Universitetet i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Information Technology – Automation and Signal processing	Spring semester, 2019 Open/~~Confidential~~
Author: Niroshan Thangalingam	..... (signature of author)
Programme coordinator: Ivar Austvoll Supervisor(s): Ivar Austvoll	
Title of master's thesis: Detection of interest points Norwegian title: Deteksjon av egenskapspunkter	
Credits: 30	
Keywords: - Interest point detection - KAZE, SURF and SUSAN - Computer vision - Matlab - Synthetic image - Real-life image	Number of pages: 97 + supplemental material/other: 9 Stavanger, 15.06.2019

University of Stavanger

Master's thesis

Detection of interest points

Author:
Niroshan Thangalingam

Supervisor:
Ivar Austvoll

Faculty of Technology and Science
Department of Electrical Engineering and Computer Science
University of Stavanger

June 2019

Abstract

This thesis reviews methods for interest point detectors to detect salient points that have an expressive structure such as corners or blob features. Properties of good interest point detectors are that interest points are being detected regardless of geometric or photometric changes such as scaling, rotation, noise, and brightness in the image.

Two of the more common interest point detectors are Speeded Up Robust Features (SURF) and Smallest Univalue Segment Assimilating Nucleus (SUSAN), both being essential milestones in feature detection and are still wildly used in various applications in fields like interest point detection. In contrast, research on KAZE algorithm has been scarce. However, interest in further research has been gaining traction within the community. This stands to the reason why the focus will be on how the KAZE detector will perform compared to SUSAN and SURF detectors. KAZE was selected as it is quite new compared to the others, and the fact that it uses nonlinear scale space compared to algorithms such as SURF that uses Gaussian scale space. Furthermore, three images were used for the experiments, where the first image is a synthetic image with different geometrical shapes. Second and third images are real-life images of a building and a boy, respectively.

The results of the synthetic image with different geometrical shapes show that the detectors have different results on each experiment carried out for this type of image. While for the real-life images, KAZE and SURF proved vastly superior compared to SUSAN. This led to the conclusion that the detectors should be selected according to the type of experiment carried out. Additionally, in terms of the detector speed, SURF was the fastest detector compared to the other detectors.

Preface

This report presents the process and results of a master thesis given by the University of Stavanger within the field of automation and signal processing. The thesis was part of the subject called ELEMAS, which constitutes of 30 out of 120 study points necessary to receive a master degree. The main purpose of this thesis was for the student to show their knowledge within the field of automation and signal processing based upon what was taught during the study.

First, I want to thank my supervisor, Ivar Austvoll, for good guidance and constructive feedback throughout this project. The help he gave me was crucial for the completion of this thesis, and his supervision guided me through the steps of the process.

I would also like to thank my family and friends for the support throughout this thesis as well.

Niroshan Thangalingam

Stavanger, June 15th, 2019

Contents

Figures

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter A

List of Abbreviations

AOS	Additive Operator Splitting
BRISK	Binary Robust Invariant Scalable Keypoints
DoG	Difference of Gaussian-filter
DoH	Determinant of Hessian
FAST	Features from Accelerated Segment Test
FED	Fast Explicit Diffusion
PDE	Partial Differential Equations
M-SURF	Modified version of SURF
NDF	Nonlinear Diffusion Filtering
NMS	Non Maximum Suppression
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SURF	Speeded Up Robust Features
SUSAN	Smallest Univalue Segment Assimilating Nucleus
U-SURF	Upright version of SURF
USAN	Univalue Segment Assimilating Nucleus

1. Introduction

Point- or blob features are frequently applied detectors types where detection is a primary asset of the workload within computer vision. Whether it be detecting faces of various live feeds at the airport or preventing potential hazardous situations for workers when moving about on oil platforms, the possibilities are endless. Nowadays, detectors are more versatile with the introduction of KAZE, although depending on the method of development, such as non-diffusion filtering, or DoG (Difference of Gaussian-filter). Alongside KAZE, there are numerous different approaches to detect interest points from digital images. Interest points are a series of point in an image with an expressive texture such as corner which is an intersection point between two edges or obvious point that is distinguished from the rest of the image. The characteristics of a good interest point detector are to detect many true interest points as possible and no false interest points. Besides, an interest point detector should also be robust to different transformations. In this project, three interest point detectors are chosen to detect interest points, which are further analyzed and tested. The purpose of conducting this analysis is to test how the detectors are detecting interest points at a synthetic image with different figures and real-life images and to analyze what kind of interest points are being detected.

1.1 Feature detection methods

Feature detection is an important element in the field of computer vision. This is due to the depth and size of feature detection approaches and the fact that it is used in many applications like image registration, camera calibration, or object detection. It is important to introduce the essential parts of the feature detection such as images, local features, global features, and the characteristics for good features.

Images are an artificial creation of what the naked eye sees through the lens of the camera, which is made from elements called pixels and vectors. For the human eye to detect something particular inside an image, a simple look is all that is needed. Unlike humans, computer vision analysis the pictures by using specific algorithms to achieve a similar result. In computer vision tasks, images can be represented by using either global features and local features. The global feature is used to describe an image by

one multidimensional feature vector, precisely, the image seen as a whole. The local feature is used to describe the different regions within the image. Moreover, describe the image with a set of features like image regions or keypoints [1].

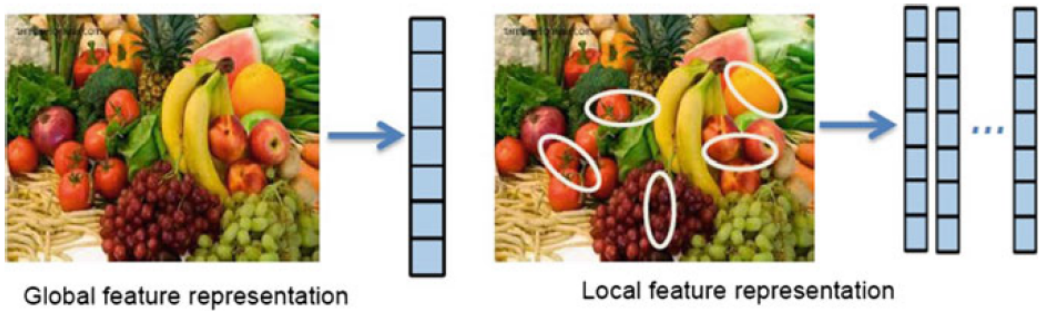


Figure 1.1: Global feature representation (left) and local feature representation (right) [1]

Identifying local features are an essential task within the field of computer vision. Local features can be divided into several categories all depending on what kind of features are desired to detect. The first kind of local features are the ones that are in a specific location, such as building corners, windows, or geometric shapes. These features are usually referred to as interest points or corners. The second kind of local features are edges, which is all about finding boundaries, straight lines, or curves [2]. The last type is the texture that has a unique structure on their surface, which can be found in mountains, buildings, or natural scenes [3]. There are many more local features, but the ones mentioned above are the most important one.

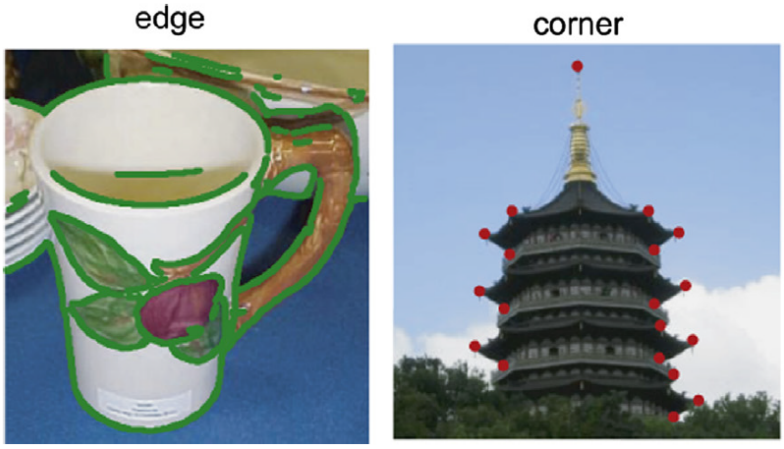


Figure 1.2: Edge detection on a cup (left) and Corner detection on a building (right) [4]

To apply feature detection, the location and sizes of any features must be first found and then extracts this feature from the background. An ideal feature detector should be able to identify and locate these features regardless of their positions, scale, or orientation. History shows that many different approaches are considering different properties for great detection. This idea applies for feature detection in general, whether it is a corner or an edge that is to be detected. Moreover, a good detector is determined by several criteria. As this project focuses heavily on interest points, the characteristics for good point features are as following [5] [1]:

- *Repeatability:* The detected point features are visible in multiple images of the same object taken from different viewpoints.
- *Robustness:* The same point features are detected regardless of scaling, rotation, noise, and brightness changes. Additionally, the point features should be geometric and photometric invariant as well.
- *Distinctiveness:* The detected point features have variations in the intensity pattern, and thereby such features can easily be distinguished from each other.
- *Accuracy:* Accurate localization of the detected point features.
- *Efficiency:* Proficient detection of point features. A detector should be both fast and efficient.
- *Invariance:* Point features are detected independent of scale changes or transformations within an image.
- *Quantity:* All or most of the point features should be detected, so that all objects are detected even the small objects. Note that an ideal number of features depends on the application.

1.2 Overview of feature detection methods

The most common point feature detector is the Harris Detector, developed and published by Chris Harris and Mike Stephens in 1988 [6]. It improves upon the Moravec

detector [1] when detecting small image variations and points near edges. In other words, a desirable detector in terms of repeatability rate and detection can be obtained by changes and alterations of auto-correlations at any orientations. The Harris detector permitted the development and widespread utilization of the Harris-Affine and the Shi-Tomasi/Kanade-Tomasi detectors [7]. SUSAN (Smallest Univalue Segment Assimilating Nucleus) is favorable in application whenever image noise reduction is a vital part of the workload, including corner- and edge detection. The detector uses a circular mask to find intensity differences in the image regions. All the pixels which have the same intensity value as the center pixel is verified as the USAN (Univalue Segment Assimilating Nucleus) [8]. In order to find a corner or edge, a threshold value is set to distinguish the intensity values in the center pixel from the rest. Furthermore, based on the number of identical pixels found in the circular mask, a corner or edge can be found. In addition, the performance of this detector is most satisfactory in the presence of noise by not making use of image derivatives, as well as yield substantial noise rejection. This is easily comprehended by understanding that noise is ignored when it is too small for the USAN function [8]. Both SUSAN and detectors based on the Harris detector are known as corner detector, as they can detect features of any given object within a picture that comprises of corners, or other point-characteristics and compare it to similar images.

There are several well-established algorithms within interest point detection or corner detection like Moravec corner detection algorithm, Harris corner detection algorithm, SUSAN corner detector. For computational efficiency and high-speed performance, Rosten and Drummond proposed FAST (Features from Accelerated Segment Test) corner detector in 2006. The purpose of the FAST algorithm was to develop an interest point detector to be used in 'real time frame rate applications' such as SLAM (Simultaneous Localization And Mapping) [9] on a mobile robot, which has limited computational resources [10]. To detect interest points at an arbitrary pixel, the algorithm compares the intensity between pixels on a circle surrounding with the examined pixel p. If the intensity of the surrounding pixels is all above or all below p, then the algorithm is said to have found an interest point at p. The most exciting advantage of the FAST detector is its computational efficiency as well as the fact that it is several times

faster than the other corner detectors, which enables it to be very suitable for real-time video processing application because of its high-speed performance[11]. Despite these advantages, the FAST algorithm is not robust to high-level noises and is dependent on a threshold.

On the other hand, most of the mentioned methods are not invariant to scale changes. Therefore, a new approach had to be created to deal with changes in image scale. One of the most well-known and widely used today is the SIFT (Scale Invariant Feature Transform) detector [12]. This was a breakthrough proposed by Lowe in 2004 which finds scale invariant features in an image. SIFT is an approach for extracting keypoints from images using local extrema detection and Difference of Gaussians (DoG) which is an approximation of the Laplacian of Gaussian (LoG) [13]. The detector aims to produce scale-invariant features, which means it can detect features at different scale levels [14]. From the results in "*Comparison and Study of Classic Feature Point Detection Algorithm*" [15], the SIFT algorithm has shown positive results in dealing with translation, rotation, scaling and brightness change which make this algorithm more robust and accurate compared to others. Nevertheless, it is slow in terms of high computational cost and execution time [16].

In 2008, Bay, Tuytelaars and Van Gool, published a new algorithm called SURF (Speeded Up Robust Features) that is a faster version of SIFT. The SURF algorithm is based on the same principles and steps as SIFT, but it utilizes a different scheme, and it aims to provide a better and faster result. SURF uses integral images and Hessian determinants to detect interest points. According to the article *Speeded-Up Robust Features (SURF)* [17], Hessian-based detectors are more stable and repeatable.

For that reason, SURF and SIFT are often used as baselines in evaluations of other detectors. BRISK (Binary Robust Invariant Scalable Keypoints) is a detector that is based on the FAST detector. It detects keypoints in a scale-space pyramid by performing non-maxima suppression and interpolation across all scale levels [18]. Recently, a new algorithm called KAZE has been published with claims to surpass SIFT in both precision and speed [19]. The KAZE detector is based on the determinant of Hessian

computed at multiple scale levels. Besides that, the KAZE feature detector is also constructed through nonlinear diffusion filtering, which means that the image is locally adapted to the image. This means that noise, i.e., Gaussian noise, are reduced and object boundaries in a subject image are maintained at the same time [16].

1.3 Thesis objective

The objective of this thesis is to identify interest points within a synthetic image with different transformations and real-life images. Three detectors called KAZE, SURF and SUSAN are chosen for this purpose. The majority of the focus will be on how these detectors are recognizing true interest points for different experiments of geometric or photometric changes such as scaling, rotation, Gaussian noise, and brightness in the synthetic image. After that test these detectors to identify interest points in real-life images.

1.4 Thesis outline

Chapter 1 - Introduction

The reader is introduced to the background and existing algorithms of the thesis topic. The problem statements are also defined, and assumptions and limitation described.

Chapter 2 - Theory

Theory relevant to this thesis is presented.

Chapter 3 - Methods and Experiments

A detailed description of the methods and experiments are arranged to verify the performance of the proposed methods (KAZE, SURF, and SUSAN).

Chapter 4 - Results and discussion

This chapter contains all results obtained from the experiments described in chapter 3. Discussions are based on these results.

Chapter 5 - Conclusions

Conclusions based on the report's topics.

2. Theory

This master thesis addresses various methods utilized in KAZE, SURF, and SUSAN. It is therefore crucial to provide a theory to make a foundation to help understanding this type of detection. The following chapter and sections introduce the main theories of KAZE, SURF, and SUSAN detectors. Most theories are based upon information learned through various articles.

2.1 KAZE features

Originating from the Japanese language, KAZE means 'wind' and therefore an appropriate word describing the flow of air ruled by nonlinear processes on a large scale. The KAZE feature detector algorithm was developed in 2012 by Pablo F. Alcantarilla, Adrien Bartoli and Andrew J.Davison [19]. The main purpose of this algorithm is to detect and describe 2D features through nonlinear diffusion filtering by operating entirely in nonlinear scale space. KAZE features make use of nonlinear diffusion filtering (NDF) (explained in appendix A) alongside Additive Operator Splitting (AOS) schemes (explained in Appendix B) which makes the input image locally adaptive to blurring, and simultaneously not affecting details and edges. Put differently, the KAZE features is an improvement of SIFT using nonlinear diffusion filtering instead of the difference of Gaussian scale space for detection purposes. This approach is preferred since algorithms like SURF adds the same degree of smoothness on details and noise at all scale levels, which affects the localization accuracy of interest points. The KAZE feature algorithm can be divided into three steps [20]:

1. Build a nonlinear scale space using AOS schemes and conductivity functions.
2. Detect 2D features by computation of the determinant of Hessian response and execute non-maxima suppression in images within the nonlinear scale space.
3. Calculations of main orientation and a descriptor for all interest points.

More detailed illustrations of these steps are shown through the flowchart in Figure 2.1. This figure shows the step by step flowchart of the KAZE algorithm. Additionally, the flowchart was made with regard to the KAZE features [19].

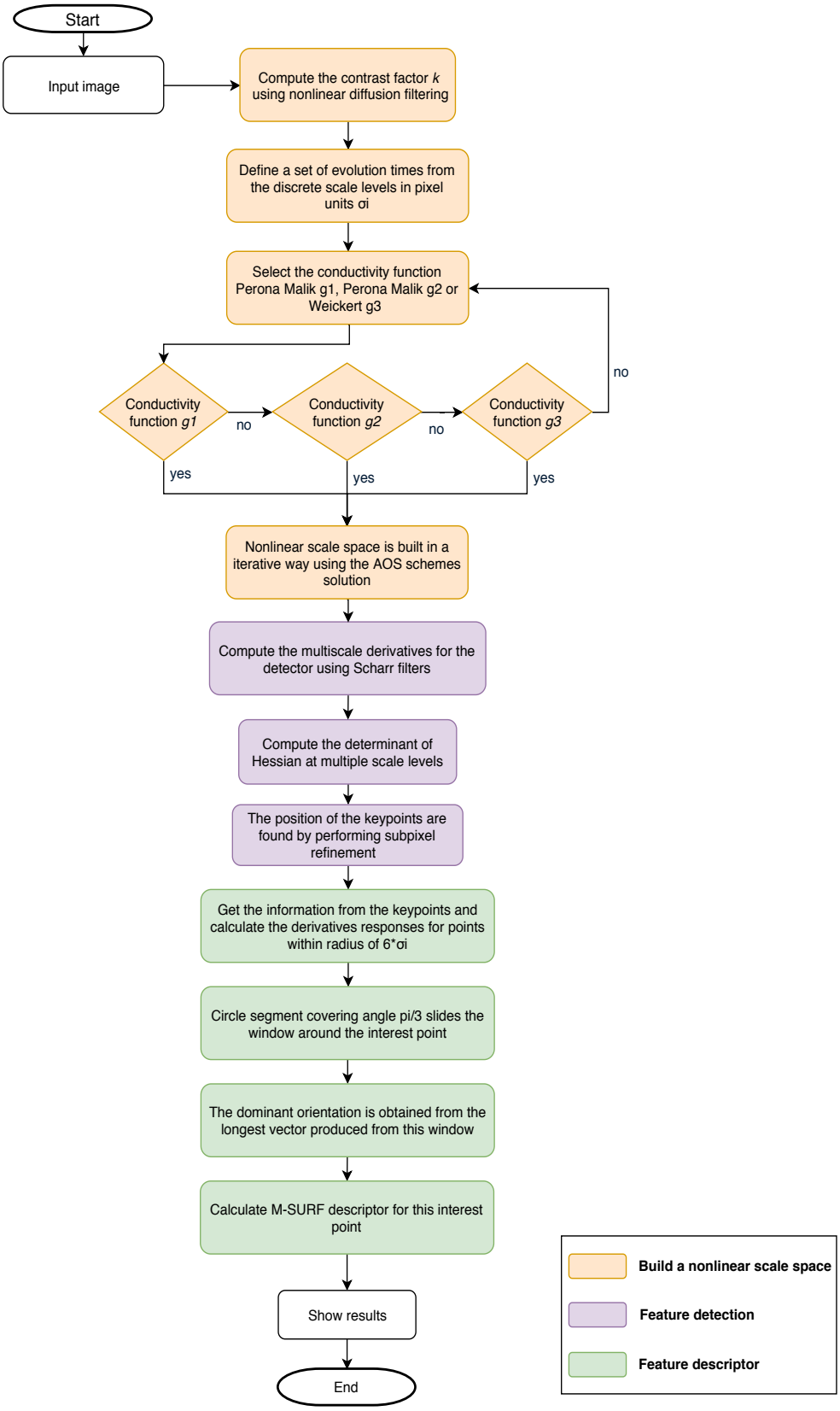


Figure 2.1: Shows step by step flowchart of the KAZE algorithm

2.1.1 Build a Nonlinear Scale Space

The purpose of building a scale space is to differentiate the image structures in the original image, so that good scale image structures only exist at the better scales in multi-scale representations like pyramids, scale-space representations, and nonlinear diffusion methods [21]. The nonlinear scale space built for KAZE feature algorithm was realized as diffusion-type Partial Differential Equations (PDEs), permitting implementation in the Perona and Malik equation [22] [23]. Keep in mind that the scale space in this project makes use of AOS schemes (see appendix B) and conductivity functions from NDF (see appendix A). The nonlinear scale space corresponds to a set of O octaves and S sub-levels, similar to approaches in SIFT and SURF. To be more precise, the scale space is discretized in logarithmic steps of organized series of these octaves and sub-levels in the original image resolution without downsampling every new octave. The discrete values of octave index o and sub-level s are therefore used to identify the sets of octaves and sub-levels respectively [19]. The scale is defined by its standard deviation, which is represented as follows:

$$\sigma_i(o,s) = \sigma_0 2^{o+s/S} \; where \; o \in [0 \dots O-1], \; s \in [0 \dots S-1], \; i \in [0 \dots N] \tag{2.1}$$

Where N represents the total number of filtered images and σ_0 is the base scale level. However, it is necessary to convert the collection of discrete scale levels to time units t_i as NDF is defined in terms of time. The application of this conversion will yield sets of evolution times, enabling transformation of the scale space $\sigma_i(o,s)$ to time units from the mapping $\sigma_i \longrightarrow t_i$ [19]. Note that the purpose of this mapping is to utilize the obtaining sets of evolution times in the development of the nonlinear scale space. The following expression shows the formula for mapping $\sigma_i \longrightarrow t_i$:

$$t_i = \frac{1}{2}\sigma_i^2, \;\; i = \{0 \dots N\} \tag{2.2}$$

In other terms, building a nonlinear scale space utilizing techniques from AOS schemes requires both the contrast parameter k and the set of evolution times t_i [19]. The con-

trast parameter is obtainable by computing the image gradient histogram from an altered base image (see section A.1). This image has been convolved with a Gaussian kernel of standard deviation σ_0 to reduce noise and possible artifacts. After obtaining both contrast parameter k and the sets of evolution times t_i, the nonlinear scale space is built in an iterative way using the AOS schemes solution L^{i+1} (equation (B.2) in appendix B) [19].

Figure 2.2 displays blur with Gaussian scale space and nonlinear scale space respectively over certain evolution times t_i.

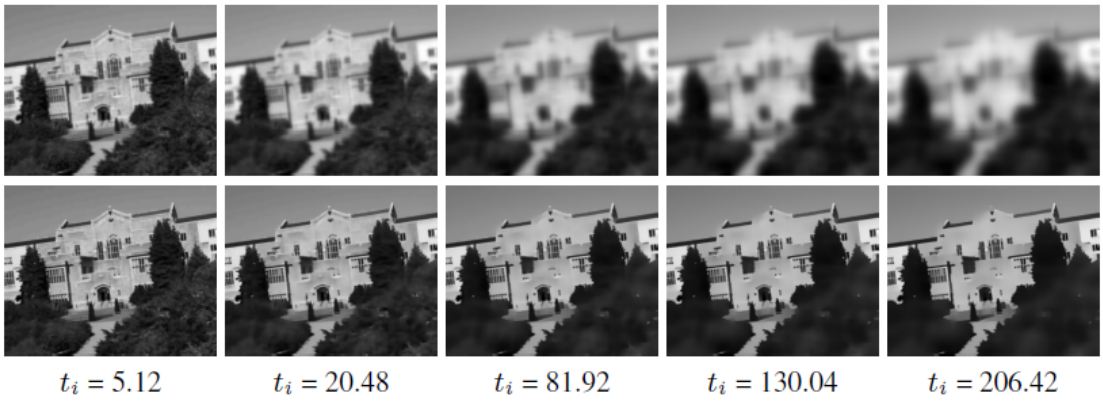


Figure 2.2: Difference between the bluriness Gaussian scale space and nonlinear diffusion scale space for different evolution times t_i [19]. First row: Shows that the image get blurred using the Gaussian scale space (linear diffusion) and Second row: Shows that the image is retaining important object details using the Nonlinear diffusion scale space with conductivity function g_3

2.1.2 Feature Detection of KAZE

The incorporation of a feature detection when developing a KAZE algorithm will permit the identification of image primitives of interest, e.g., points and regions. This algorithm attribute highlights visual cues in any given image and allows the user to extract stable features effectively. Note that the detection accuracy of interest points can be increased by computing the response of scale-normalized determinant of the Hessian (DoH) in the nonlinear scale space at multiple scale levels (Figure 2.4), subsequently enhancing the algorithm identification capability to multi-scale feature detection.

