumination. As explained in section 2.5 the algorithms that ORB was based on are not built to provide invariance to rotation and this is reflected in the results. However these issues manly appear when the object have a slight non-planar rotation in addition to the planar rotation as mentioned in section 4.1.

What was surprising was the results from the general detection tests. While SIFT, KAZE and AKAZE all showed good accuracy in the invariance tests, the general detection tests proved different. On the tests of illumination and rotational invariances KAZE and AKAZE had similar results to SIFT and on the test of scale invariance their results were no more than 16% lower . However the difference in results on the general detection tests were much larger. Here SIFT proved vastly superior to the other algorithms in terms of detection with a lead of 41% over KAZE and AKAZE and 67% over ORB.

We believe this difference might have two explanations. First, because SIFT rests on a well studied theoretical base this gives the algorithm substantial robustness. Since real-world images always contain a large amount of variables, this need for robustness is proved by the general detection tests where in addition to a cluttered background the object might be rotated and in varying amounts of light.

A second explanation could be a third type of invariance that was not tested in this report, invariance to affine rotation. Affine rotation is when the object is rotated in a way that exposes new parts of the object to the camera. While none of the algorithms claim to have invariance to affine rotation, and complete invariance is impossible, SIFT might have more robustness to this type of rotation. We believe this might explain why the difference between SIFT and the other algorithms where so large in the test of general detection.

One of the main goals of this report was to investigate the claims made by the creators of KAZE, that it could outperform SIFT, and investigate the new type of non-linear diffusion method used by KAZE to obtain scale invariance. The results of this report contradict these claims with SIFT outperforming KAZE in all tests. Due to the high number of test images used for the test of scale invariance we believe that the results presented in this paper should be seen as more reliable than those presented by the

creators of KAZE when considering real-world applications. In addition to this, as previously mentioned, KAZE proved inferior to SIFT in the general detection tests.

We were however pleased to see how well AKAZE performed in the first three tests of illumination, rotation and scale invariance. AKAZE was far faster than KAZE and even faster than SIFT while having better accuracy than KAZE and close to that of SIFT.

ORB proved to play in a different game, while it had the lowest accuracy in all tests it was more than 10 times faster than the other algorithms proving that it was build for speed, not accuracy.

5.1.1 *Limitations*

The results of this study contradict those published by the creators of KAZE, it is therefore necessary to discuss the limitations of this study. Due to the time required to create the tests the number of testing images were limited to 172 test images. This means that the percentages obtained from the study might not be fully reliable. In addition to this it proved hard to isolate the invariances when creating the tests. The test of rotational invariance often contained slight differences in illumination and scale and the same goes for the other tests. This is especially relevant when looking at the result of ORB in the test of rotational invariance where the problem likely was due to other factors than planar rotation.

Another limitation was the lack of an affine rotation test. The inclusion of this test might have helped explain the results on the general detection test. However even with these limitations the results were very clear on the hierarchy of the algorithms and enable us to give solid recommendations on what algorithm to use for which propose.

5.1.2 *Future research*

While this study was able to answer the questions raised in the problem statement it left us with a new question. Why is SIFT so superior to the other algorithms when it comes to general detection? This appears to be a hard question to answer since each algorithm has many steps that each would have to be investigated. However one could start by doing a thorough investigation of robustness to affine rotation.

Another question is why AKAZE showed better performance than KAZE?. We could see that creating a binary descriptor provides high accuracy but the purpose of AKAZE was to be faster than KAZE not better. So further research in this area has to be done.

5.2 CONCLUSIONS

Computer vision still has a long way to go before being able to match the capabilities of a human child and progress is slow. Though SIFT is by far the oldest of the algorithms tested in this study it still proved to have superior performance to the newer algorithms. The claim from the creators of KAZE, that it could surpass SIFT are not supported by the test results from this study and in fact AKAZE outperformed KAZE in most cases leaving few reasons to chose KAZE over AKAZE. In general, SIFT had among the highest score in each tests and if one was to use one algorithm for general detection purposes SIFT would be the clear choice. There is however the issue with

the copyright. If this is an issue the second best performing algrithm is AKAZE. While having lower detection rates on the general detection test than SIFT it still had high detection rates on most tests and managed to score 100% on the test of illumination invariance. AKAZE is also faster than SIFT which can be a significant advantage on devices with low computational power.

If speed is of the essence, for example if objects need to be tracked or for use in a moving car ORB might the correct choice. While having lower accuracy than the other algorithms it can still detect the object in simpler cases and is extremely fast. The lower detection rate can also be mitigated by prior knowledge of a scene. If you for example know that the traffic signs will appear in the top half of the left part of the image, detection will be significantly easier. This is also true for tracking since you have prior knowledge of the scene and the location of the sought after objects.

Based on the results from this report we are unable to recommend the use of KAZE over AKAZE in any circumstance. AKAZE if both faster and had a higher total score.

BIBLIOGRAPHY

[1] David G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, Computer Science Department, University of British Columbia 2004.

[2] Pablo F. Alcantarilla, Adrien Bartoli, and Andrew J. Davison, *KAZE Features*, Universite d'Auvergne, Clermont Ferrand, France, 2012

[3] Pablo F. Alcantarilla, Jesus Nuevo and Adrien Bartoli, *Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces*, Bristol, UK, 2013

[4] Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary Bradski, *ORB: an efficient alternative to SIFT and SURF*, Willow Garage, Menlo Park, California, 2011

[5] Herbert Bay, Andreas Ess, Tinne Tuytelaars and Luke Van Gool, *Speeded-Up Robust Features*, 2008

[6] Seymour Papert, *The Summer Vision Project*, Massachusets Institute of Technology, 1966

[7] D. Parks and D.P. Gravel, *Corner Detection*, University of McGill, 2004

[8] Anne Kaspers, *Blob Detection*, Image Science Institute, UMC Utrecht, 2011

[9] *OpenCV SIFT tutorial* `http://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html#gsc.tab=0` Accessed 2016-05-11

[10] *About OpenCV* `http://opencv.org/about.html` Accessed 2016-05-11

[11] *OpenCV Harris Corner Detection tutorial*, `http://docs.opencv.org/3.1.0/dc/d0d/tutorial_py_features_harris.html#gsc.tab=0` Accessed 2016-05-11

[12] *OpenCV Python tutorials* `https://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_tutorials.html` Accessed 2016-05-11

[13] *Numpy.org* `http://www.numpy.org/` Accessed 2016-05-11

[14] Krystian Mikolajczyk and Cordelia Schmid, *A Performance Evaluation of Local Descriptors*, 2004

[15] Tony Lindeberg, *Feature Detection with Automatic Scale Selection*, KTH, 1996