The Hessian matrix is a square matrix of second-order partial derivatives of a function [24]. Each element of Hessian Matrix using indices i and j for determining the position are described in equation (2.3).

$$\mathbf{H}_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j} \tag{2.3}$$

The determinant of Hessian (DoH) is used to find a point of interest in an image where the determinant is maximum. In KAZE feature detection, DoH computes for each of the filtered images L_i in the nonlinear scale space. The matrix below (figure 2.3) represents the Hessian matrix $\mathcal{H}(\mathbf{x}, \sigma)$, where $\mathbf{x}$ = (x,y) is the given point and σ are chosen as the scale in given image I [17].

$$\mathcal{H}(\mathbf{x},\, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x},\, \sigma) & L_{xy}(\mathbf{x},\, \sigma) \\ L_{xy}(\mathbf{x},\, \sigma) & L_{yy}(\mathbf{x},\, \sigma) \end{bmatrix}$$

Figure 2.3: The Hessian matrix $\mathcal{H}(\mathbf{x}, \sigma)$ in point $\mathbf{x}$ with scale σ [17]

where $L_{xx}(\mathbf{x}, \sigma)$ represents the convolution between image I in point $\mathbf{x}$ and the Gaussian second order derivative (Gaussian kernels) is given as $\frac{\partial^2}{\partial x^2} g(\sigma)$. Similarly for $L_{xy}(x, \sigma)$ and $L_{yy}(x, \sigma)$. With regard to this matrix, the determinant of the Hessian used for the KAZE detector can be expressed as follows:

$$L_{Hessian} = \sigma^2 (L_{xx} L_{yy} - L_{xy}^2) \tag{2.4}$$

Where the second order horizontal and vertical derivatives are given as L_{xx} and L_{yy} respectively, and L_{xy} is the second order cross derivative. The set of first and second order derivatives are estimated by 3×3 Scharr filters (See section 2.2) of different scale levels σ_i. The Scharr filters have a greater rotation invariance and capability to differentiate central differences compared to popular filters such as Sobel filters and standard central differences differentiation [20] [25]. Consecutive Sharr Filters in desirable derivative coordinates enable approximation of second order derivatives. Furthermore, it is recommended to analyze the detector response at the previously mentioned different scale levels σ_i provided the set of filtered images from the nonlinear scale space

L_i has been calculated in advance.

First and foremost, the detector response analysis require a search of maxima in scale and spatial location before the search for each extrema in all filtered images except $i = 0$ and $i = N$ over a rectangular window of size σ_i x σ_i on the current i, upper $i + 1$ and lower $i - 1$. The size of the search window is set to 3×3 pixels to efficiently search and discard non-maxima responses [19] within the set of values from the analysis. The descriptor step in the algorithm therefore saves computational efforts by utilizing the same set of aforementioned calculated derivatives if incorporated with this Hessian-based feature detection, even when the necessary computations of multi-scale derivatives for every pixel are taken into account [19]. Finally, the position of each interest point is calculated with sub-pixel refinement by appropriate a quadratic function to the determinant of the Hessian in a 3×3 pixels neighborhood and finding its maximum [22].

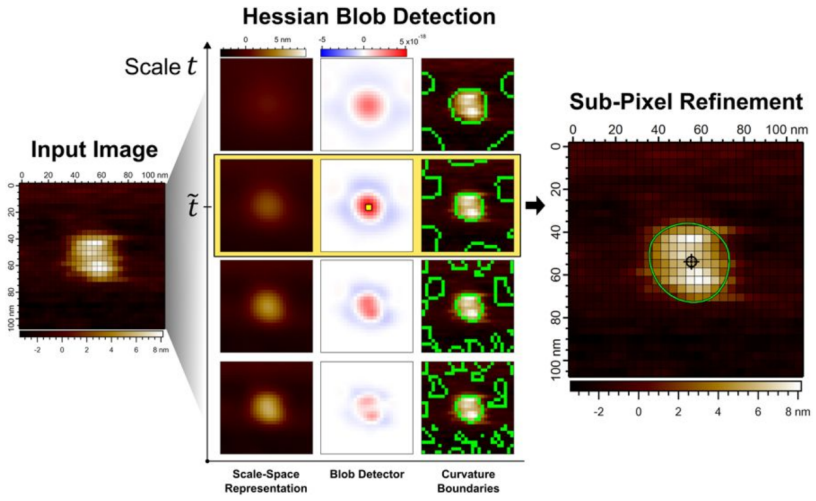


Figure 2.4: Shows how the Hessian blob detection algorithm is detecting blob features of the input image using Gaussian scale space. KAZE uses similar approach only difference is the use of nonlinear scale space instead of Gaussian scale space, where nonlinear scale space is keeping the image details instead of blurring the image as done here with the Gaussian scale space. Note that, the pixels for the input image here are given in the unit *nm*

2.1.3 Feature descriptor of KAZE

The last step of the KAZE feature algorithm is to develop a feature descriptor which differentiates one feature in an image from another. An ideal descriptor is invariant to image transformations. Simply put, a feature descriptor that distinguishes key characteristics in an image through the calculative process of feature description, a process which assimilates interest points from an image and yields feature descriptors [20]. Note that a KAZE feature descriptor is the Modified-SURF (M-SURF) descriptor adapted to the nonlinear scale space framework mentioned in section 2.1.1 [19].

Bear in mind that it is necessary to compute the dominant orientation for all interest points before building the descriptor. If the objective is to build a rotation invariant descriptor, the dominant orientation for each detected interest point should be calculated. The dominant orientation can be estimated with a sampling step of size σ_i in a circular area of radius $6\sigma_i$. Each sample within this circular area includes first order derivatives L_x and L_y where the derivative responses are represented as points. The dominant orientation is then defined from the longest vector from calculating the sum of all derivative responses as points in vector space within the sliding aforementioned orientation circle covering an angle of $\pi/3$ [19] [20].

With the dominant orientation estimated, the next step is to construct the descriptor. KAZE descriptor utilize the structure of the Modifed-SURF interest point descriptor embedded to the framework of the nonlinear scale space stated in section 2.1.1. The first order derivatives L_x and L_y of size σ_i are calculated over a $24\sigma_i$ x $24\sigma_i$ rectangular grid for a detected feature at scale σ_i. This grid is then divided into 4×4 subregions of size $9\sigma_i$ x $9\sigma_i$ with an overlap of $1.5\sigma_i$ to preserve important information. In each subregions, the derivative responses is Gaussian weighted using ($\sigma_1 = 2.5\sigma_i$) centered in the subregion center and summed into a descriptor vector $d_v = (\sum L_x, \sum L_y, \sum |L_x|, \sum |L_y|)$ [19]. Each subregion vector is thereafter Gaussian weighted using ($\sigma_2 = 1.5\sigma_i$) defined over a mask of size 4×4 with an interest point as center. The derivatives are computed in consideration of the dominant orientation of interest points rotated accordingly for both samples in the rectangular grid. Finally, to achieve invariance to contrast, the de-

scriptor vector of length 64 is converted into a unit vector [19] [20]. Figure 2.5 shows the rectangular grid and subregions in a Modified Surf descriptor building process.

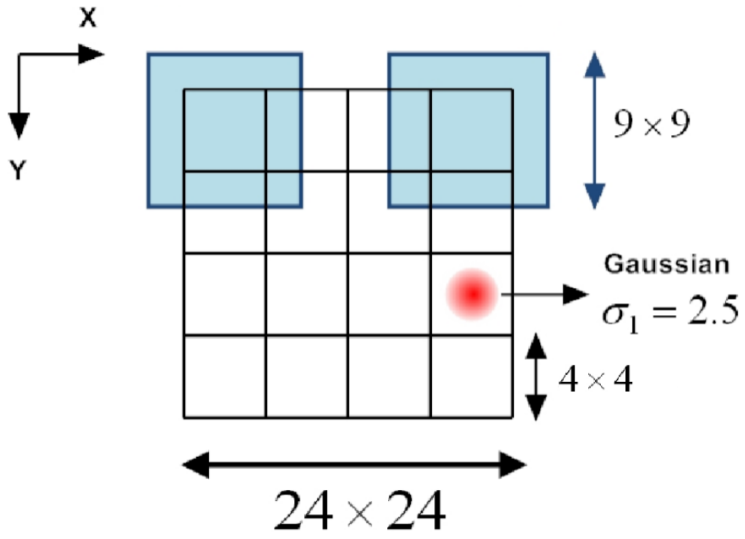


Figure 2.5: Modified Surf Descriptor building process [26]

2.2 Scharr filter

The approximation used for Sobel operator lack accuracy when it comes to small kernels. Specifically, for a 3-by-3 Sobel filter, the inaccuracies are more frequent when the gradient angle is farther away from horizontal or vertical. For that reason, a filtering method called Scharr filter can be used. This is a filtering method used to identify gradient features along the x-axis (horizontally) and y-axis (vertically) using the first derivatives. Besides, the Scharr filter is also more precise, better in terms of rotation invariance and equally fast as the Sobel filter [27] [25]. Figure 2.6 shows the operator for both horizontal and vertical direction.

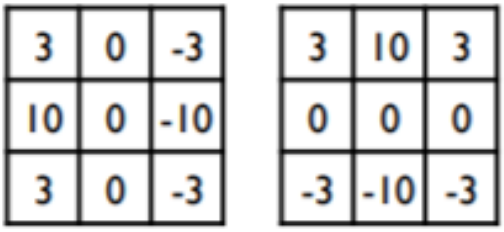


Figure 2.6: The 3-by-3 Scharr filter for both horizontal and vertical direction. horizontal kernel on right and vertical kernel on left.

2.3 SURF - Speeded Up Robust Features

SURF - Speeded Up Robust Features is another scale- and rotation invariant detector and descriptor. SURF was developed in 2008 by Herbert Bay, Tinne Tuytelaars and Luc Van Gool. As mentioned earlier, the SURF detector uses Hessian determinants to detect interest points while the descriptor uses Haar-wavelets response. The following subsections give a detailed description of the detector and the descriptor.

2.3.1 SURF detector

Similar to KAZE, the SURF detector [17] also uses an approximation of the determinant of the Hessian matrix to find interest points as well. This is due to its great performance in terms of computation time and accuracy (see section 2.1.2 for the explanation of the Hessian).

The second order partial derivatives must be discretized and cropped before applying 9x9 box filters. Top right and top left of Figure 2.7 shows the Gaussian second order partial derivative in y-direction, L_{yy} and xy-direction, L_{xy}. The bottom right and bottom left of Figure 2.7 shows how the SURF algorithm approximates these Gaussian second partial order derivatives with σ = 1.2 by using box filters of size 9x9 pixels. The Illustrated gray area corresponds to value 0, and the black area is -1, and the white area is 1 [17].

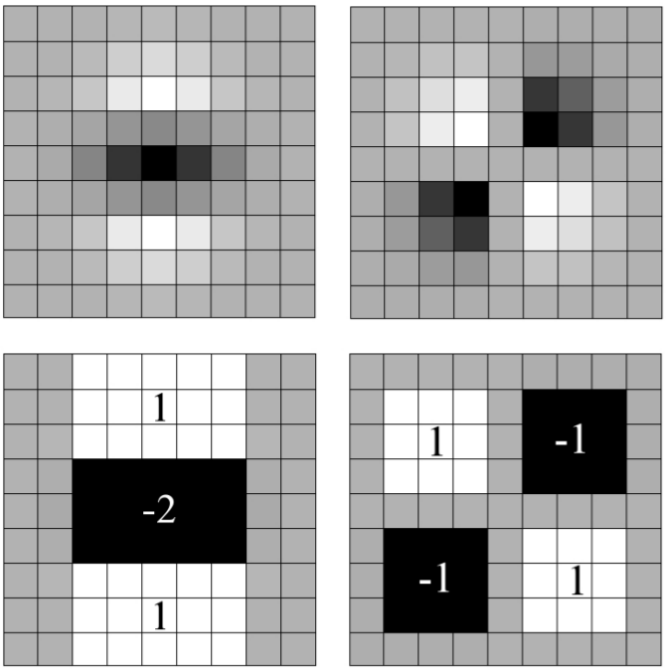


Figure 2.7: Top left: Gaussian second order partial derivative in y-direction, L_{yy} [17]. Top right: Gaussian second order partial derivative in xy-direction, L_{xy}. Bottom left: 9x9 Boxfilter in y-direction (D_{yy}). Bottom right: 9x9 boxfilter in xy-direction (D_{xy}).

The computational time of the approximated Gaussian derivatives is reduced by using integral images or *Summed Area Table*. An integral image represents the sum of all gray level pixels in a rectangular area where each point $\mathbf{x} = (x, y)^T$ stores the sum of all pixels between origo and $\mathbf{x}$ in the input image I [17]. The formula for the sum of all possible rectangle is given as follow.

$$I_{\sum}(\mathbf{x}) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} i(i, j) \tag{2.5}$$

This method calculates the average intensity within the given image, allowing us to be more effective than first computing the darker and lighter pixels in different regions and then calculating the sum of these pixels [28].

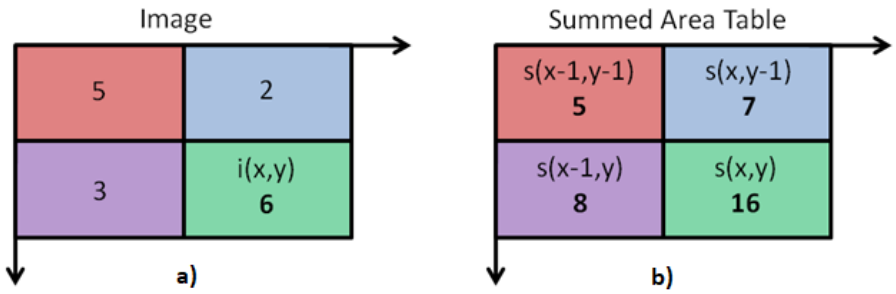


Figure 2.8: Input Image vs Integral Image [29]

Figure 2.8 illustrates 4 array references in the input image. The formula can be written as following;

$$s(x,y) = i(x,y) + s(x-1,y) + s(x,y-1) - s(x-1,y-1) \tag{2.6}$$

Note that, in the summed area table, the value at any point (x,y) is the sum of all the pixel values above and to the left of the interest area. For example for i(x,y) in figure 2.8, the summed area table are calculated as follow by using formula 2.6.

$$s(x,y) = 6+3+2+5 = 16 \tag{2.7a}$$

$$s(x-1,y) = 3+5 = 8 \tag{2.7b}$$

$$s(x,y-1) = 2+5 = 7 \tag{2.7c}$$

$$s(x-1,y-1) = 5 \tag{2.7d}$$

The integral image from Figure 2.8 b) is calculated using the values from input image Figure 2.8 a). Once the integral image is computed, it needs three additions to compute the average intensity of the rectangular region. Figure 2.9 illustrates how the total

intensity is calculated using 3 additions [17].

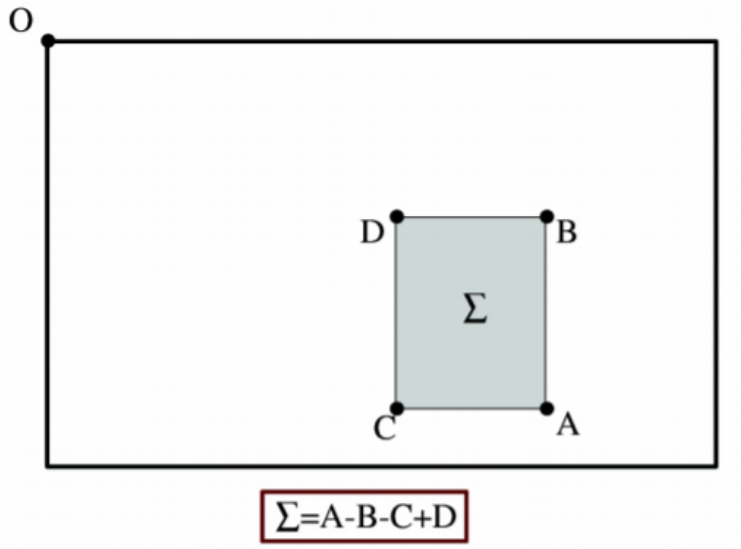


Figure 2.9: Calculating the total intensity using 3 additions [17]

Thus, the calculation time is independent of the rectangular region. This is important due to SURF is utilizing big filter sizes.

The approximated Gaussian second partial derivatives (Gaussian kernel) is denoted as D_{xx}, D_{xy} and D_{yy}. Also, weights are applied to the rectangular regions to make the calculations more efficient. The determinant of the Hessian with this weight is approximated as:

$$det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{2.8}$$

Note that this equation is similar to the one used for KAZE detection (Equation 2.4). The only differences are that KAZE analyzes the detector response at different scale levels by using σ while SURF adds the weight w to the rectangular regions to enhance the computation.

The reason for using the weight w of the filter response is to balance the expression for the Hessian determinant. Additionally, this is also required for the energy conser-

vation between Gaussian second order partial derivatives (Gaussian kernels) and the approximated Gaussian second order partial derivatives (Gaussian kernels) [17]. The equation for the weight w is as following:

$$w = \frac{\left|L_{xy}(1.2)\right|_F \left|D_{yy}(9)\right|_F}{\left|L_{xx}(1.2)\right|_F \left|D_{xy}(9)\right|_F} = 0.912... \simeq 0.9 \tag{2.9}$$

Where $|\sim|_F$ is the Frobenius norm, which is a matrix norm, σ is equal to 1.2, and 9x9 is the area utilized. Theoretically the weight w changes according to the scale. In SURF algorithm this weight is kept as 0.9 [17].

Usually, many feature detection algorithms use the standard method, which consists of implementing the scale using an image pyramid. SURF, on the other hand, make use of box filters and integral images. Instead of iteratively applying the same filter at the output image from the previous filtering, box filters of any size can be directly applied to the original image. By doing this, the image doesn't lose any resolution during the process, and the filter size is upscaled [17]. Figure 2.10 illustrates both downscaling and upscaling of image and filter size, respectively.

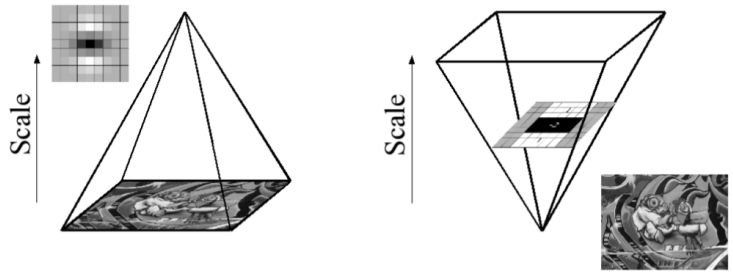


Figure 2.10: Right: Other feature detection down scales the image. Left: SURF is up-scaling the filter size and image keeps the same size [17]

The first scale layer considers the output of the 9x9 filter as its initial scale layer, which is referred to as s = 1.2. The subsequent layers are made by gradually increasing the filter size over an image. The reason for this type of sampling is due to its efficiency in terms of computation. This results in filters of size 9x9, 15x15, 21x21, 27x27, etc.

The scale space in an image pyramid is divided into octaves. The SURF algorithm defines an octave in scale space as a series of filter response maps achieved by convolving a filter with increasing size with input image [17]. Figure 2.11 shows how the filter is upscaled from 9x9 to 15x15.

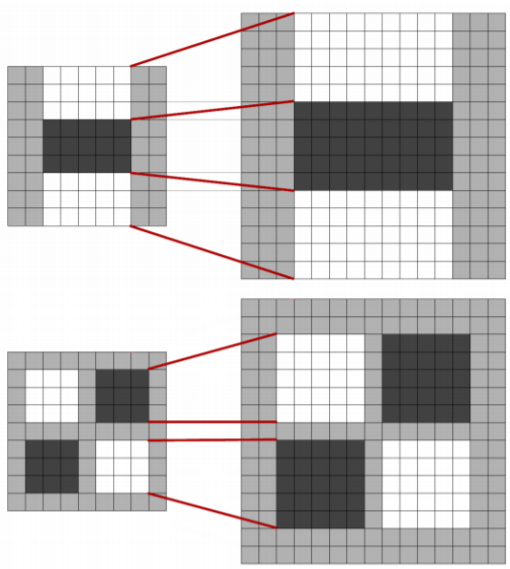


Figure 2.11: Filter D_{yy} (top) and D_{xy} (bottom) are up-scaled from filter size 9x9 to 15x15 [17]

To ensure the appearance of the central pixel and keep the size of the filter irregular, the filter needs to be increased by a minimum of 2 pixels (one pixel on each side). Thus, the filter increases its size with 6 pixels in the first octave. This means the first octave will be represented by filters with sizes 9x9, 15x15, 21x21, and 27x27. For each new octaves, the filter increases its size with a factor of 2, which means the filter size of the second octave is increased from 6 to 12 and third octave from 12 to 24. An octave can be added if the image size is larger than the filter size of the previous octave. However, an increase in the number of octaves causes a decrease in the detection of interest points in each octave. Note that the interest points are detected at the lowest filter size at given octave [17]. Figure 2.12 illustrates the overlay between two given octaves and the increase of the range in filter sizes for each octave.

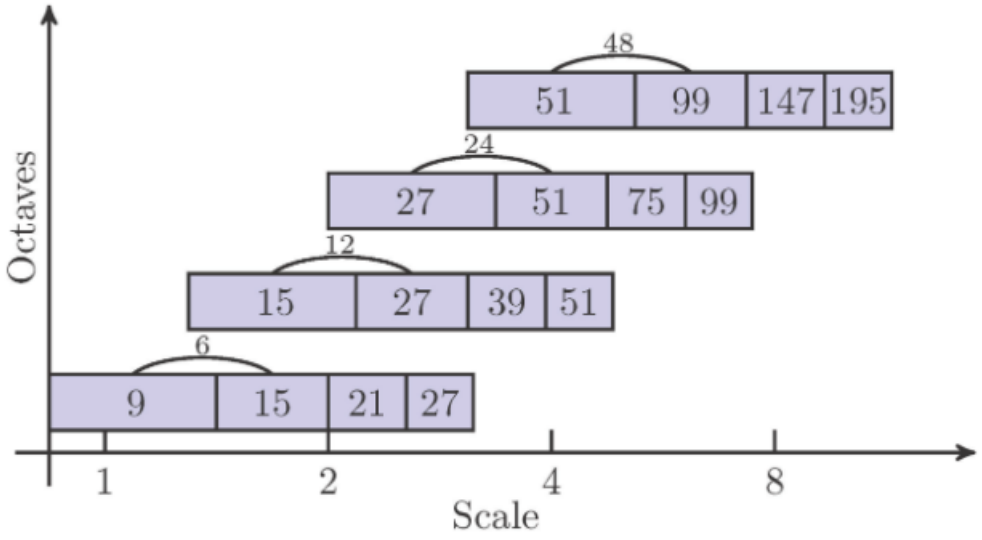


Figure 2.12: Depicts how various filters are increasing at different octave steps [30]

A fast variant of Non-Maxima Suppression (NMS) introduced by Neubeck and Van Gool is applied to localize interest points in the image [31]. After that, the maximum value of the determinant of the Hessian matrix is interpolated in scale and image space with a method developed by Brown and Lowe [32].

2.3.2 SURF descriptor

The SURF descriptor approach consists of finding the orientation assignment and extract the SURF descriptor from a square region. To make the descriptor invariant to rotation, SURF utilizes Haar Wavelets responses in x and y-direction within a circular area of radius $6s$, where s is the scale at which the interest point is detected. By adjusting the size of the wavelets to a side length of $4s$, integral images can be used for fast computation of the filter responses. This implies that only six operations are required to calculate wavelets response [17]. Figure 2.13 shows the Haar wavelets filters in x and y-direction.

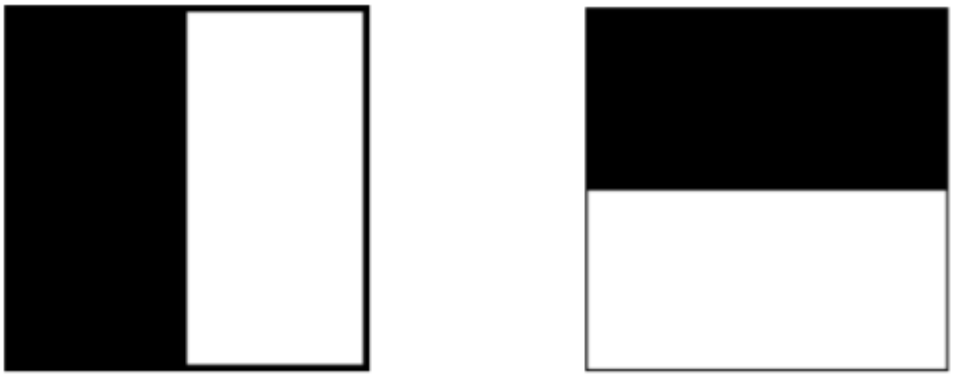


Figure 2.13: Depicts Haar Wavelets filter in x-direction (left) and y-direction (right). The weight on the dark parts are -1 and white part +1 [17]

After the wavelet's responses are calculated and weighted with Gaussian ($\sigma = 2s$) centered at the interest point, the responses are represented as points alongside the horizontal and vertical axis. Afterward, the dominant orientation is estimated by summing all responses within a sliding orientation window of size $\pi/3$ [17]. Figure 2.14 illustrates this. For applications where rotation invariance is not necessary, an alternative method called Upright version of SURF (U-SURF) can be used. It is robust to rotation about $\pm 15^{\circ}$ and increased computational speed.

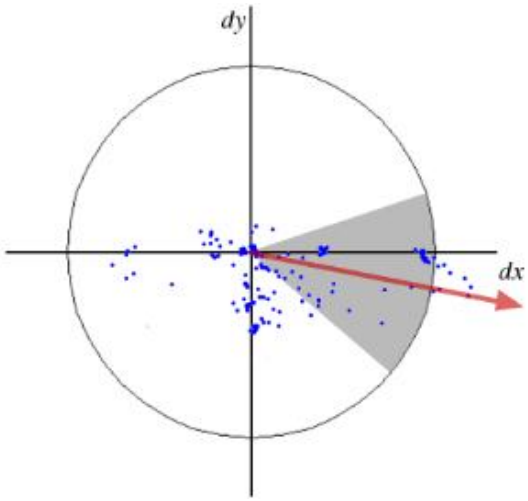


Figure 2.14: Orientation assignment: Blue points represent the Haar wavelet responses, gray region is the window of size $\pi/3$ and red arrow is the longest vector the dominant orientation is obtained [17]

The extraction of the descriptor consists of a square region area with a window size $20s$ constructed around the interest point alongside the orientation described in the previous section. To preserve important information, the region is divided into smaller square sub-regions with size 4 x 4. After that, the Haar wavelet responses are computed in horizontal direction d_x and vertical direction d_y respectively. The responses d_x and d_y are weighted with a Gaussian (σ = 3.3s) centered at the interest point in order to increase the robustness of the geometric deformations and the localization errors [17].

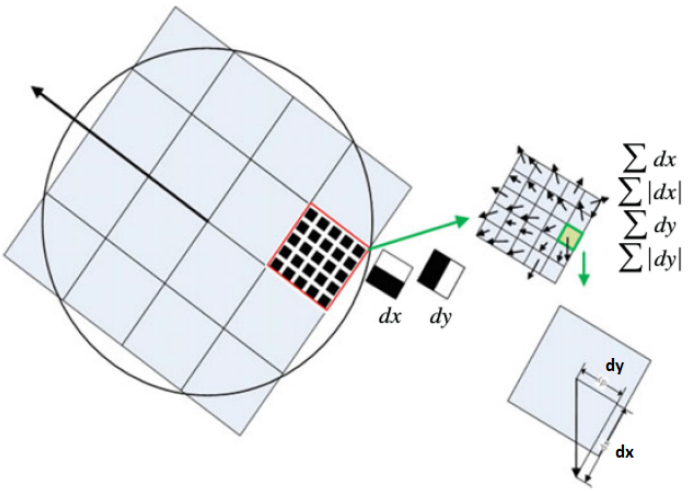


Figure 2.15: Shows how the descriptor is build [1]

The Haar wavelet responses d_x and d_y in each sub regions are summed and a feature vector is formed from this. In addition the sum of absolute values of the responses $|d_x|$ and $|d_y|$ computed in order to include the polarity of the intensity changes. Consequently, each sub-regions will include a four dimensional feature vector $\mathbf{v} = [\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|]$. Thus, the interest point can be described with a descriptor vector of length 64. Contrast invariance is achieved by turning the descriptor into a unit vector.

2.4 SUSAN

SUSAN (Small Univalue Segment Assimilating Nucleus) feature detector was developed in 1995 by S.M. Smith and J.M Brady [8]. SUSAN is known for its capability of detecting both edge and corner. In addition, the SUSAN detector has several advantages such as no derivatives are used, high repeatability and invariant to translation and rotation. The drawbacks of this detector are that it is not invariant to scaling and using a fixed global threshold value is not appropriate [1]. The following subsections will give a more detailed explanation of this detector.

2.4.1 SUSAN feature detector

The SUSAN principle for feature detection is as follows. SUSAN detector applies a circular mask with a given radius at each pixel in an image. The usual radius value for such circular mask is 3.4 (giving a mask of 37 pixels) [8].

For each pixel, the difference between the brightness of each pixel within the circular mask is compared with the circular mask center also called nucleus, to determine if they have the same or different intensity values. The area of the mask which has similar intensity values as the nucleus is known as USAN (Univalue Segment Assimilating Nucleus) [8]. The equation (2.10) shows the comparison function $c(\vec{r}, \vec{r_0})$, which verifies if a pixel is within USAN or not.

$$c(\vec{r}, \vec{r_0}) = \begin{cases} 1, & if\ |I(\vec{r}) - I(\vec{r_0})| \leq t \\ 0, & if\ |I(\vec{r}) - I(\vec{r_0})| > t \end{cases} \tag{2.10}$$

Where $\vec{r_0}$ is the position of the nucleus and $\vec{r}$ is the position of any other point within the circular mask. The intensity value of any given point and nucleus are given as $I(\vec{r})$ and $I(\vec{r_0})$ respectively. The equation (2.11) illustrates how the USAN area is calculated from the circular mask.

$$n(\vec{r_0}) = \sum_{\vec{r}} c(\vec{r}, \vec{r_0}) \tag{2.11}$$

Where the value n represents the number of pixels within the circular mask that are verified as USAN, from which the USAN area can be retrieved from.

The value t is the threshold that is used to determine the difference in intensity value between the nucleus and any other pixel within the circular mask. If this difference is higher than the threshold value t, then the tested pixel isn't part of the USAN. Figure 2.16 shows how the pixels used for USAN calculations are extracted.

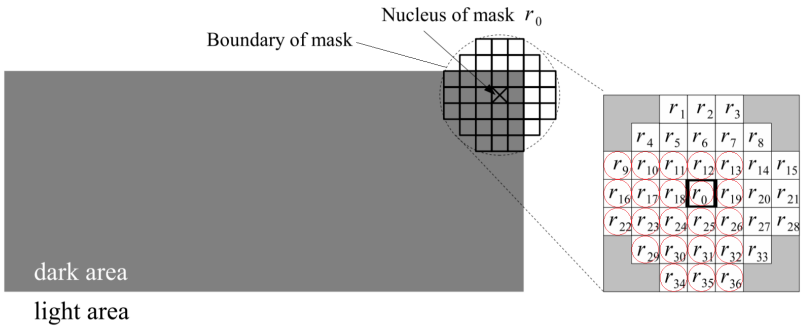


Figure 2.16: A circular mask placed on a figure (left) and the pixels verified as USAN are marked with red circles (right). In addition the nucleus of mask is labelled as r_0 [33]

Even though the equation (2.10) gives good results, it is not stable in practice. For that reason, an improved version of the comparison function can be used.

$$c(\vec{r},\vec{r_0}) = e^{-(\frac{I(\vec{r})-I(\vec{r_0})}{t})^6} \tag{2.12}$$

This change allows to slightly vary the pixel's brightness without affecting the comparison function $c(\vec{r},\vec{r_0})$, even if it's near the threshold value t. Also, using this equation gives an optimal balance of improvement and stability [8]. This corresponds to fulfill the criterion of a minimum number of false negatives and false positives, which is expressed as follows:

$$F(d,t,s) = \frac{\sqrt{var(R_S)} + \sqrt{var(R_N)}}{< R_S > - < R_N >} \tag{2.13}$$

Where F is proportional to the number of false positives and false negatives, s is the

standard deviation of the image noise, R_S is the SUSAN edge response strength when the mask is centered on an edge with a strength d and R_N represents the SUSAN edge response strength with no edge present. The value F is dependent on the values d, t and s. A more detailed explanation of these variables are given in the article *SUSAN - A New Approach to Low-Level Image Processing* [8].

Figure 2.17 clearly shows that for the best possible optimization of the SUSAN filters is by setting the exponent factor in the brightness comparison to $J = 6$, which is also done in equation (2.12). This is due to the lowest number of false negatives, and false positives are settled around this value $J = 6$.

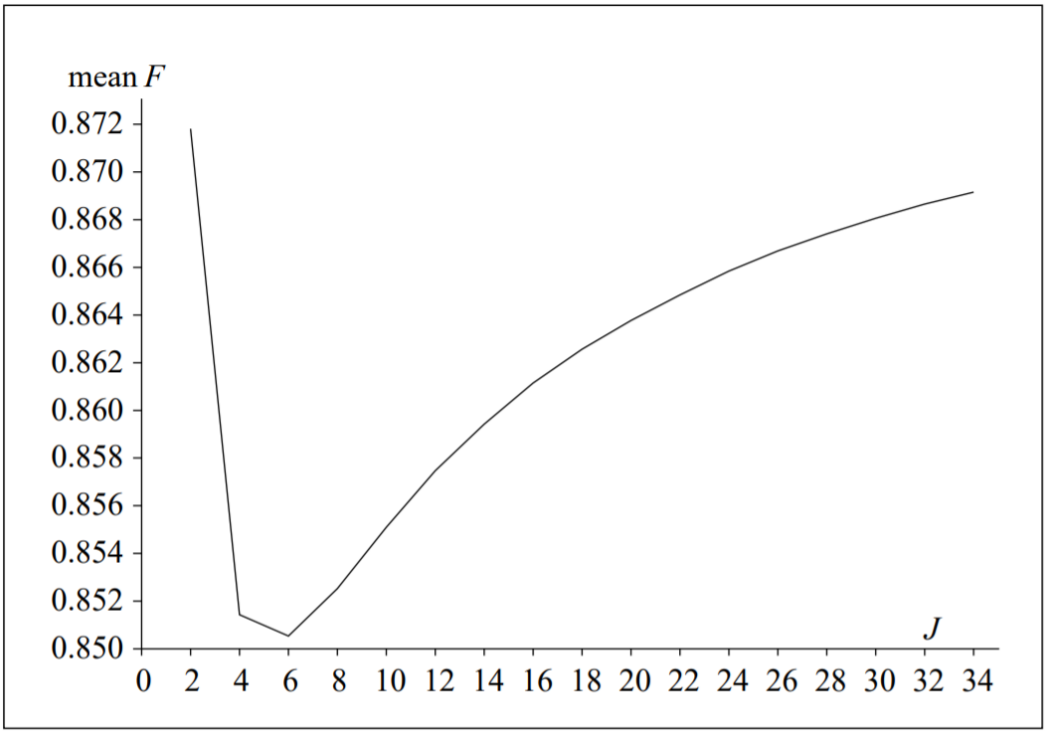


Figure 2.17: illustrates the resulting plot of F against J, which uses the mean results F and J factor [8]

Figure 2.18 shows the difference between between **a**, representing equation (2.10), and **b** which represents the equation (2.12). The purpose of Figure 2.18 is to show how large the threshold value has to be for it to be verified as USAN. For equation (2.10) represented as graph **a**, the threshold value t has to be somewhere between ± 27 for the

pixel to be verified as USAN. Equation (2.12) represented as **b** shows a more smoother version comparing to **a**, which gives a more stable result. The graph **c** illustrates the boundary detector B, which is just a method of selecting the narrow area that lies on the boundary between the USAN and non-USAN regions [8].

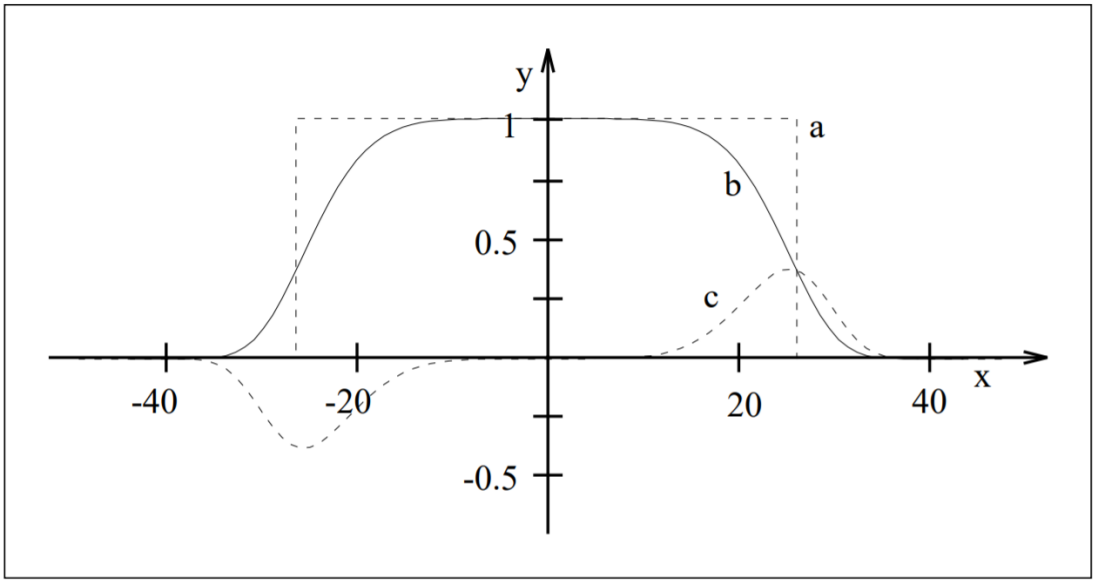


Figure 2.18: shows the comparison between the equation (2.10) (shown as **a**) and the equation (2.12) (shown as **b**). The y-axis represents the original comparison function with no units, the x-axis is the grey level value for the pixel brightness and the threshold value t is set at ± 27 grey levels [8]

Note that to distinguish edges and corners, a geometric threshold g is fixed to a specific value and compared with equation (2.11). This threshold value decides whether a corner or an edge is detected. Following subsections gives a more detailed description.

2.4.2 SUSAN edge detector

In order to detect edges, the geometric threshold g is set to $\frac{3n_{max}}{4}$ where n_{max} represents the maximum number of pixels in the circular mask. Furthermore, this geometric threshold of g is compared with n only if the USAN area is smaller than the geometric

threshold [8]. The following equation shows how the edge response can be obtained:

$$R(\vec{r_0}) = \begin{cases} g - n(\vec{r_0}), & if\ n(\vec{r_0}) < g \\ 0, & otherwise \end{cases} \tag{2.14}$$

where $R(\vec{r_0})$ represents the initial edge response. Besides, this formula is a straightforward formulation where the smaller the USAN area is, the greater is the edge response [8].

After the initial edge response is calculated, the edge direction needs to be computed. This is due to different reasons like computational efficiency using methods such as NMS, which requires the edge direction. The edge direction is related to a point in an image which has non zero edge strength. Moreover, the edge response can be found by analyzing the USAN area, which results in either the inter-pixel edge case or intra-pixel edge case depending on the edge type examined [8].

Inter-pixel edge case is used if the USAN area (in pixels) is larger than the mask diameter (in pixels) and the center of gravity of USAN and the nucleus is perpendicular to the local edge direction [8]. Both the centre of gravity and the edge direction for this type of edge point (see Figure 2.19 **a** and **b**) are found by using the following formula:

$$\bar{\vec{r}}(\vec{r_0}) = \frac{\sum_{\vec{r}} \vec{r} c(\vec{r}, \vec{r_0})}{\sum_{\vec{r}} c(\vec{r}, \vec{r_0})} \tag{2.15}$$

Intra-pixel edge case is used if the USAN area (in pixels) is smaller than the mask diameter (in pixels) or else if the center of gravity of USAN lies less than one pixel away from the nucleus. The edge direction for this kind of point (see Figure 2.19 **c**) is estimated from the following sums:

$$\overline{(x - x_0)^2}(\vec{r_0}) = \sum_r (x - x_0)^2 c(\vec{r}, \vec{r_0}) \tag{2.16}$$

$$\overline{(y - y_0)^2}(\vec{r_0}) = \sum_r (y - y_0)^2 c(\vec{r}, \vec{r_0}) \tag{2.17}$$

$$\overline{(x-x_0)(y-y_0)}(\vec{r_0}) = \sum_r (x-x_0)(y-y_0)c(\vec{r},\vec{r_0}) \tag{2.18}$$

The orientation of the edge is determined by using the following ratio $\frac{\overline{(y-y_0)^2}}{\overline{(x-x_0)^2}}$ and whether a diagonal edge has positive and negative gradient is determined by the sign of $\overline{(x-x_0)(y-y_0)}$.

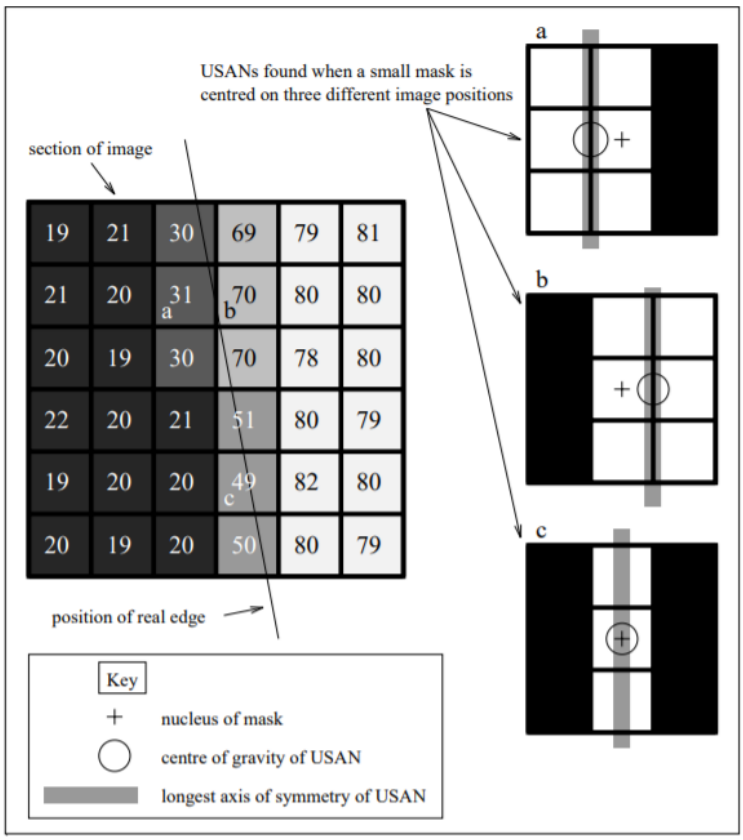


Figure 2.19: The USAN area with 3x3 mask on three different image positions. Point **a** and **b** are edge points lying on each side of the edge, while the point **c** lies on a thin gray band that represent a mixture of two regions [8]

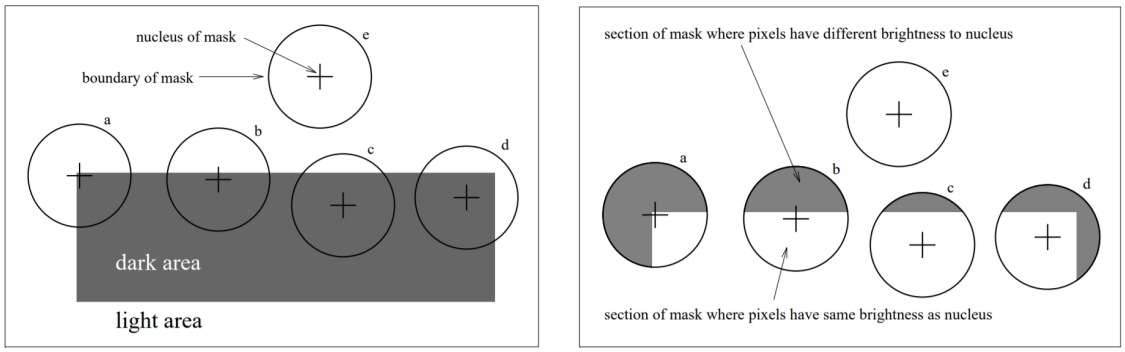
Ultimately, the NMS is applied on the edge responses so that the non-maxima points perpendicular to the edge are avoided being defined as edge points. Besides by using a thinning process called "binary thinned", the incorrectly removed edge points by NMS can be replaced. For more details, see article *SUSAN - A New Approach to Low-Level Image Processing* [8].

2.4.3 SUSAN corner detector

There are some similarities between SUSAN edge and corner detection. The only difference is the geometric threshold g, which is set to $\frac{n_{max}}{2}$. Thus, the formula for corner detection can be written as the following:

$$R(\vec{r_0}) = \begin{cases} \frac{n_{max}}{2} - n(\vec{r_0}), & if\ n(\vec{r_0}) < \frac{n_{max}}{2} \\ 0, & otherwise \end{cases} \tag{2.19}$$

Figure 2.20 illustrates how USAN is extracted. The dark rectangle represents an object within an image. Five circular masks a, b, c, d, and e are applied at different positions on the image. The circular mask with the smallest USAN has a corner, and an edge is detected when USAN is covered more than half but less than $\frac{3n_{max}}{4}$. This means that the circular mask (**a**) has a corner, while a circular mask (**b**) has an edge.



(a) Four circular masks at different places on the image

(b) USAN shown as the white parts of the mask are extracted from the four circular masks

Figure 2.20: SUSAN feature detection **(a)** using circular masks at different places on the dark area and USAN **(b)** shown in white, which is extracted from the dark area [8]

To reduce false positive corner responses caused by noise or an edge, two procedures have been developed [34].

1. The first method is to find the center of gravity of USAN (see equation (2.15)). Then find the length from the nucleus to the center of gravity. If the center of gravity is far away from the nucleus, then the USAN equals to a corner, while short distance corre-

sponds to a thin line passing through the nucleus. Thus, false positives are discarded.

2. The second method forces the contiguity in the USAN area. To detect a corner, all of the pixels in the circular mask lies in a straight line pointing towards the direction of the center of gravity of USAN from the nucleus. This reduces false positives and is useful in terms of forcing the USAN to have a degree of uniformity.

The last step consists of using the NMS to find corners.

3. Methods

This chapter presents descriptions of methods to provide solutions to the problem statement. The code use MATLAB, a common choice in development of feature detection and description due to several toolboxes dedicated for work with computer vision in its vast library. It includes functions from built-in toolbox. Images and the experiments regarding these images are described as well. The following sections explain the different functions, test images and experiments in this project.

3.1 Experimental setup and source code

The experiments have been carried out on an Intel Core i5-6300HQ 2.30GHz portable computer with 8GB of RAM. The source code was made in MATLAB with calling functions from a built-in library named *Image Processing Toolbox*. Matlab is a mathematical program which lets you model, compute, and do different calculations. All coding and experiments were done in a 2017b version of Matlab for compatibility reasons and given requirements by the task. The following subsection briefly describes the different code files used in this project.

3.1.1 Description of code based on KAZE features

Made with respect to the KAZE feature detection algorithm, the code detects interest points by using the built-in MATLAB-function called **detectKAZEFeatures**, where additional options such as threshold, conductivity functions (diffusion type), number of octaves (multiscale detection factor) and scale levels within each octave have to be specified with input arguments [35]. Note that diffusion types in the function documentation are based on equations (A.3) and (A.4).

The flowchart in Figure 3.1 shows the steps to take in MATLAB code **KAZEdetect.m** (Appendix C.2). Note that the built-in MATLAB-function **detectKAZEFeatures** is generated from the programming language *C++* written by the author Pablo F. Alcantarilla. The source code can be found in his GitHub account [36].

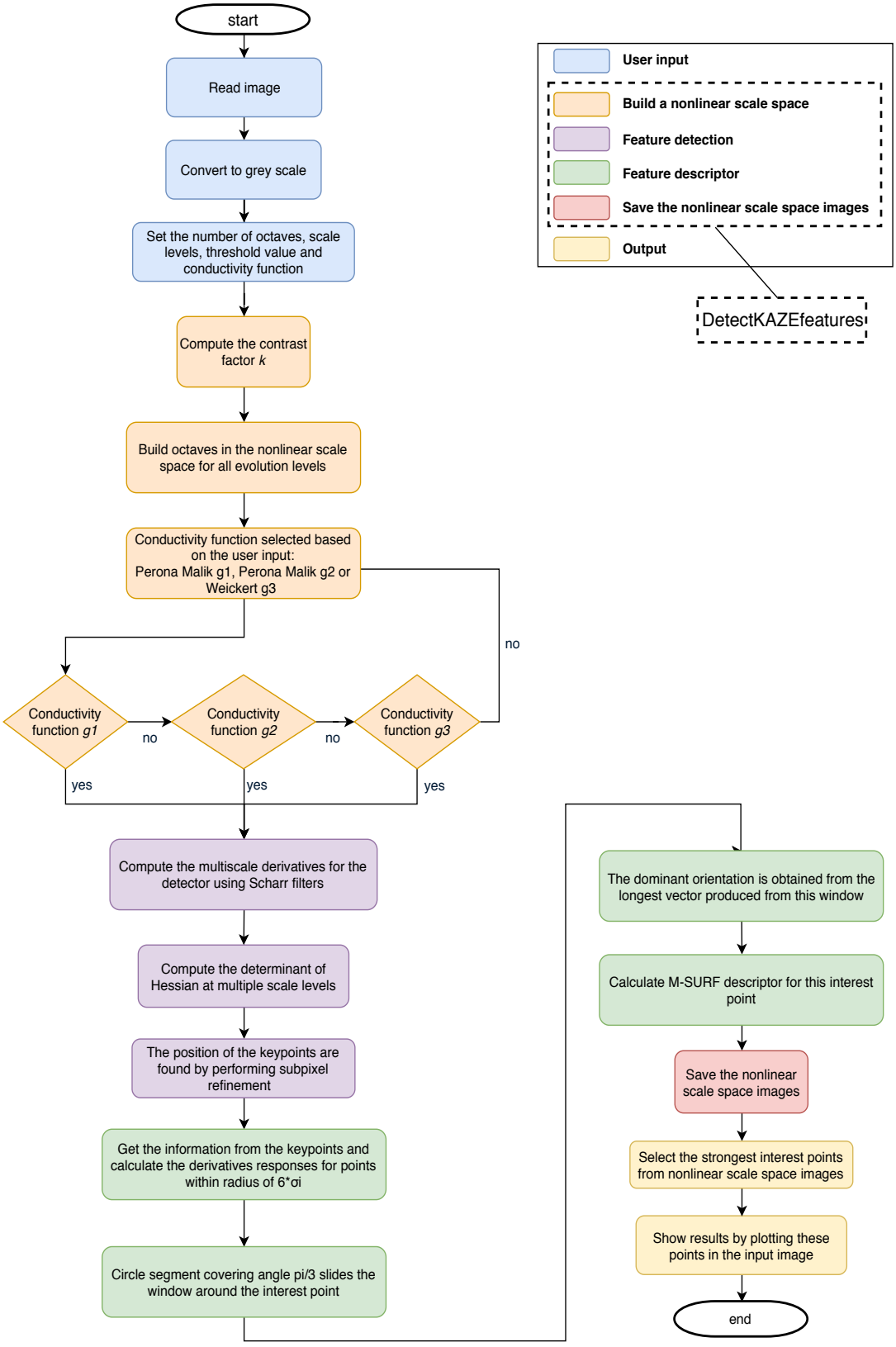


Figure 3.1: Shows step by step flowchart of the MATLAB code **KAZEdetect.m**.

KAZEdetect.m:

This script file converts an input image to grayscale image using the function *rgb2gray*. Furthermore, the user has to include the four aforementioned input arguments required by **detectKAZEFeatures** in order for this code-function to work. The input threshold value is determined by experimenting with several threshold values to determine which value gives the most true interest points, while the number of octaves (O) and scale levels (S) are selected based on the image size and scale changes respectively. Diffusion type in this case is set to ”Region” which means the KAZE algorithm detect features like interest points such as corners or blobs using the conductivity function g_2 in equation (A.3).

After these input arguments are determined and applied to **KAZEdetect.m**, the built-in function **detectKAZEfeatures** will produce results based on the KAZE algorithm. The algorithm steps include building a nonlinear scale space from the input image, interest point detection using Hessian determinant, build an M-SURF descriptor from the dominant orientation, and save the nonlinear scale space images by plotting the interest points found. Finally, the strongest interest points are selected from the saved nonlinear scale space images and shown alongside both time and number of keypoints labels.

Note that conductivity functions g_1 and g_3 are not used here as they are only suitable for edge detection such as sharp-edges and edges respectively. For that reason, the only conductivity function used in this project to detect interest points is g_2.

3.1.2 Description of code based on SURF features

SURF.m:

This script file call upon the built-in MATLAB function called **detectSURFFeatures** in the *Image Processing Toolbox* [17]. Similar to **detectKAZEFeatures**, **detectSURFFeatures** can only return values from a grayscale image. The script file is therefore capable of converting an input image to a grayscale image using the function *rgb2gray*. Furthermore, the metric threshold, number of octaves and scale levels are set to appropriate

numbers in order to achieve most true interest points. The results with the strongest interest points are plotted alongside both time and number of keypoints labels. The code for SURF can be found in Appendix C.3.

3.1.3 Description of code based on SUSAN

SUSAN.m:

This script file is a modified version of the code made by Ke Yen, which was taken from the MATLAB file exchange web page [37]. The code is capable of detecting both edges and corners using two geometric thresholds and two intensity values. These values allow the code to distinguish between an edge or a corner. In this project, both geometric thresholds and intensity values are manually assigned as the default values calculated from $\frac{n_{max}}{2}$ explained in section 2.4.3 did not detect enough corner points. Despite the detectors' main capabilities of detecting both edge and corners, it can be adapted to identify interest points in images. It is therefore suited for this project; the code for the adapted SUSAN detector can be found in Appendix C.4.

3.2 Images used to evaluate the detectors

Several images are selected to be analyzed with KAZE, SURF and SUSAN. The synthetic image with geometrical shapes allow a review of detection capabilities for each detector. Other images will be used to evaluate detectors when detecting interest points in real life images.

3.2.1 Synthetic image with geometrical shapes

Figure 3.2 is an image designed in "Paint 3D", a program for drawing figures and editing images. This synthetic image consists of different geometric shapes, i.e triangle, rectangle, pentagon, lightning, diamond, square and circle. Through the application of detectors on this image easily illustrate the different capabilities of each detector to detect interest points on given geometrical shapes. This image will be referred to as image 1 in the experimental part of the project later.

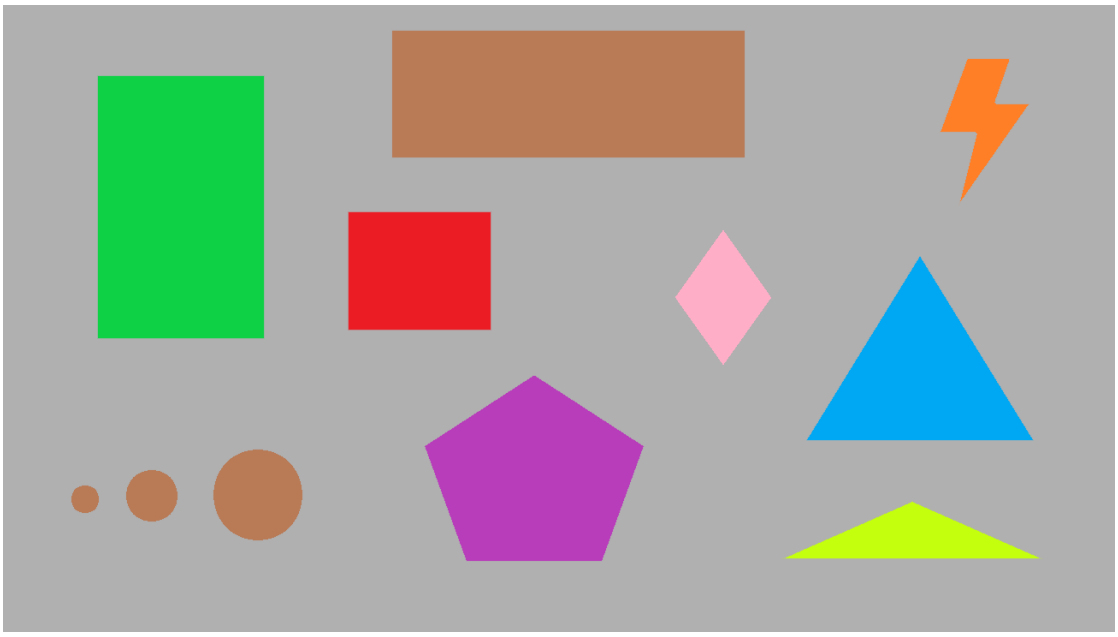


Figure 3.2: An image designed to experiment KAZE, SUSAN and SURF

3.2.2 Real life images taken at different locations

The following images are taken from a test dataset called ”2015 Test images”. This dataset is from a database called COCO, an extensive database that contains numerous datasets for various purposes like object detection, segmentation and captioning [38]. Additionally, an experiment with this dataset allow the assessment of each detectors' detection capability on real life images.

There are a total of two real life images from the dataset used in this experiment. The first image displays a tower, and the second image represents a boy that throws a baseball. In the experimental part of the project, the images will be labelled as image 2 and 3, respectively.

(a) Image of a clock tower

(b) Image of a boy throwing a baseball

Figure 3.3: Shows different real life images

3.3 Application of interest point detectors

In order to implement the detectors in real life applications, it is important that the interest point detectors are adaptable to changes in parameters and scenes. First, the detectors are assessed for their detection capabilities of interest points on the synthetic image with geometrical shapes without any image transformations. Afterwards, the same experiment will be done, however, with regards to four image transformations, i.e. rotations, scale changes, brightness changes and detection in the presence of Gaussian noise. The focus will be on how well these detectors locates interest points, in this case corners of the geometrical shapes, as well as the consistency of these points. The experiments are divided into the following five categories:

- Image without any transformations
- Experiment of rotational invariance
- Experiment of scale invariance
- Experiment of brightness changes
- Experiment of detection in the presence of Gaussian noise

The goal of each experiment is to detect corner points with the different detectors of KAZE, SURF, and SUSAN, respectively. Note that there are no image transformations added to the real life images as the main purpose of these images is to see how well the detectors are detecting interest points in natural surroundings.

3.3.1 Images without any transformations

The aim of this experiment is to examine the detection abilities of the detectors of corner points in image without any transformations.

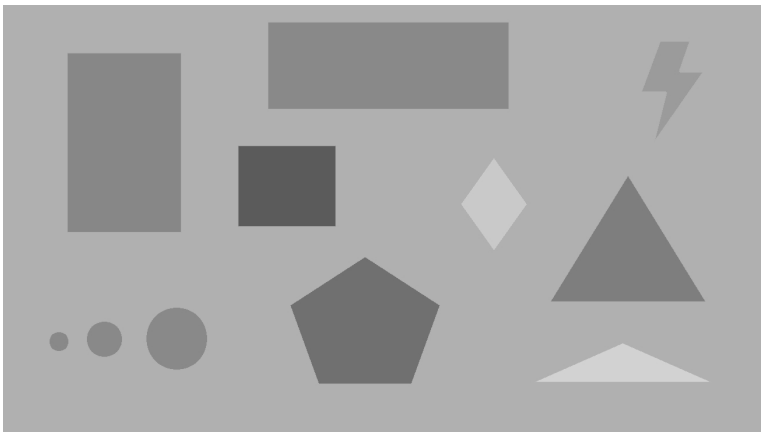
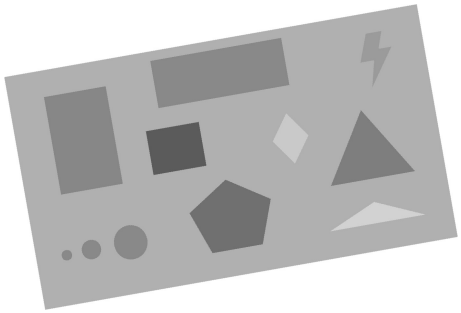


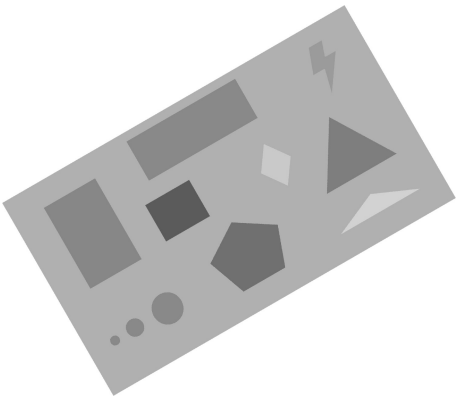
Figure 3.4: Image 1 converted to grayscale and without any transformations or noise

3.3.2 Experiment of rotational invariance

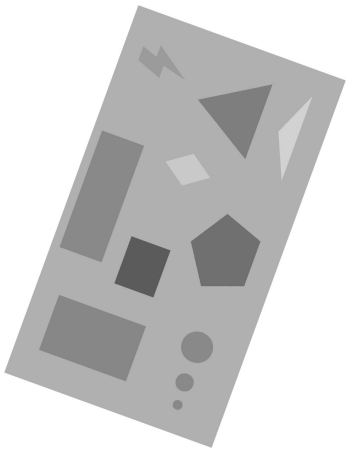
The goal of this experiment is to evaluate the rotational invariance of the detectors. As all three detectors are allegedly rotation invariant, each image were rotated using the function *imrotate*.



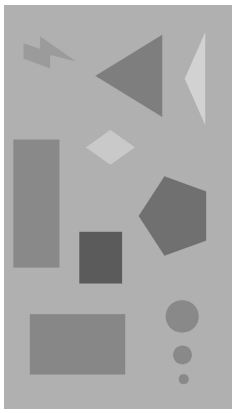
(a) Image rotated 10 degrees



(b) Image rotated 30 degrees



(c) Image rotated 70 degrees

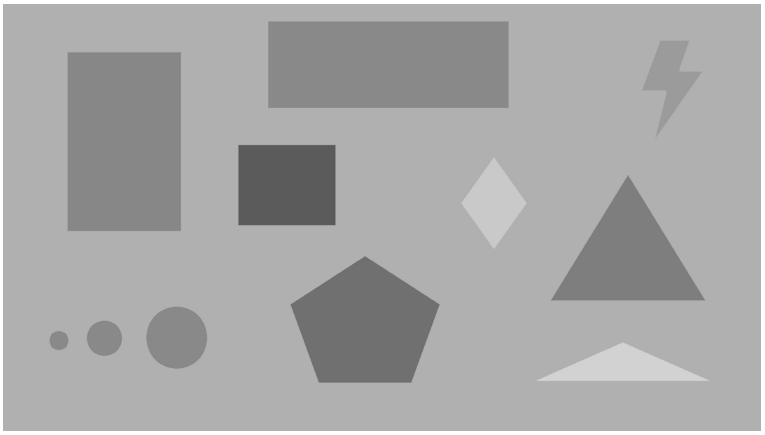


(d) Image rotated 90 degrees

Figure 3.5: Illustration of various rotated images of image 1

3.3.3 Experiment of scale invariance

The experiment of scale invariance is crucial as it confirms or denies the detectors claims of being scale invariant. Images are resized with the MATLAB function *imresize* using scale factor 2 and 0.5, respectively. Note, *imresize* use the interpolation method ”bicubic”.



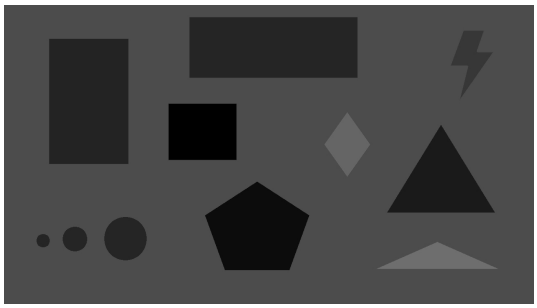
(a) Image 1 resized with scale factor 2

(b) Image 1 resized with scale factor 0.5

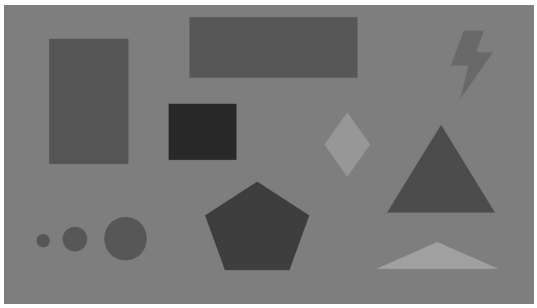
Figure 3.6: Illustration of image 1 with different scale factors

3.3.4 Experiment of brightness changes

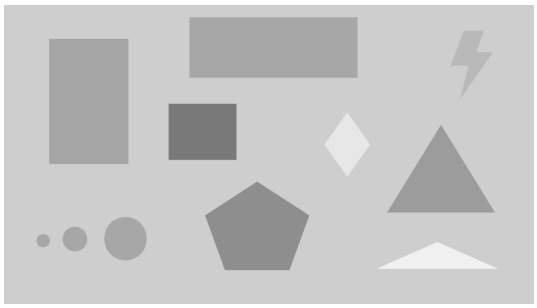
Brightness can differ in many different ways. A change in image brightness is a common problem in real life applications. The detectors are therefore examined for their performance of locating interest points at different brightness variations by adding the number to the output image from *imread*.



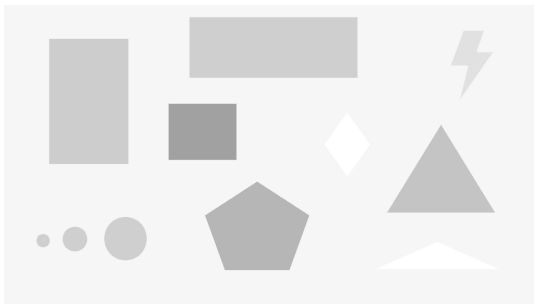
(a) Image 1 with very low brightness



(b) Image 1 with low brightness



(c) Image 1 with high brightness

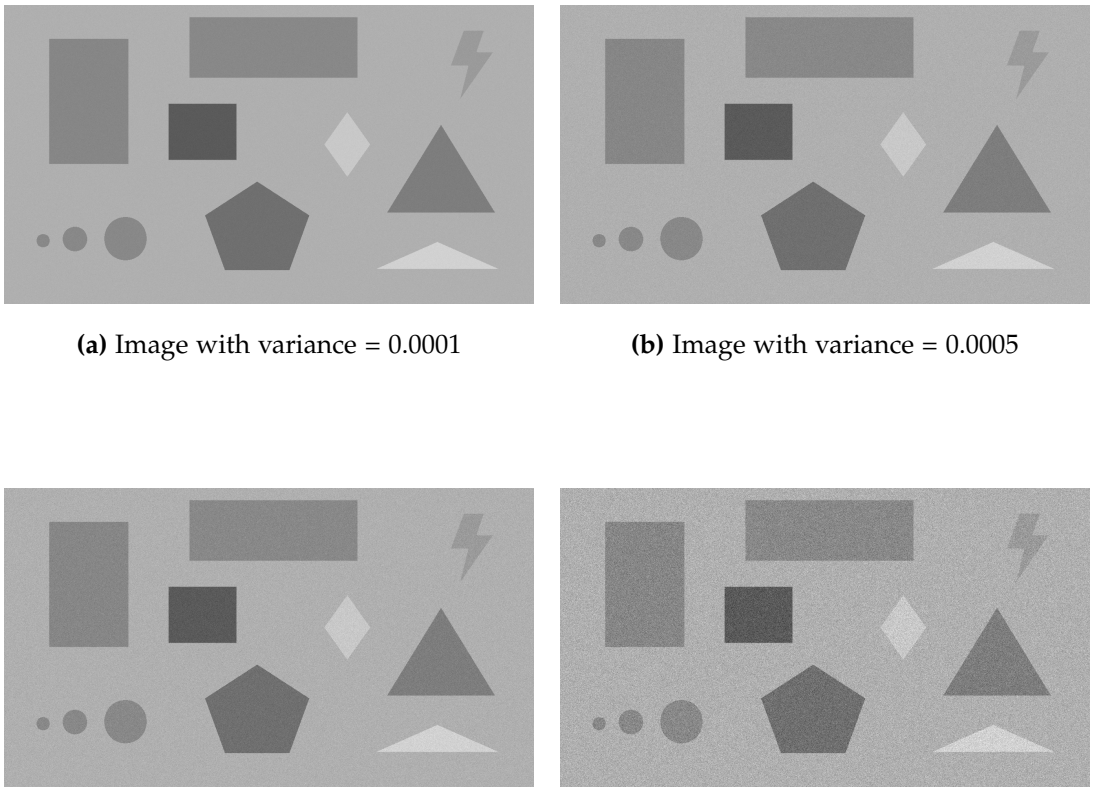


(d) Image 1 with very high brightness

Figure 3.7: Image 1 with different brightness change

3.3.5 Experiment of detection in the presence of Gaussian noise

In this experiment, the goal is to detect the interest points in the presence of Gaussian noise with increasing variance considering the grayscale value images. The mean value is set to zero for all images.



(a) Image with variance = 0.0001

(b) Image with variance = 0.0005

(c) Image with variance = 0.001

(d) Image with variance = 0.005

Figure 3.8: Illustration of Gaussian noise with increasing variance on image 1

4. Results and discussion

In this chapter, the results of all the tests that were done with different images are presented by values, tables and figures. Additionally, the challenges encountered and results that were attained are discussed here. The main purpose of this chapter is to evaluating the KAZE detector abilities to detect interest points in comparison to SURF and SUSAN detectors. There are two main experiments carried out in this project. First main experiment will produce results to each detectors abilities to identify interest points in image 1 based on the experiments mentioned in section 3.3. Second main experiment checks each detectors' ability to detect interest points in real life images. The results will enable interpretation and discussion when comparing KAZE detector with SUSAN and SURF. Additionally, it should be noted that all the detectors mark interest points differently. Both KAZE and SURF mark interest points within the area of the green detection ring. SUSAN allow interest points to be marked with a green detection cross.

4.1 Main experiment on image 1

Before conducting the main experiment on image 1, suitable values for the KAZE detector has to be chosen. The following subsection is an evaluation and comparison of KAZE detectors with different property values used on image 1 without any transformations. The purpose of this evaluation is to determine the property values of the KAZE detector that gives the best results in terms of detecting many true interest points as possible. Thereafter, the KAZE detector with best values are used in comparison with SURF and SUSAN detectors.

4.1.1 Evaluation of KAZE detector with different threshold values

The property values of KAZE consisted of selecting octaves (O), scale levels (S), diffusion type, and threshold value. The octave (O) is chosen according to the resolution and the size of the input image, while the scale levels (S) is selected with regards to that higher scale space value results into smoother scale changes [35]. The diffusion type is set to "Region" which is equivalent to the conductivity function g_2. The threshold value is selected according to an experiment of the image by using multiple threshold values. Furthermore, the image used for this evaluation process was image 1 without

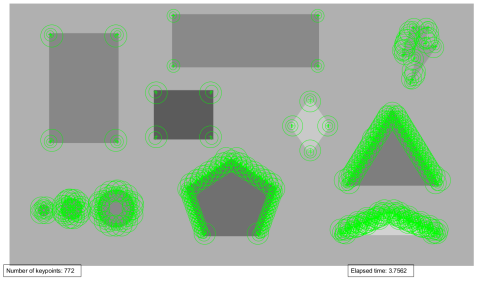
any transformations.

KAZE Detector				
Detectors	*Threshold*	*Octaves*	*Scale levels*	*Diffusion*
KAZEdetect.m	0.00005	2	4	'Region'
	0.0001			
	0.0005			
	0.001			

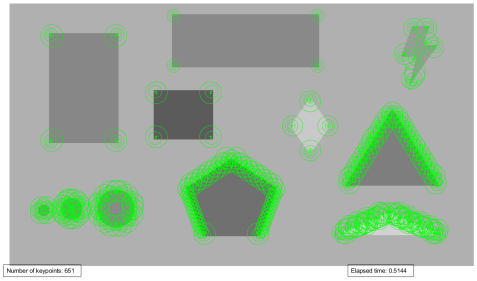
Table 4.1: Property values for the KAZE detector for experiment of different threshold values

Table 4.1 shows the values that are used for the evaluation of the KAZE detector. As seen from the table, the octave (O) is set to the value 2 due to the low resolution of image 1. The scale levels (S) is set to the value 4 which is the highest number of scale levels. The reason for high number of scale levels is to achieve smoother scale changes. When it comes to the threshold values, there are a total of four different threshold values that are selected to be examined against each other to see how many true interest points that are being identified. Note that the threshold value with the best accuracy and most true interest points detected are used throughout this experiment. Additionally, it should be noted that only the threshold values are being examined for this evaluation. The other values remain unchanged.

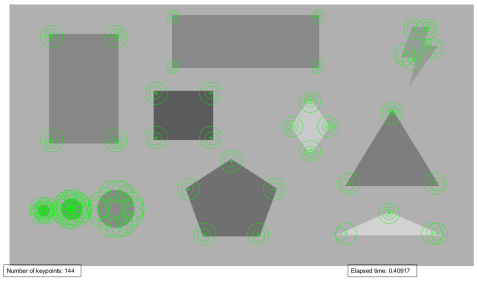
The following Figure 4.1 shows the results of the evaluation with the KAZE detector using the aforementioned threshold values.



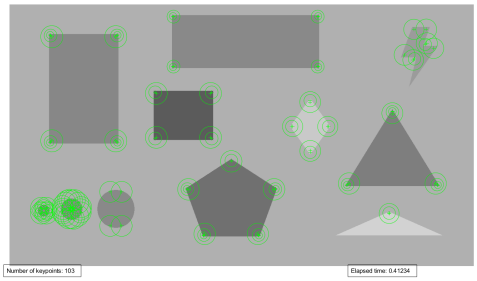
(a) Image 1 with threshold T=0.00005



(b) Image 1 with threshold T=0.0001



(c) Image 1 with threshold T=0.0005



(d) Image 1 with threshold T=0.001

Figure 4.1: Image 1 with different threshold values

The detector has a lot of false interest points detected when using the threshold values 0.00005 and 0.0001. Both thresholds are especially struggling on figures such as the pentagon and both triangles. For that reason, these thresholds are therefore not suitable for further experiment. Furthermore, the threshold values 0.0005 and 0.001 have many true interest points detected. Even though there are a lot more false detected interest points in the threshold value 0.0005 than 0.001, the threshold value 0.0005 is slightly better than 0.001 by detecting two true interest points more. With regards to that, the threshold value 0.0005 is used for the main experiments on image 1 alongside the other values from Table 4.1. The following Figure 4.2 shows the accuracy rate in % for the threshold values evaluated.

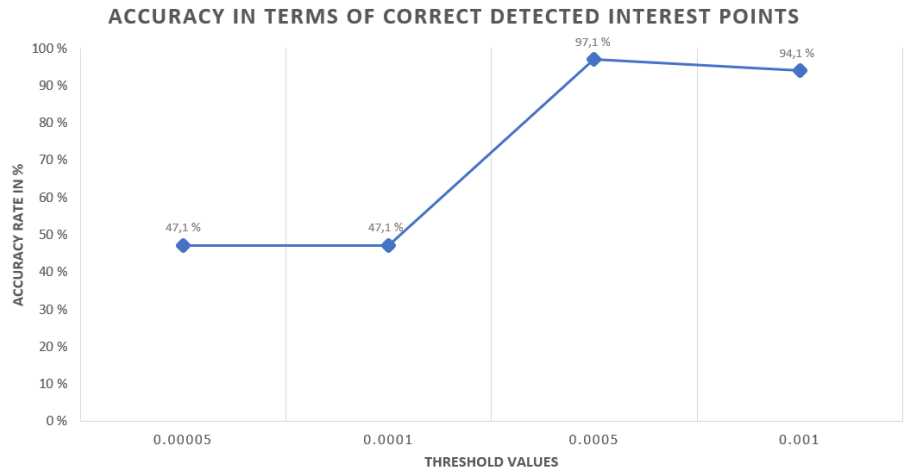


Figure 4.2: Accuracy rate in terms of true detected interest points for various thresholds given in precentage %

4.1.2 Property setup for the main experiment on image 1

The detector setup for KAZE, SURF, and SUSAN for the main experiment on image 1 are explained here. Table 4.2 shows the property values used by the KAZE detector throughout this experiment.

KAZE Detector				
Detectors	*Threshold*	*Octaves*	*Scale levels*	*Diffusion*
KAZEdetect.m	0.0005	2	4	'Region'

Table 4.2: Property values of the KAZE detector for experiment of image 1

The 'MetricThreshold' of the SURF detector was used to determine the number of interest points to appear. Smaller threshold value is equivalent to more interest points, and vice versa. Table 4.3 shows the values used for the SURF detector throughout this experiment. Note that both KAZE and SURF uses the same number of octaves (O) and scale levels (S). The reason for that is to compare the performance of these detectors in terms of detecting interest points using the same octave and scale levels. There were some differences between how the detectors used these octaves and scale levels. However, this will be mentioned in the experiments later.

SURF Detector			
Detectors	*MetricThreshold*	*Octaves*	*Scale levels*
SURF.m	100	2	4

Table 4.3: Property values of the SURF detector for experiment of image 1

The SUSAN method is quite different from the other detectors. SUSAN make use of two geometric threshold values and two intensity values in order to distinguish corners from the edge. The values are set according to what gives corner points rather than edge. Table 4.4 shows values that are used for the SUSAN method throughout this experiment:

SUSAN Detector				
Detectors	*GeoThresh*	*GeoThresh 2*	*IntThresh*	*IntThresh 2*
SUSAN.m	17	19	0.05	0.06

Table 4.4: Property values of the SUSAN detector for experiment of image 1

The following experiments show the results achieved from using the detectors KAZE, SURF and SUSAN with the property values from above.

4.1.3 Results of image 1 without any transformations

The results obtained from KAZE, SURF and SUSAN detecting image 1 without any transformations are discussed here.

KAZE detector

The result obtained from using the KAZE detector is as follows:

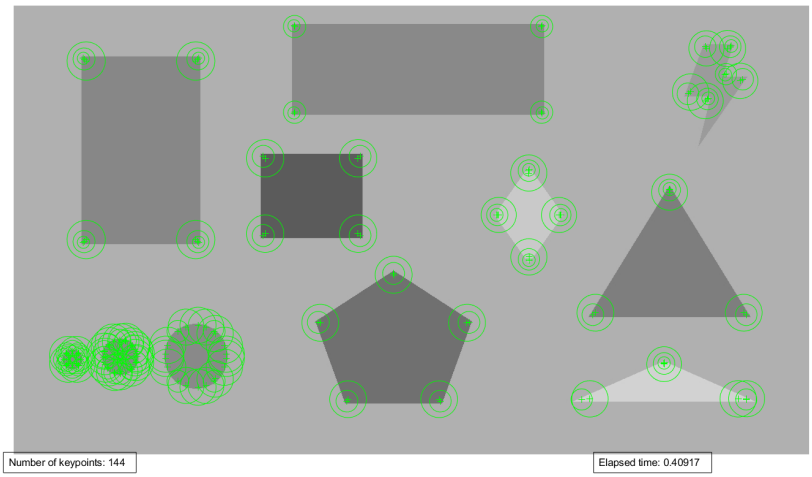


Figure 4.3: Results of image 1 without transformations using KAZE

Figure 4.3 shows that the KAZE detector is capable of detecting almost all true interest points. Only one true interest point is missed out of all possible true interest points. Note that since KAZE detector is detecting in multiple scale levels, there are multiple true interest points detected at the same area. Besides this detector also detects interest points at circles that do not contain any corners. These are false interest points as these

figures do not have any interest points characteristics.

SURF detector

The result obtained from using the SURF detector is as follows:

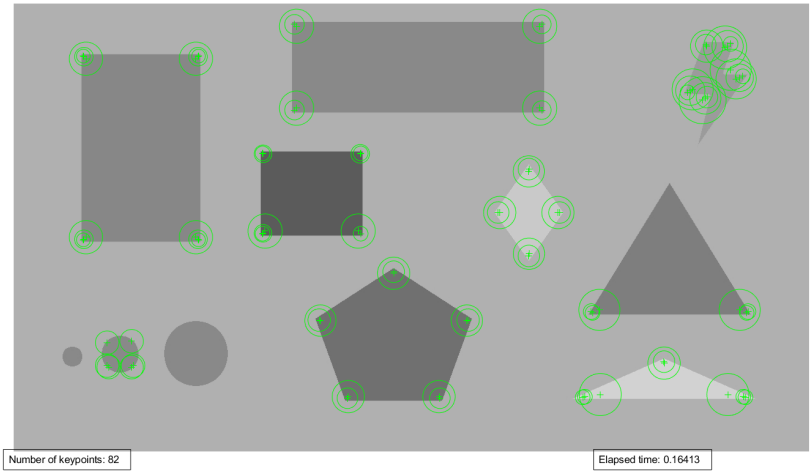


Figure 4.4: Results of image 1 without transformations using SURF

Figure 4.4 shows that SURF detects interest points very well with only two true interest points missed. Note that there are many true interest points with green scale rings of different size that are detected in the same area. This is due to SURF, similar to KAZE, is detecting these interest points at multiple scale levels as well. SURF is also known for being more suitable for blob features rather than corner. However, in this experiment, it can be seen that the detector is detecting corner points accurately. Based on the results obtained here, SURF can be utilized for corner detection as well.

SUSAN detector

The result obtained from using the SUSAN detector is as following:

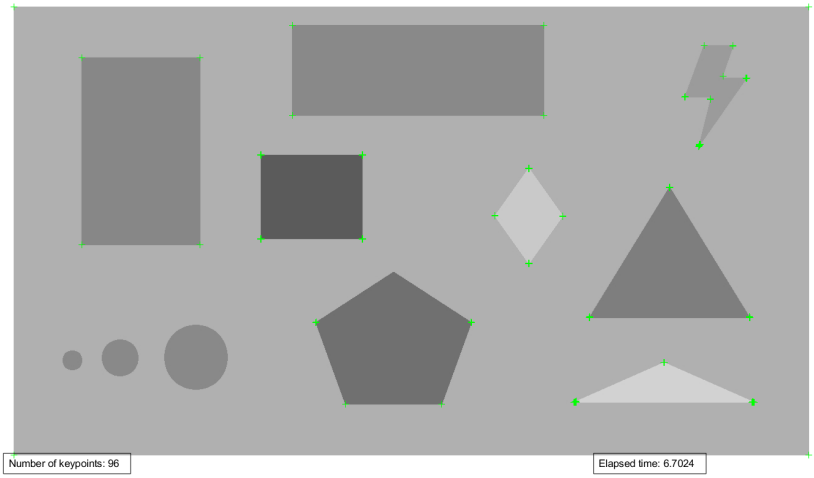


Figure 4.5: Results of image 1 without transformations using SUSAN

The SUSAN detector is very accurate in terms of detecting corner points using the manually assigned values from Table 4.4. Only one interest point is missed out from all possible true interest points. The reason for not localizing this specific interest point is because the angle of the point is either exceeding or equal to the geometric threshold for corner detection. Thus, disregards this point as a corner. A noticeable issue with this detector is that there are many interest points detected in the same area. This is because several circular masks are overlapping each other in the related point area, which results in to many corner points.

Despite the issues mentioned, the detection accuracy of this detector is quite impressive as it is detecting actual corners. Most the points being from the true interest point area. A possible solution is to decrease number of interest points by changing the intensity thresholds, however, this will also decrease the number of true interest points and increase false interest points. For that reason, the values are kept the same in this experiment.

Comparison of the detectors in terms of images without any transformations

Table 4.5 shows the number of interest points detected, time used for the whole process, and time each interest points. Whereas Table 4.6 shows the number of true and false interest points detected and the accuracy the detectors in terms of true interest points detected.

Detectors	*Number of interest points*	*Time [s]*	*Time each interest points [s]*
KAZE	144	0.40917	0.00284
SURF	82	0.16413	0.002
SUSAN	96	6.7024	0.06982

Table 4.5: Number of interest points found using KAZE, SURF and SUSAN and the time usage for these

Table 4.5 shows the detection speed of the detectors. The fastest detector in terms of detecting interest points was SURF with around 0.2 sec, and following up was KAZE detector with around 0.4 sec. The reason for SURF being faster, is due to constructing the nonlinear scale space for KAZE detector is a time consuming process compared to the Gaussian scale space used by SURF. Lastly with almost seven seconds is the SUSAN detector, which was excpected as SUSAN is known for its time-consuming process of detecting interest points accurately.

Detectors	N_{true}	N_{Total}	N_{False}	*Accuracy rate in %*
KAZE	33	34	67	97.1%
SURF	32	34	9	94.1%
SUSAN	33	34	6	97.1%

Table 4.6: Illustrates the accuracy of true detected interest points, as well as the number of true and false detected interest points. N_{true} represents the number of true detected interest points, N_{False} represents the number of false detected interest points and N_{total} represents the total number of interest points in the image. Note that, the N_{true} is number of true detected interest points compared to the total number of interest points N_{total}

Concerning Table 4.6, it can be seen that KAZE alongside SUSAN achieve the best results for both detectors by detecting 33 out of the total 34 true interest points. SURF also does a good job as well by detecting 32 out of the total 34 true interest points as well.

The following Figure 4.6 shows the accuracy rate of the different detectors in terms of detecting true interest points. KAZE alongside SUSAN had the best accuracy rate with 97.1% with regard to detecting true interest points. SURF, on the other hand, had a decent accuracy rate as well with 94.1%. Overall, all detectors are performing well in terms of detecting true interest points.

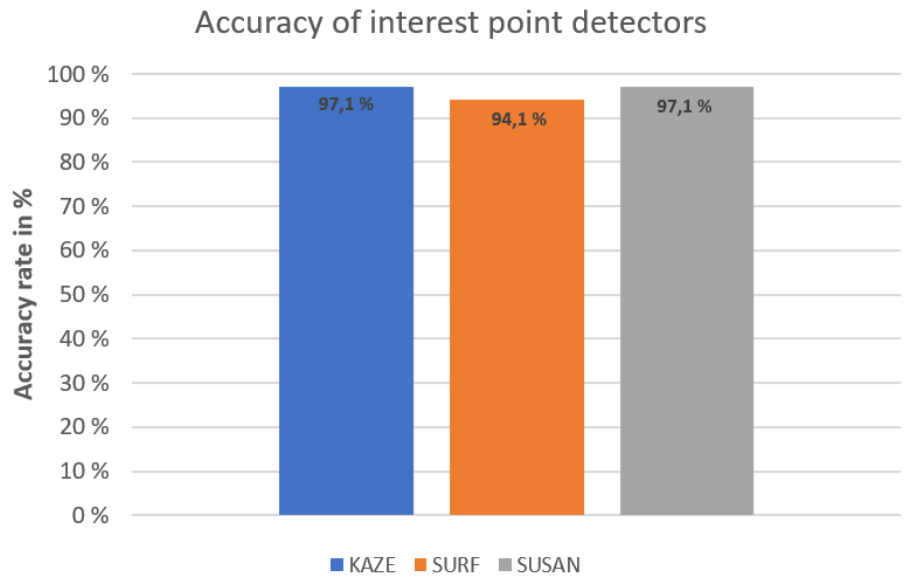
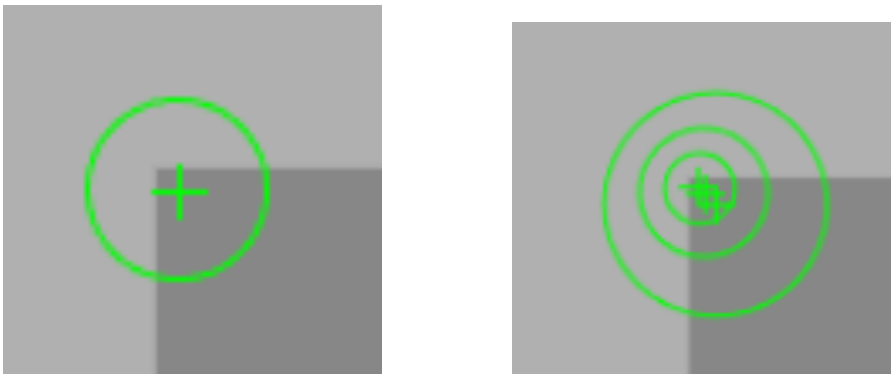


Figure 4.6: Accuracy rate in terms of true detected interest points given in precentage %

As mentioned in the experiments **KAZE detector** and **SURF detector**, these detectors detects multiple interest points in the same point area. This implies to that many interest points are overlapping each other at the same point area. These are not necessarily false interest points, but rather an interest point detected at different scale levels in the same point area. This can be seen in this experiments as well, where some of the true interest points have many green scale rings around it. The following figures show an interest point detected in two different ways.



(a) Single scale detection **(b)** Multiscale detection

Figure 4.7: KAZE detecting an interest point detected using single scale **(a)** and same interest point detected using multiscale **(b)**.

Figure 4.7 shows the detector detecting the same interest point in two different ways, **a** single scale detection and **b** multiscale detection. The difference between these is that in **a**, the detector is detecting the interest point at one scale level, which means that only one interest point is detected in the point area. Whereas in **b**, the detector is detecting the interest point at different scale levels, which means that many interest points are detected in the same point area. More specifically, interest points detected from a multiscale detection are the same interest point detected at different scale levels. This entails to that true interest points obtained from a multiscale detection in a specific point area, are equal to one true interest point in the same specific point area.

Table 4.6 takes this into account. The table section N_{true} from this table considers only one true interest point from each point area. Thus, if there are multiple points detected at different scale levels as seen in Figure 4.7 (**b**), only one interest points are regarded from this point area.

Thus, if all interest points from the other scale levels are included, then, the number of true detected interest points for the KAZE detector is 77. Hence using the knowledge from previous paragraph, the KAZE detector only detects 33 true interest points which is what is written in Table 4.6. When it comes to the rest of the 44 true interest points, these are disregarded due to fact that these interest points are the same as the already labelled 33 true interest points. Additionally, it should also be noted that these are

not included in the number of false detected interest points N_{False} because these are true interest points. Note that this also applies to SURF, as this detector also detects at multiple scale levels. The number of true detected interest points for the SURF detector including the detection from the other scale levels are 73 true interest points. However, same as KAZE, the SURF detector only regards one true interest point from the same true point area. This means that only 32 true interest points are detected for SURF. The rest of the 41 true interest points are therefore disregarded as these are the same interest points as the 32 true interest points. Furthermore, it should be noted that this information implies to all further experiments as well, which means that the main experiments of image 1 regarding rotation, scale changes, brightness changes and presence of Gaussian noise are using this information for their experiment as well.

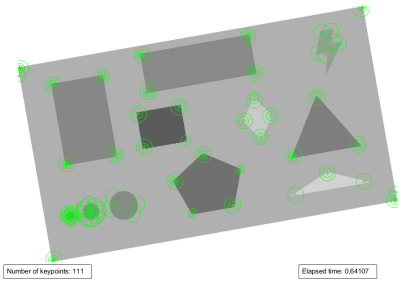
SUSAN, on the other hand, also has many interest points detected on the same point area as well. However, SUSAN is not using any scale space, which means that there might be several circular masks overlapping each other on that related point. Thus, gives many interest points from that area. Similar to KAZE and SURF, SUSAN also regards one true interest point from an area where there might be many true interest points detected for further experiments as well.

4.1.4 Results of image 1 in terms of rotational invariance

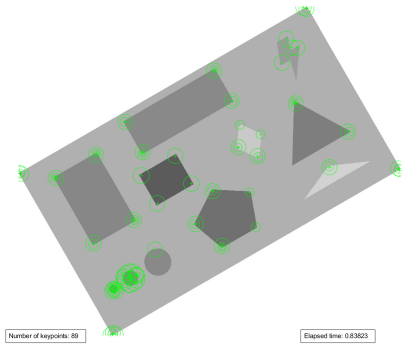
The essential qualities of all the detectors used for this experiment are the fact that they are claimed to rotational invariant. In this section the results obtained from KAZE, SURF and SUSAN of image 1 with different rotations are shown.

KAZE detector

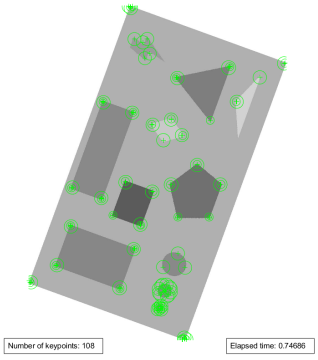
The results obtained from using the KAZE detector are as following:



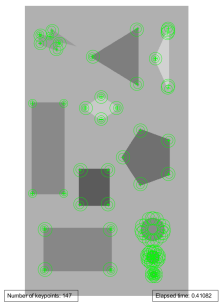
(a) Image rotated 10 degrees



(b) Image rotated 30 degrees



(c) Image rotated 70 degrees



(d) Image rotated 90 degrees

Figure 4.8: Results of various rotated images of image 1 using KAZE

The KAZE detector shown in Figure 4.8 detects the same number of true interest points at rotated angles: 10° and 90°. While for angles 30° and 70°, there are only 5 and 3 true interest points missed respectively. Additionally, it can be seen that there some false

interest points detected as well. It was excepted for the detector to detect all true interest points as KAZE is claimed to be rotational invariant [19]. However, the KAZE detector is based on the determinant of Hessian, which is known for having weakness of detecting interest points around rotated angles, especially for odd multiples for $\frac{\pi}{4}$ or 45°. This is a weakness that all Hessian-based detectors generally have [17]. Figure 4.9 shows the repeatability score between two hessian based detectors that are detecting interest points in terms of rotation up to 180°.

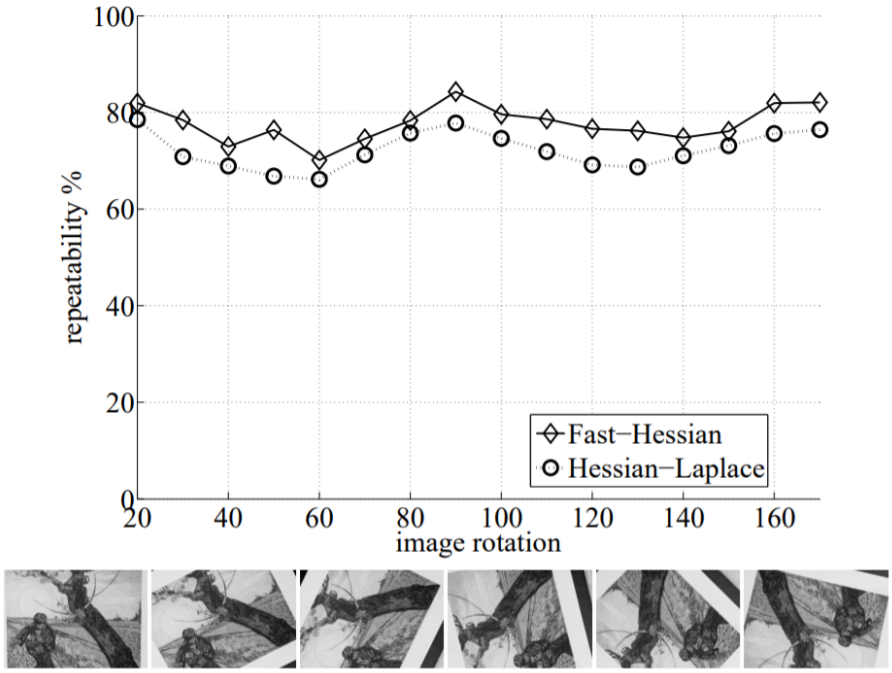
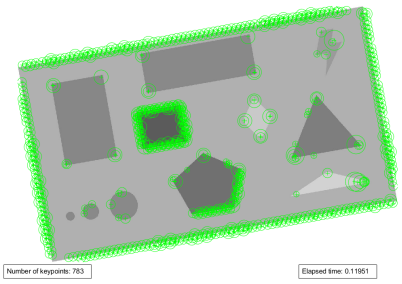


Figure 4.9: Shows how the repeatability score between two Hessian based detectors in terms of rotating the image [17]

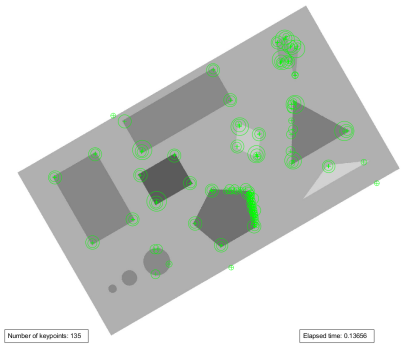
The average repeatability score for Hessian-based detector to detect interest points in rotated images is between 60%-80%. Thus, from this figure it can be concluded that Hessian-based detectors have weakness of detecting interest points in image with different rotations. This is also why KAZE struggled to detect the same amount of true interest points in all angles in this experiment.

SURF detector

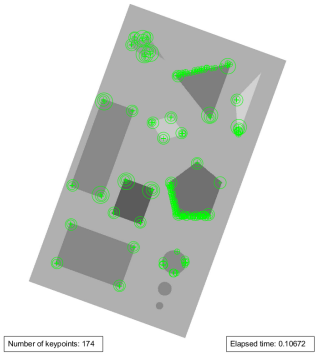
The results obtained from using the SURF detector is as following:



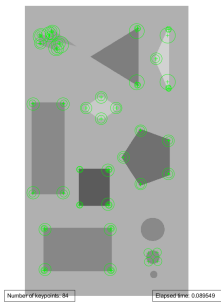
(a) Image rotated 10 degrees



(b) Image rotated 30 degrees



(c) Image rotated 70 degrees



(d) Image rotated 90 degrees

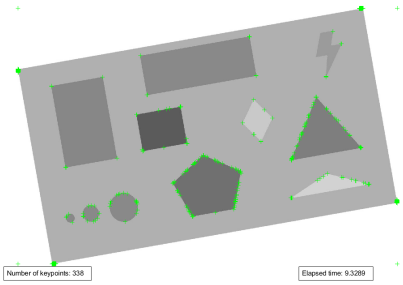
Figure 4.10: Results of various rotated images of image 1 using SURF detector

SURF detector has a massive increase of false detected interest points when rotated with angle 10°, with almost 700 interest points more than the others. The reason for that might be because of the edges having interest point characteristics. Nevertheless, the false detected interest points decreases drastically when the image is rotated more. For 30°, it can be seen that there are some false detected interest points as well. Notably, on the pentagon edge having a lot these false detected points. Pentagon still have false interest points on 70°. Although, the number of false detected interest points eventually decreases when rotating the image to 90°.

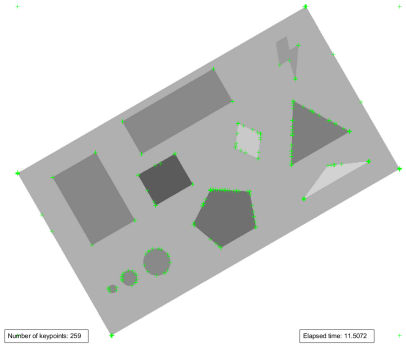
SURF detector, similarly to KAZE, also uses the determinant of Hessian, which means that the weaknesses described in KAZE detector regarding the detection of interest points around rotated angles also applies here.

SUSAN detector

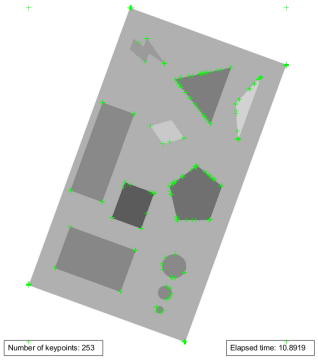
The result obtained from using the SUSAN detector is as following:



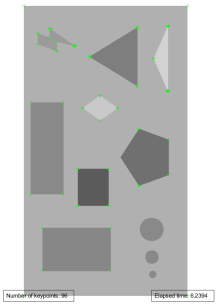
(a) Image rotated 10 degrees



(b) Image rotated 30 degrees



(c) Image rotated 70 degrees



(d) Image rotated 90 degrees

Figure 4.11: Results of various rotated images of image 1 using SUSAN detector

The number of false interest points detected for the SUSAN detector is elevated at 10° but gradually decreases when rotating to 70°. Regardless of that, the number of false interest points detected are still high. This could be due to the circular mask of SUSAN is placed on figures that might be too big, which makes it more of an edge detector rather than a corner. Thus many false interest points are being detected.

However, when it comes to true interest points, SUSAN manages to detect a lot more true interest points at 90° compared to the other angles.

Comparison of detectors in terms of rotational invariance

Detectors	*Angles*	*Number of interest points*	*Time [s]*	*Time each interest points [s]*
KAZE	0°	144	0.40917	0.00284
	10°	111	0.64107	0.00578
	30°	89	0.83823	0.00942
	50°	97	0.86565	0.00892
	70°	108	0.74686	0.00692
	90°	147	0.41082	0.00279
SURF	0°	82	0.16413	0.002
	10°	783	0.11951	0.00015
	30°	135	0.13656	0.00101
	50°	126	0.20546	0.00163
	70°	174	0.10672	0.00061
	90°	84	0.089549	0.00107
SUSAN	0°	96	6.7024	0.06982
	10°	338	9.3289	0.02760
	30°	259	11.5072	0.04443
	50°	284	13.4162	0.04724
	70°	253	10.8919	0.04305
	90°	96	6.2394	0.06499

Table 4.7: Number of interest points found using KAZE, SURF and SUSAN in terms of rotation and the time usage for these detectors

Table 4.7 illustrates the number of interest points and the time usage of the different detectors in for various rotations. As seen from the table, the SURF detector is still the fastest detector. Next is KAZE detector and the slowest detector is SUSAN. The speed of the different detectors is similar to the first experiment. Although, there was

a difference in the number of interest points detected. Especially for SURF and SUSAN which had many more interest points detected.

Detectors	*Angles*	N_{true}	N_{Total}	N_{False}	*Accuracy rate in %*
KAZE	0°	33	34	67	97.1%
	10°	33	34	43	97.1%
	30°	29	34	38	85.3%
	50°	32	34	38	94.1%
	70°	31	34	44	91.2%
	90°	33	34	77	97.1%
SURF	0°	32	34	9	94.1%
	10°	22	34	754	64.7%
	30°	25	34	92	73.5%
	50°	28	34	72	82.4%
	70°	20	34	131	58.8%
	90°	32	34	9	94.1%
SUSAN	0°	33	34	6	97.1%
	10°	19	34	319	55.9%
	30°	18	34	241	52.9%
	50°	11	34	273	32.4%
	70°	20	34	233	58.8%
	90°	33	34	6	97.1%

Table 4.8: Illustrates the accuracy of true detected interest points, the number of true and false detected interest points.

Table 4.8 shows how many true and false interest points are detected for different rotations. The detectors tested here claimed to be rotational invariance. This claim is partly true as the detectors are only detecting the same amount of true detected interest points when the image is rotated with 90° as done in the image at 0°. However, these detector failed for the other angles. It can therefore be concluded that these detectors fail in terms of rotational invariance due to not detecting the same amount of true interest points.

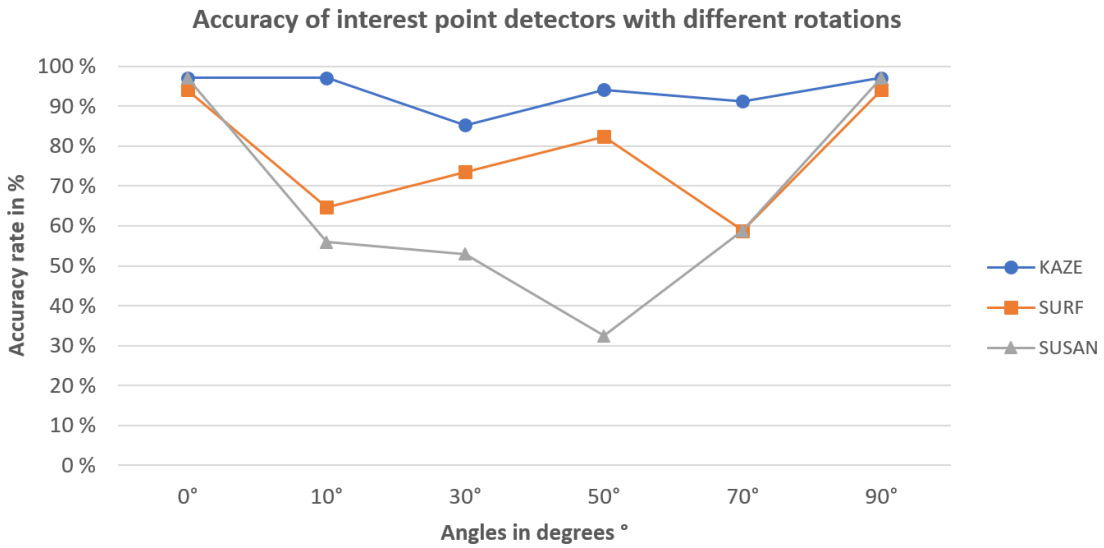


Figure 4.12: Accuracy in terms of true detected interest points at different rotated angle

Detectors	*Average accuracy rate*
KAZE	92.7%
SURF	72.8%
SUSAN	66.2%

Table 4.9: Illustrates the average accuracy rate of true detected interest points

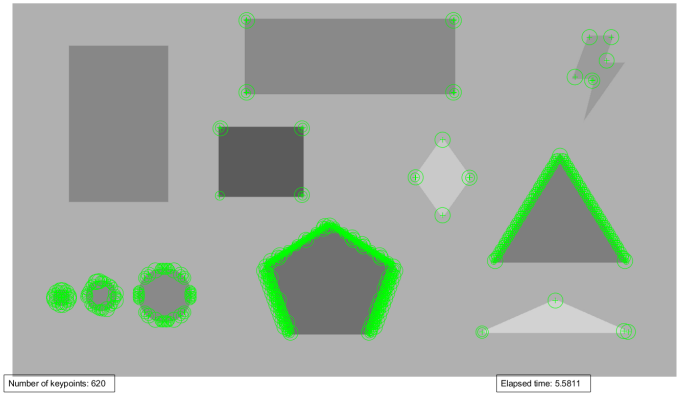
Figure 4.12 shows the accuracy rate of the different detectors in terms of detecting the true interest points at different angles. It can be seen that KAZE had the best accuracy rate through all angles in terms of detecting true interest points with an average accuracy rate of 92.7%. Following up was SURF with an average accuracy rate of 72.8%, and SUSAN had the worst average accuracy rate of 66.2% compared to the others.

4.1.5 Results of image 1 in terms of scale invariance

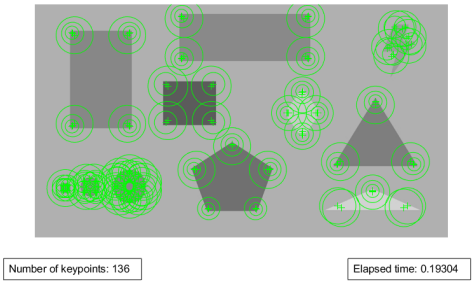
Both SURF and KAZE are known for being scale invariant detectors, while SUSAN is a corner detector. In this section, the results obtained from KAZE, SURF, and SUSAN of image 1 with different scale changes are shown.

KAZE detectors

The results obtained from using the KAZE detector in terms of different scale changes are as following:



(a) Image 1 resized with scale factor 2



(b) Image 1 resized with scale factor 0.5

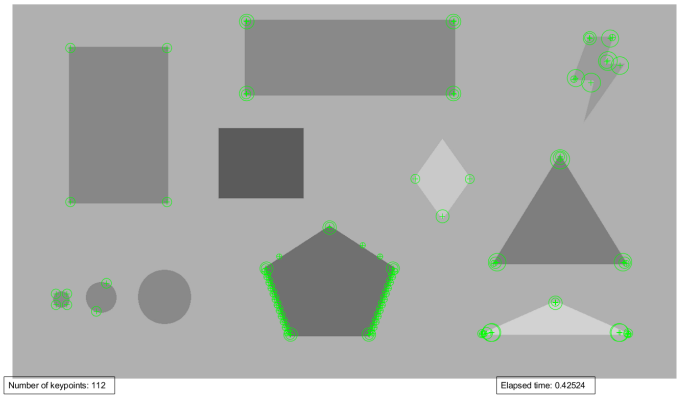
Figure 4.13: Results of image 1 with scale changes using KAZE detector

It can be seen that the KAZE detector struggles to detect true interest points at all figures when the image is resized with scale factor 2, whereas for image resized with scale factor 0.5, the detector only misses one true interest point. Besides, there is a lot

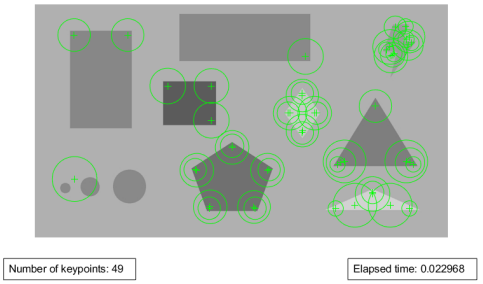
of false detected interest points in the image with scale factor 2, especially on the dark triangle and the pentagon. KAZE is known for being scale invariant, but it is only apparent when the image is resized with scale factor 0.5 but fails when the image is resized with scale factor 2.

SURF detector

The results obtained from using the SURF detector are as following:



(a) Image 1 resized with scale factor 2



(b) Image 1 resized with scale factor 0.5

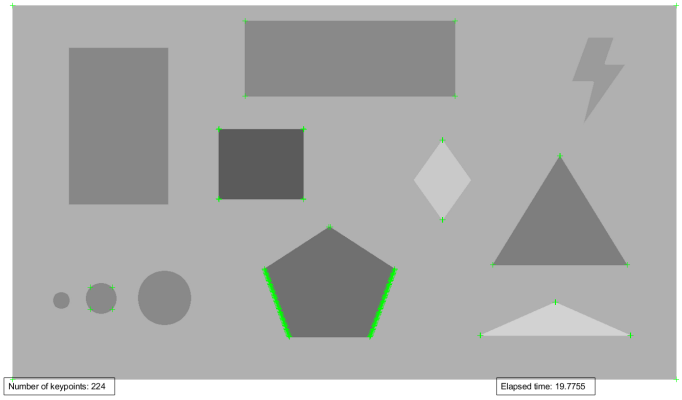
Figure 4.14: Results of image 1 with scale changes using SURF detector

SURF detector is detecting a lot more true interest points than KAZE in the image with scale factor 2. However, the detector has some issues at some of the figures. The detector misses out all interest points on the black square, and there are too many false interest points detected on the pentagon. When it comes to image with scale factor 0.5, the detector is missing out six true interest points. Despite detecting more

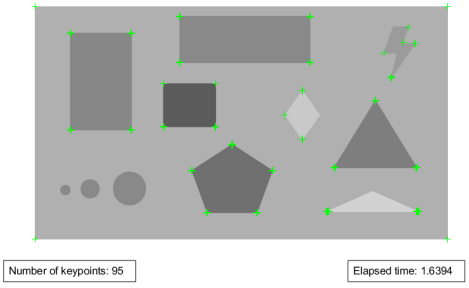
true interest points than KAZE in the image with scale factor 2, the SURF detectors falls a bit behind KAZE in terms of detecting interest points at images resized with scale factor 0.5.

SUSAN detector

The results obtained from using the SUSAN detector are as following:



(a) Image 1 resized with scale factor 2



(b) Image 1 resized with scale factor 0.5

Figure 4.15: Illustration of image 1 with different scale factors

SUSAN detector also misses out figures when it comes to the image with scale factor 2. Similar to KAZE and SURF, SUSAN has a lot of false interest points detected on the pentagon. While for the image with scale factor 0.5, SUSAN only misses out two true interest points. SUSAN does a decent job considering that this detector is not a scale invariant to scale changes.

Comparison of detectors in terms of scale change

Detectors	*Scale factor*	*Number of interest points*	*Time [s]*	*Time each interest points [s]*
KAZE	0.5	136	0.19304	0.001419
	2	620	5.5811	0.009
SURF	0.5	49	0.022968	0.00047
	2	112	0.42524	0.003797
SUSAN	0.5	95	1.6394	0.01726
	2	224	19.7755	0.08828

Table 4.10: Number of interest points found using KAZE, SURF and SUSAN in terms of rotation and the time usage for these detectors

Table 4.10 illustrates the speed of the different detectors in terms of scale changes. In terms of speed, the SURF detector is fastest to detect interest points for images resized with scale factor 0.5 and 2 respectively, while SUSAN was the slowest for both scale factors. When it comes to quantity of interest points detected, the KAZE detector detects most interest points for images resized with both scale factors 0.5 and 2.

Detectors	*Scale factor*	N_{true}	N_{Total}	N_{False}	*Accuracy rate in %*
KAZE	0.5	33	34	51	97.1%
	1	33	34	67	97.1%
	2	20	34	568	58.8%
SURF	0.5	27	34	4	79.4%
	1	32	34	9	94.1%
	2	24	34	70	70.6%
SUSAN	0.5	31	34	8	91.2%
	1	33	34	6	97.1%
	2	17	34	207	50%

Table 4.11: Illustrates the accuracy of true detected interest points, the number of true and false detected interest points.

Table 4.11 shows how many true and false interest points are detected for different scale changes. Note that the scale factor 1 represents the image at the original scale, which is the same as image without any transformations. The reason for including is to see how the detectors with different scale change are detecting compared to the original image.

As seen from the table, KAZE only misses one true interest point when the image is resized with scale factor 0.5. It detects the same amount of true interest points as done in the image with scale factor 1. The detectors are struggling when it comes to detecting true interest points at image resized with scale factor 2. SURF detector was the only detector that had a decent result with 70.6% in accuracy rate when the image was resized with scale factor 2. Additionally, the SURF detector was also the detector with fewest false interest points detected as well. The following Figure 4.16 shows the accuracy rate of the different detectors in terms of detecting true interest points at different scale changes. Note that both SUSAN and SURF have a slightly less accuracy rate when the image is resized with scale factor 0.5, while KAZE has the same accuracy rate. In regards to scale factor 2, all detectors have smaller accuracy rate when compared to the other scale factors.

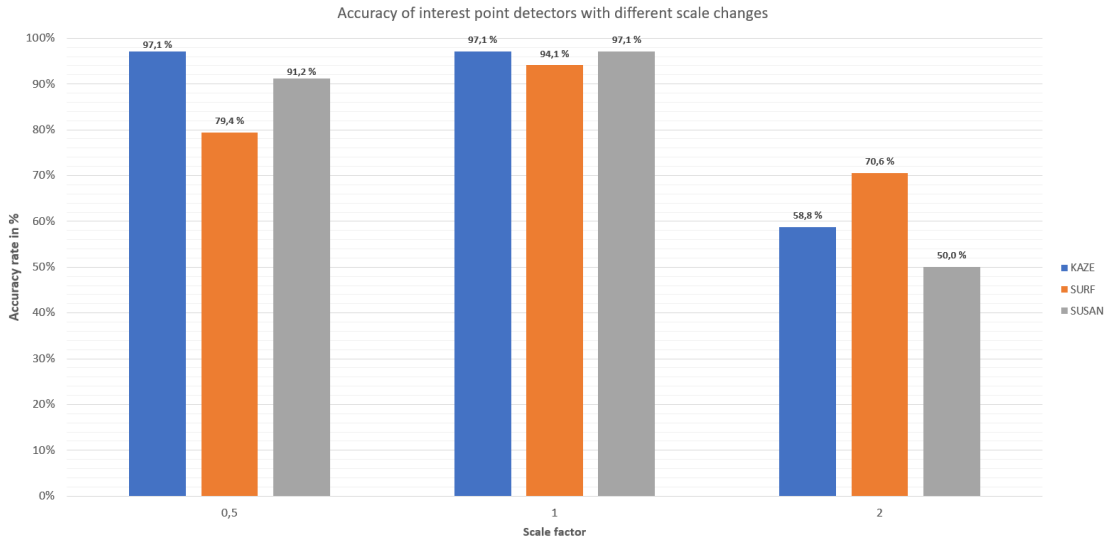


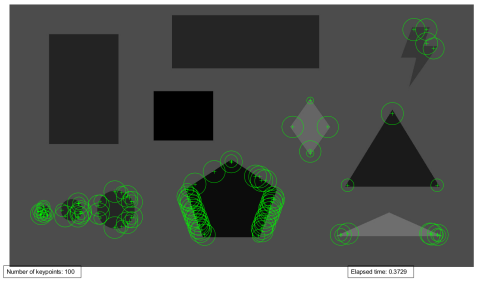
Figure 4.16: Accuracy in terms of true detected interest points at different scale factors

4.1.6 Results of image 1 in terms of brightness changes

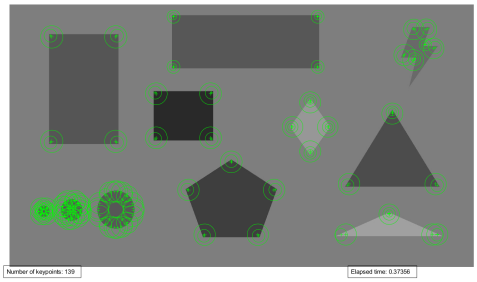
A change in image brightness is a common problem that might occur in real life applications. In this experiment, the detectors are tested at the different change of brightness. The brightness is tested in the presence of low and high brightness changes.

KAZE detector

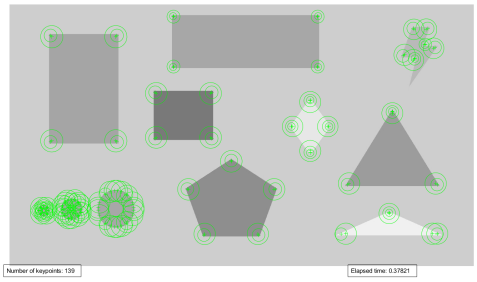
The results for KAZE in terms of different brightness changes are as following.



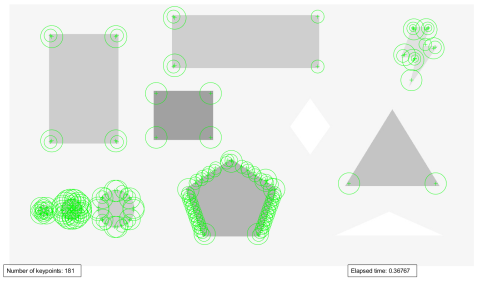
(a) Image 1 with very low brightness change



(b) Image 1 with low brightness change



(c) Image 1 with high brightness change



(d) Image 1 with very high brightness change

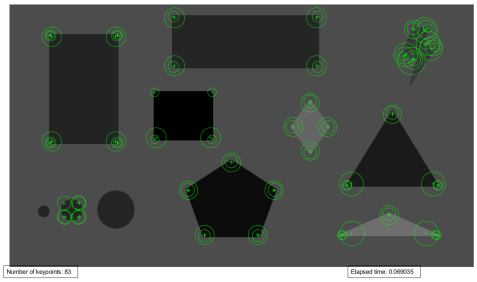
Figure 4.17: Results of image 1 with different brightness change using KAZE detector

By looking closely at all figures, it can be seen that the KAZE detector misses a lot of true interest points at very high and low brightness change. Additionally, there are a lot of false detected interest points in these figures e.g. pentagon and circles. Note that the detector still detecting false interest points at circles as well. Despite that, the detector is actually capable of detecting interest points at low and high brightness. It

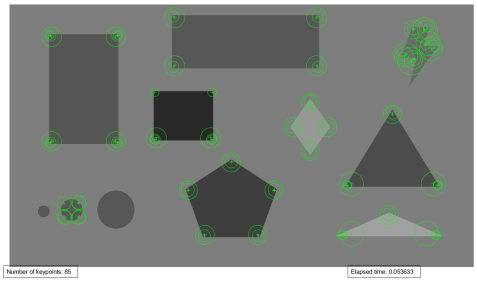
should be noted that detectors that are based on the determinant of Hessian might struggle to detect interest points at very high brightness. This is due to the fact that in order for the KAZE detector to detect any interest points, the determinant of Hessian searches for strong change in the image gradient in two directions. A corner is detected where the change in the image gradient occurs in two directions [39]. However, in the image with very high brightness, the both the diamond and triangle are too similar to the background, which means there is no change in the image gradients in any direction. Therefore interest points on these figures are not detected.

SURF detector

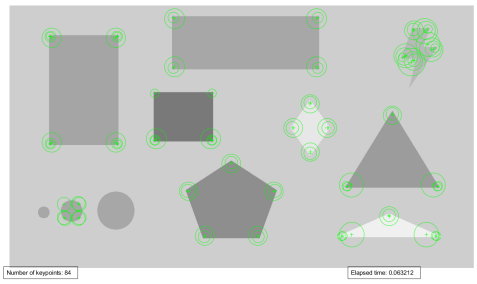
The results obtained from using the SURF detector are as following:



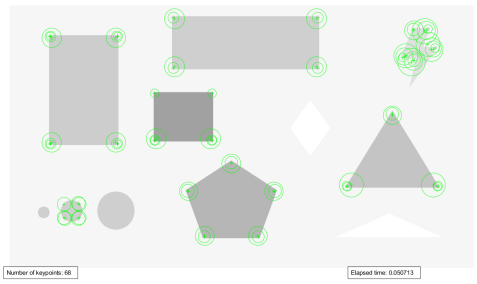
(a) Image 1 with very low brightness change



(b) Image 1 with low brightness change



(c) Image 1 with high brightness change



(d) Image 1 with very high brightness change

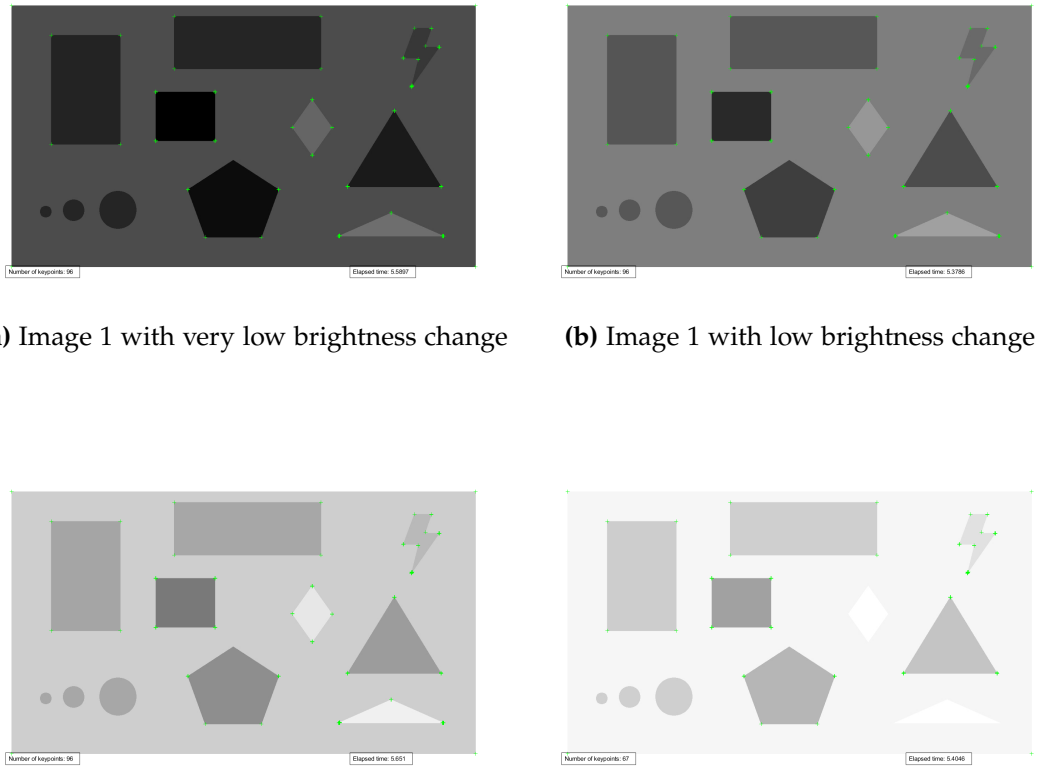
Figure 4.18: Results of image 1 with different brightness change using SURF detector

SURF is capable of detecting many true interest points at very low brightness, which makes SURF useful for detection in the presence of low brightness changes. However,

the detector struggles when the brightness is increased to a very high value. This is due to the fact that this detector also uses determinant of hessian, which means the same things that were mentioned for the KAZE detector applies here as well.

SUSAN detector

The results obtained from using the SUSAN detector are as following:



(a) Image 1 with very low brightness change

(b) Image 1 with low brightness change

(c) Image 1 with high brightness change

(d) Image 1 with very high brightness change

Figure 4.19: Results of image 1 with different brightness change using SUSAN detector

Similar to SURF, the SUSAN detector is also capable of detecting interest points at very low brightness. In fact, SUSAN had the capabilities of detecting the same amount of true interest points for three of the experiments. However, similar to both KAZE and SURF, SUSAN did not manage to detect interest points at both diamond and triangle at very high brightness. This is because these figures were too bright, almost the same as the background. For that reason, SUSAN was not able to place the circular mask as there is no clear distinction between the background and figures. Despite this

experiment, the detector was capable of detecting the same amount of true interest points for three of the experiments.

Comparison of detectors in terms of brightness change

Detectors	*Brightness change*	*Number of interest points*	*Time [s]*	*Time each interest points [s]*
KAZE	None	144	0.40917	0.00284
	Very low	100	0.3729	0.003729
	Low	139	0.37356	0.002687
	High	139	0.37821	0.002721
	Very high	181	0.36767	0.002031
SURF	None	82	0.16413	0.002
	Very low	83	0.069035	0.000832
	Low	85	0.053633	0.000631
	High	84	0.063212	0.000753
	Very high	68	0.050713	0.000746
SUSAN	None	96	6.7024	0.06982
	Very low	96	5.5897	0.05823
	Low	96	5.3786	0.05603
	High	96	5.651	0.05886
	Very high	67	5.4046	0.08067

Table 4.12: Number of interest points found using KAZE, SURF, and SUSAN for brightness variations and the time usage for these detectors

Table 4.12 shows the number of interest points detected and time usage for the different detectors in terms of detecting at various brightness variations. It can be seen that in terms of speed, SURF is still the fastest detector and SUSAN is still the slowest. Note that SUSAN is capable of detecting the same amount of interest points at three of the experiments. All detectors are struggling at very high brightness by detecting more or less interest points compared to image with no change. Additionally, it can also be seen that KAZE has a different amount of detected interest points at very high

and low brightness.

Detectors	*Brightness change*	N_{true}	N_{Total}	N_{False}	*Accuracy rate in %*
KAZE	None	33	34	67	97.1%
	Very low	14	34	83	41.2%
	Low	33	34	66	97.1%
	High	33	34	67	97.1%
	Very high	21	34	146	61.8%
SURF	None	32	34	9	94.1%
	Very low	33	34	7	97.1%
	Low	33	34	9	97.1%
	High	33	34	7	97.1%
	Very high	26	34	7	76.5%
SUSAN	None	33	34	6	97.1%
	Very low	33	34	6	97.1%
	Low	33	34	6	97.1%
	High	33	34	6	97.1%
	Very high	26	34	12	76.5%

Table 4.13: Illustrates the accuracy of true detected interest points, the number of true and false detected interest points.

Table 4.13 shows how many true and false interest points are being detected for different brightness changes. Both SURF and SUSAN do a great job of detecting true interest points in the three first experiment with only one interest point missed out in these experiments. When it comes to KAZE, the detector struggled to detect the same amount of interest points at very high and low brightness. However, KAZE detects the same amount of true interest points at low and high brightness. Common for all detectors, is the fact that they all struggled at very high brightness where two important figures were missed out due to the similarity between the background and the figures. Additionally, it can also be seen that SURF alongside SUSAN have few false interest points detected compared to KAZE.

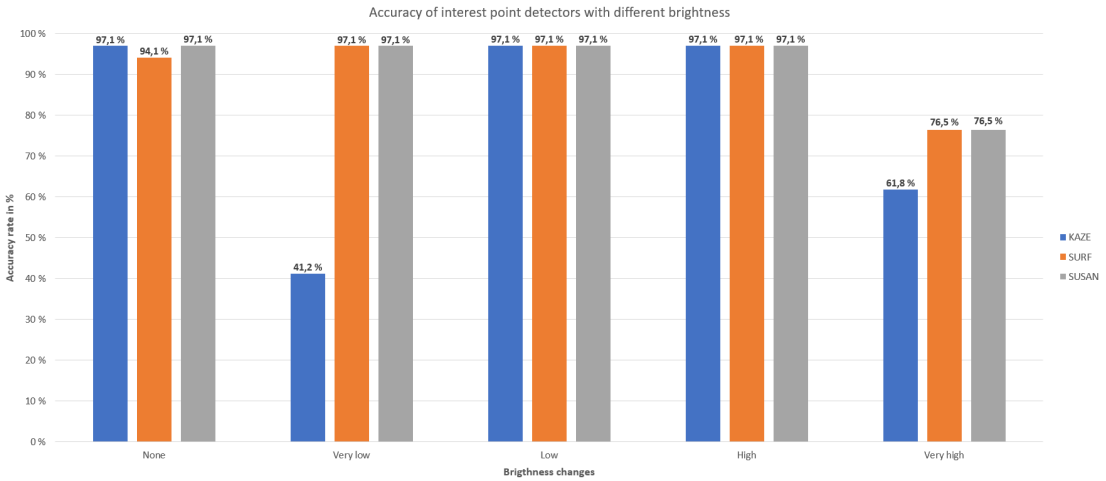


Figure 4.20: Accuracy in terms of true detected interest points at different brightness

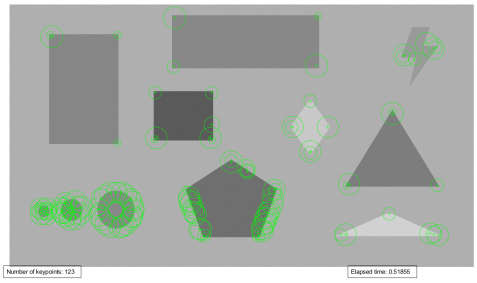
Figure 4.20 shows that all detectors have a good accuracy rate for low and high brightness. However, the KAZE detector struggles at both very low and very high brightness, while SURF and SUSAN struggle at very high brightness. Note that SURF detector has better accuracy rate at very low, low, and high brightness compared to the image without any brightness which is quite impressive.

4.1.7 Results of image 1 in the presence of Gaussian noise

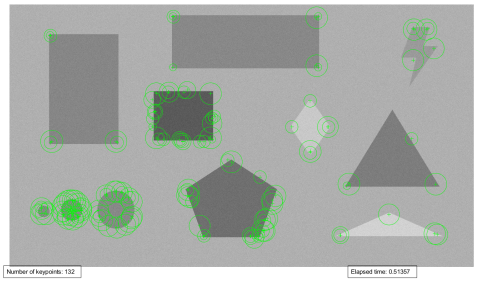
This experiment is carried out in the purpose of testing the KAZE detector in the presence of Gaussian noise as this detector makes use of nonlinear diffusion filtering (NDF) which is good in the presence of noise [19]. KAZE is tested in comparison with SURF and SUSAN. SURF uses linear filtering (Gaussian) and SUSAN uses circular masks. SUSAN is also known for being robust to noise as this detector does not use any image derivatives [8]. The results obtained from KAZE, SURF, and SUSAN of image 1 in the presence of Gaussian noise are shown as following:

KAZE detector

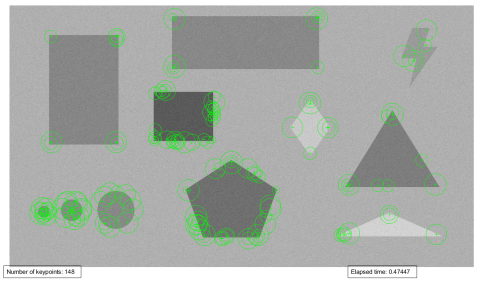
The results obtained using the KAZE detector in the presence of various Gaussian noise are as following:



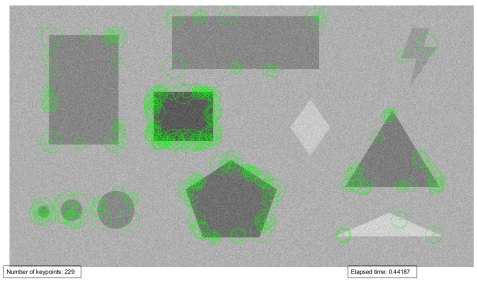
(a) Image with variance = 0.0001



(b) Image with variance = 0.0005



(c) Image with variance = 0.001



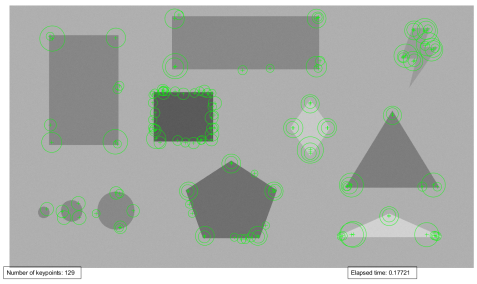
(d) Image with variance = 0.005

Figure 4.21: Results of image 1 with different variances of Gaussian noise using KAZE

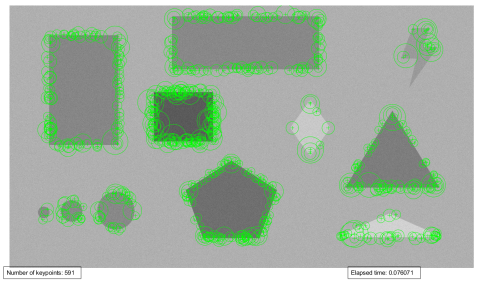
The KAZE detector is known for the utilization of the nonlinear diffusion filtering which is capable of reducing the noise, simultaneously, retain important image details such as object boundaries and edges [19]. As seen from the figures, the detector is detecting some true interest points, but at the same time having many false interest points as well. Additionally, it can be seen that false interest points are expanding when the variance is increased, especially for the black square which goes from one false interest point detected to several false interest points. Same goes for the pentagon as well, where the number false interest points increases as well.

SURF detector

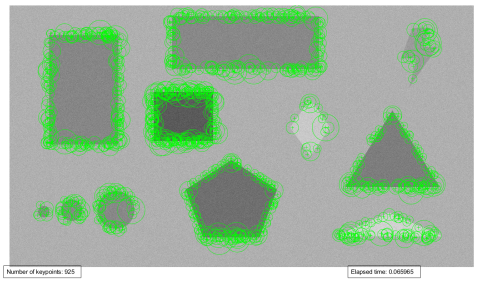
The results obtained from using the SURF detector are as following:



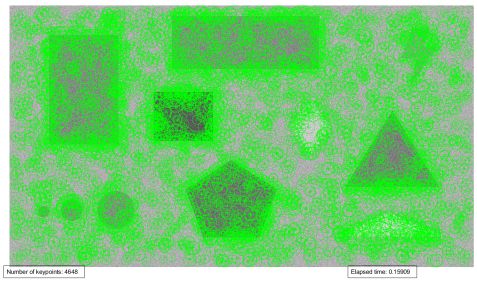
(a) Image with variance = 0.0001



(b) Image with variance = 0.0005



(c) Image with variance = 0.001



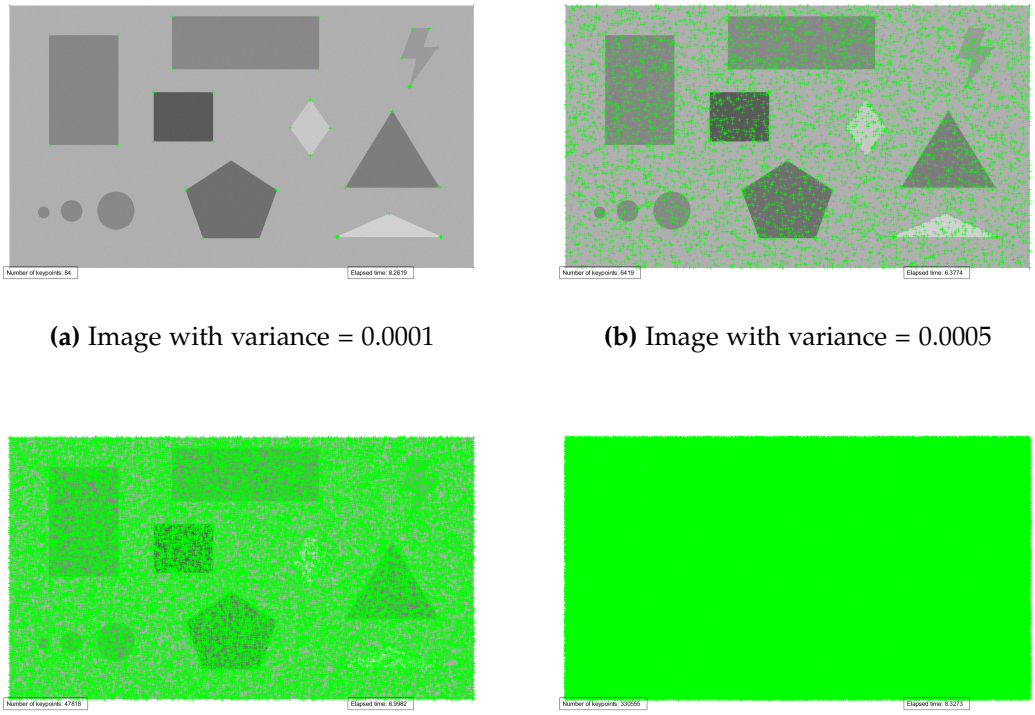
(d) Image with variance = 0.005

Figure 4.22: Image 1 with different variances of Gaussian noise using SURF

The SURF detector has a lot of false interest points detected when detecting in the presence of Gaussian noise. It can be seen that the number of false detected interest points expands when the variance of Gaussian noise is increased. It can be concluded that the detector is not robust for detecting interest points in the presence of Gaussian noise.

SUSAN detector

The results obtained from using the SUSAN detector are as following:



(a) Image with variance = 0.0001

(b) Image with variance = 0.0005

(c) Image with variance = 0.001

(d) Image with variance = 0.005

Figure 4.23: Image 1 with different variances of Gaussian noise using SUSAN

As mentioned earlier, SUSAN is robust to noise as the detector does not use any image derivative [8]. SUSAN starts promising by detecting a lot of true interest points. However, as the variance is increasing, so does the number of false interest points as well. Especially for variance values 0.001 and 0.005, which is just very poor in terms of detecting interest points. Thus, from this experiment, it can be seen that the SUSAN detector is not robust for Gaussian noise and not suited for detection in the presence of this kind of noise. Note that, tweaking with geometric threshold and intensity values can give better results. However, this will also affect the other results as well.

Comparison of the detectors in presence of Gaussian noise

Detectors	*Variance*	*Number of interest points*	*Time [s]*	*Time each interest points [s]*
KAZE	0	144	0.40917	0.00284
	0.0001	123	0.51855	0.00422
	0.0005	132	0.51357	0.00389
	0.001	148	0.47447	0.00321
	0.005	229	0.44187	0.00193
SURF	0	82	0.16413	0.002
	0.0001	129	0.17721	0.00137
	0.0005	591	0.076071	0.000129
	0.001	925	0.065965	0.00007
	0.005	4648	0.15909	0.00003
SUSAN	0	96	6.7024	0.06982
	0.0001	84	8.2619	0.09836
	0.0005	6419	6.3774	0.00099
	0.001	47818	6.9982	0.000146
	0.005	330555	8.3273	0.00025

Table 4.14: Number of interest points found using KAZE, SURF and SUSAN in presence of Gaussian noise and the time usage for these detectors

Table 4.14 shows the number of interest points detected and the time usage of these in terms of detecting in the presence of Gaussian noise. Both SURF and SUSAN have many interest points detected, most of them being false detected interest points. It is abnormal for a detector to detect these many points. These detectors are therefore not robust or suitable for detection in the presence of Gaussian. In contrast, KAZE has more stable detection. In regards to speed, there is no point in discussing it as two of the detectors have a lot of false detected interest points.

Detectors	*Gaussian*	N_{true}	N_{Total}	N_{False}	*Accuracy rate in %*
KAZE	0	33	34	67	97.1%
	0.0001	26	34	85	76.5%
	0.0005	21	34	99	61.8%
	0.001	22	34	111	64.7%
	0.005	11	34	210	32.4%
SURF	0	32	34	9	94.1%
	0.0001	25	34	62	73.5%
	0.0005	14	34	571	41.2%
	0.001	6	34	915	17.6%
SUSAN	0	33	34	6	97.1%
	0.0001	33	34	6	97.1%

Table 4.15: Illustrates the number of true and false interest points detected and the accuracy of true interest points detected.

Table 4.15 shows how many true and false interest points are detected in the presence of Gaussian noise. Note that variances such as 0.0005, 0.001 and 0.005 for SUSAN, and 0.005 for SURF are not included here as these are not detecting any true interest points at all. Both SURF and SUSAN does a decent job when the variance is 0.0001 but decreases its performance the variance is increased. SUSAN goes from detecting 33 true interest points to not detecting any true interest points at all. SURF, on the other hand, manage to detect some true interest points. However, the number of false interest points obscures the true interest points. As for KAZE, there is a different distribution of true interest points detected. Compared to the other detectors, KAZE manages to detect some interest points at the other variances as well. Despite SUSAN being the only detector to detect the same number of true interest points as the image without any Gaussian noise, KAZE detectors did the best job overall as this detector was able to detect some true interest points through all variances. Thus, it can be concluded that the KAZE detector did the best job overall in terms of detecting true interest points throughout this experiment with different variances. The following Figure 4.24 shows the accuracy rate for the true interest points detected at different variances.

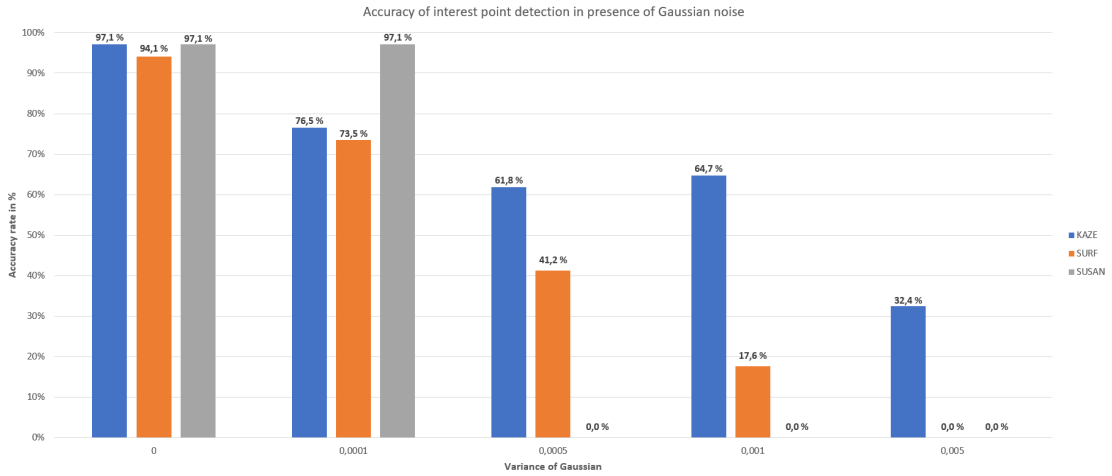


Figure 4.24: Accuracy of interest point detectors in presence of Gaussian noise with different variances

The detectors have less accuracy rate in all variances compared to the image without any Gaussian noise. Even though the decrease in performance, the KAZE detector does a better job overall compared to the other detectors when Gaussian noise is added. The reason for that is because KAZE makes use of nonlinear diffusion filtering, which is all about detecting point or region of interest at different scale levels in nonlinear scale space. Moreover, by comparing the different detectors in terms of average accuracy rate, the results are as following. KAZE has an average accuracy rate of 58.9%, while both SURF and SUSAN have an average rate of 33.1% and 24.3% respectively. Although KAZE detectors are better than the other detectors, the detector still missed many true interest points. From this experiment, it can be concluded that in terms of detecting interest points in the presence of Gaussian noise, the KAZE detector is a better choice than the other detectors even though the detector missed many true interest points.

4.1.8 Discussion about image 1

It should be noted that there were a couple of weaknesses in image 1.

One weakness was the issue when very high brightness was applied to image 1. The figures diamond and triangle were similar to the background. Thus made it difficult for both the KAZE and SURF detector to detect any interest points. This is due to the fact that these detectors are based on the determinant of Hessian (DoH), which is known for detecting corners where there are change in image gradient in two direction. Additionally, SUSAN also did not managed to detect any interest points in these figures because of the circular mask of SUSAN struggled to identify as these figures were too similar to the background as well, and therefore not regarded by the circular mask.

Another weakness was the sudden transition between background and the figures. A sudden transition like this is not very common in a real-life applications.

However, it should be noted that the sole purpose of a image like this is to test the detectors in terms of detecting interest points. Therefore a synthetic image of this type is well suited for this task as it verifies the simple aspect of how the detectors are detecting obvious interest points when different transformations are applied. Test of the detectors in synthetic image with different transformations is a step towards the use of these detectors in real-life applications.

4.2 Main experiment on image 2 and 3

This section presents the results for the performance of interest points detection on real life images. Real life images are useful for the testing the overall ability of the detectors to provide realistic interest points detection results under real life conditions. The purpose of this main experiment is to evaluate the performance of the detectors in terms of detecting interest points in real life images such as image 2 and 3. Since these images are different from the previous one, new property values for the detectors have to be chosen.

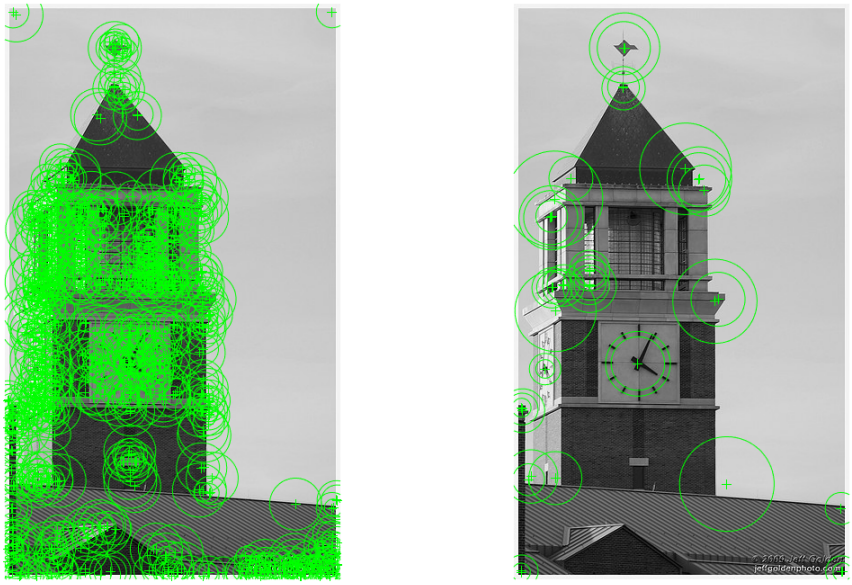
4.2.1 Property setup for the main experiment on image 2 and 3

The threshold, octaves and scale levels are set to appropriate values in order for accurate detection of true interest points. The KAZE detector utilizes a threshold value that has been tested in multiple image to give the best possible result. The most important characteristics of finding the right property values is to eliminate false interest points and promotes true interest points, which was considered when the threshold value was selected. The property values used for this experiment is as following:

KAZE Detector				
Detectors	*Threshold*	*Octaves*	*Scale levels*	*Diffusion*
KAZEdetect.m	0.01	3	4	'Region'

Table 4.16: Property values of the KAZE detector for experiment of real-life images

Table 4.16 shows the values used for this experiment. The octave is increased to 3 as these are high-resolution images and to detect extensive features like blobs and corners. The number of scale levels is kept at 4 as smoother scale changes are desired in this experiment as well. The threshold is decreased to 0.01 as this gave the best result in terms of detecting true interest points, simultaneously reduce false interest points. Figure 4.25 shows image 2 detected with the property values from the previous main experiments (Table 4.2) and the property values from this experiment (Table 4.16).



(a) Property values from the previous main experiments

(b) Property values for current main experiment

Figure 4.25: Shows images detected with different property values

There is a massively decrease in the number of false interest points. The detector goes from detecting 762 interest points to 41. Additionally, the detector also detects more large features using current property values than values from previous experiments.

SURF Detector			
Detectors	*MetricThreshold*	*Octaves*	*Scale levels*
SURF.m	5000	3	4

Table 4.17: Property values of the SURF detector for experiment of real-life images

The 'MetricThreshold' of the SURF detector, which determines the number of points to appear is set to 5000. This is done to decrease the false detected interest points. The high threshold value is equivalent to a few features, and vice versa. The following table shows the values used for SURF detector throughout this experiment. Similar to the previous experiment, both KAZE and SURF uses the same number of octaves (O) and scale levels (S). The reason for that is to be able to compare the performance of these detectors in terms of detecting interest points at the same octave and scale levels, even

though they are different from each other. Increased octave number in SURF means that larger blob features are being detected.

SUSAN Detector				
Detectors	*GeoThresh*	*GeoThresh 2*	*IntThresh*	*IntThresh 2*
SUSAN.m	7	12	0.2	0.01

Table 4.18: Property values of the SUSAN detector for experiment of real-life images

Unlike the other methods, SUSAN needs to be tested with different values to differentiate corners from the edge. Similar to the previous experiment, these values are manually assigned. Although, SUSAN needs to have different values for each image, the values from Table 4.18 are initially set to be used throughout this experiment.

4.2.2 Results of interest points detected in image 2

This experiment is carried out with the purpose of testing the KAZE detector to detect true interest points at buildings from a real image. KAZE is tested in comparison with SURF and SUSAN. The results obtained from KAZE, SURF, and SUSAN of the image are as following:

KAZE detector

The result obtained from using the KAZE detector is as following:

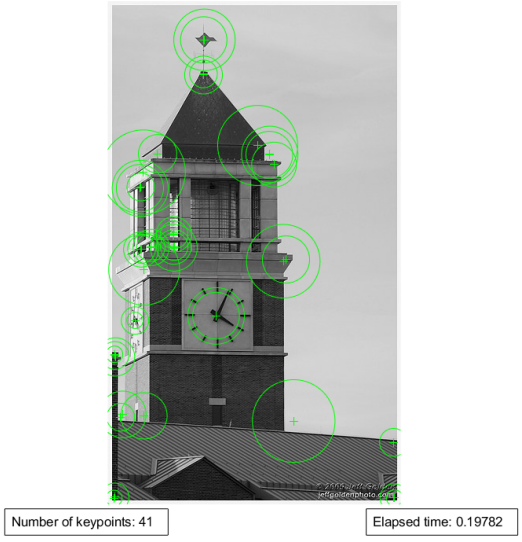
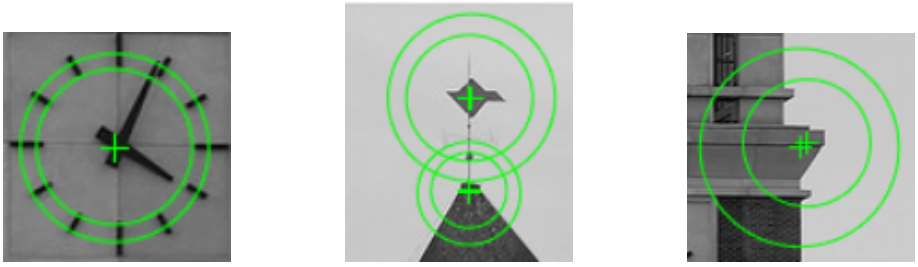


Figure 4.26: Image 2 detected with KAZE detector

The detection of true interest points of image 2 is quite impressive. By looking at the image, the detector is detecting true interest points such as corners, tower top, and center of both clocks. Additionally, it can be seen that the responses are mainly on corners and some on areas where there are a clear distinction between point area and background. Figure 4.27 shows some of the true interest points detected.



(a) Center of clock **(b)** Tower top **(c)** Corner

Figure 4.27: Results of true interest points in image 2

SURF detector

The result obtained from using the SURF detector is as following:

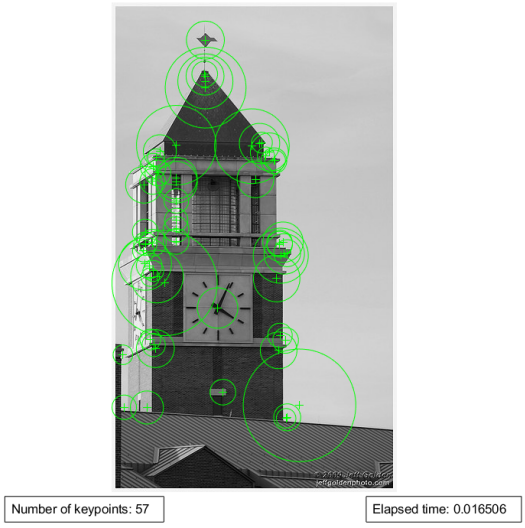
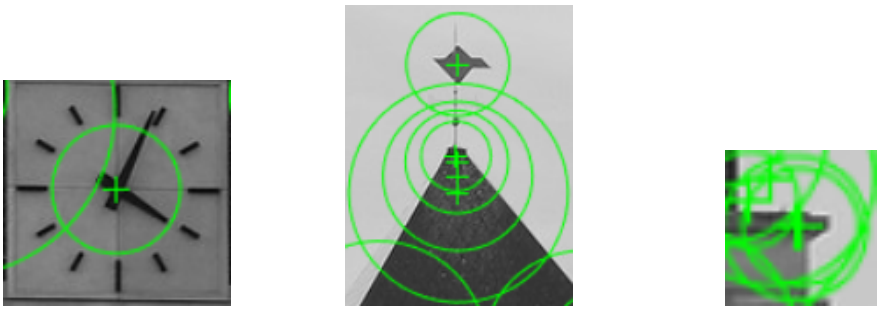


Figure 4.28: Image 2 detected with SURF detector

Similar to KAZE, SURF also detects true interest points at corners, tower top, and center of both clocks. The only difference is that SURF has a lot more interest points detected than KAZE, which is due to multiscale detection of same true interest point. Additionally, similarly to KAZE, responses are mainly on corners and some on areas where there are a clear distinction between point area and background as well. Figure 4.29 show some of the true interest points detected.



(a) Center of clock **(b)** Tower top **(c)** Corner

Figure 4.29: Results of true interest points in image 2

SUSAN detector

The result obtained from using the SUSAN detector is as following:

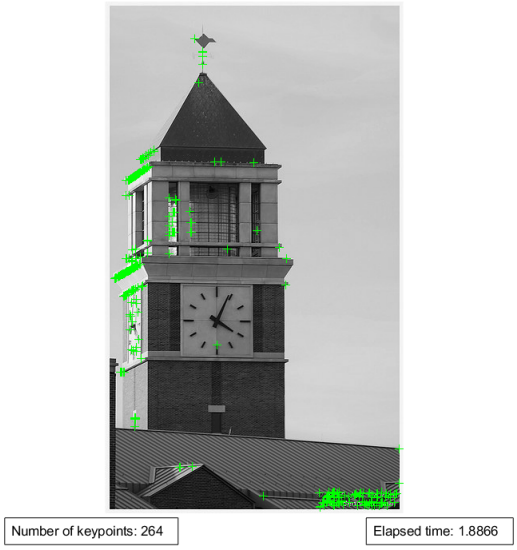


Figure 4.30: Image 2 detected with SUSAN detector

After tweaking with many SUSAN values, it can be seen that the best result for SUSAN is achieved by using the values from Table 4.18. SUSAN is not suited for this kind of experiment and manages to detect more false than true interest points, especially at the edge and bottom right corner. Additionally, it can be seen that SUSAN is detecting more edges than corner. Thus, it can be concluded that SUSAN struggles to detect true interest points in this specific real life image.

Comparison of the detectors in terms of detecting image 2

Table 4.19 shows the number of interest points detected, the time used for the whole process and time each interest points.

Detectors	*Number of interest points*	*Time [s]*	*Time each interest points [s]*
KAZE	41	0.19782	0.00482
SURF	57	0.016506	0.0002896
SUSAN	264	1.8866	0.00715

Table 4.19: Number of interest points found using KAZE, SURF and SUSAN and the time usage for these

In terms of detection speed, the SURF detector has the highest detection speed. Following up is KAZE with a decent detection speed as well. At last is the SUSAN detector having slowest detection speed. By comparing the detectors in terms of detecting true interest points, it can be seen that both KAZE and SUSAN have very good detection precision by detecting a lot of true interest points than false interest points. While SUSAN has the worst detection precision by detecting more false interest points than true interest points.

4.2.3 Results of interest points detected in image 3

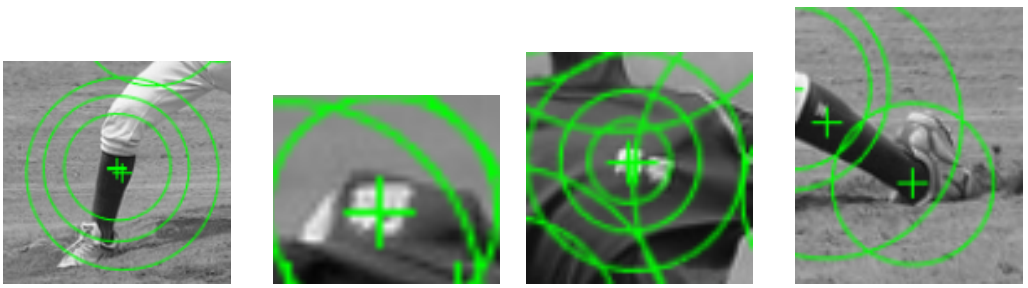
This experiment is carried out with the purpose of testing the KAZE detector to detect interest points on a person from a real image. KAZE is tested in comparison with SURF and SUSAN. The results obtained from KAZE, SURF, and SUSAN of the image are as following:

KAZE detector

The result obtained from using the KAZE detector is as following:

Figure 4.31: Image 3 detected with KAZE detector

The results are impressive in this experiment as well. The detector is capable of detecting many true interest points. In addition, there is no sign of false interest points as well. Similar to the previous experiment, it can be seen that interest points are detected in areas where there are a clear distinction between point area and background as well. Figure 4.32 shows some of the true interest points detected.



(a) Right leg **(b)** Hat **(c)** Jersey **(d)** Left leg

Figure 4.32: Results of true interest points in image 3

SURF detector

The result obtained from using the SURF detector is as following:

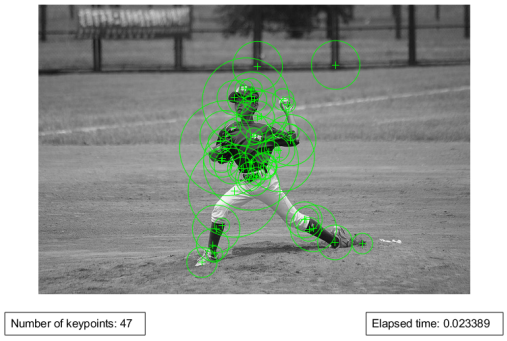


Figure 4.33: Image 3 detected with SURF detector

SURF detects many true interest points as well. However, it can be seen that there are one false interest points detected in the background. Similar to the KAZE detector, it can be seen that there are detection of areas where it is a clear distinction between point area and background. However, there are also many interest points overlapping each other which makes it difficult to distinguish them from each other. Figure 4.29 shows some of the true interest points detected and the false interest point from the background.

(a) Ball and hand **(b)** Emblem **(c)** Leg

Figure 4.34: Results of true interest points in image 3

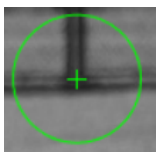


Figure 4.35: The false interest point in the background

SUSAN detector

The result obtained from using the SUSAN detector is as following:

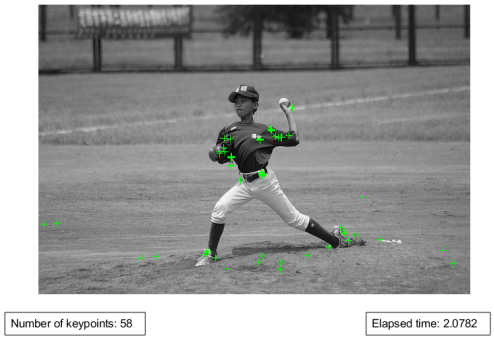


Figure 4.36: Image 3 detected with SUSAN detector

The SUSAN detector manages to detect some true interest points using the values from Table 4.18. However, the majority of detected interest points are false. Similar to the previous image, it can be concluded that SUSAN is not suited for this experiment either as there are lot more false interest points detected than true.

Comparison of the detectors in terms of detecting image 3

Table 4.19 shows the number of interest points detected, time used for the whole process and time each interest points.

Detectors	*Number of interest points*	*Time [s]*	*Time each interest points [s]*
KAZE	20	0.23569	0.01178
SURF	47	0.02339	0.000498
SUSAN	58	2.0782	0.03583

Table 4.20: Number of interest points found using KAZE, SURF and SUSAN and the time usage for these

In this image, the SURF detector still has the highest detection speed. Following up is KAZE and at last is the SUSAN detector. Both KAZE and SURF has a lot of true interest points detected. However, SURF had a false interest point which can be seen in the background. For the SUSAN detector, it is clear that from both this and the

previous experiment SUSAN somehow struggles on real-life images as the majority of the detected interest points are false.

Overall it can be concluded that both KAZE and SURF did a very good job in terms of detecting true interest points in real life images. SUSAN, on the other hand, had many false interest points. This is due to fact that SUSAN needs to adjusted according to the image experimented.

5. Conclusion

In this project, three interest point detectors have been tested for two types of images, a synthetic image with different geometrical shapes and real-life images. Based on the results received, it can be concluded that all detectors had their strengths and weaknesses detecting interest points in these images. In addition, the detectors also had a common weakness, where the threshold values and other parameters had to be adjusted according to image properties.

KAZE showed varying results throughout the experiment of the synthetic image with geometrical shapes. The detector surpassed the other detectors in terms of detecting most true interest points on image 1 with rotation and Gaussian noise. However, the detector was not that robust when it came to very high and low brightness, and up-scaled images, where the detector struggled to detect true interest points. In terms of real-life images, KAZE did a very good job by detecting many true points in both image 2 and 3. This detector was competitive in terms of detecting true interest points in many experiments, and had a decent detection speed.

SURF detector did an outstanding job at many experiments of the synthetic image, especially image 1 with different brightness changes. The detector did a better job of detecting true interest points when the image had very low, low, and high brightness changes than without any brightness. In terms of real-life image, SURF had some true interest points detected as well. The detector had most true interest points on image 3, but there were a lot of false interest points as well. Despite that, SURF did a decent job throughout a lot of the experiments and had the best detection speed compared to the other detectors.

SUSAN was very accurate, given that the correct threshold values are chosen. SUSAN had its best performance at the image with brightness changes and image without any transformations, where the detector did an outstanding performance. SUSAN also had some struggles as well, especially on the image with Gaussian noise, where it went from detecting almost all true interest points to none true interest points. SUSAN also did very bad in the experiment of real-life images and had the slowest detection speed compared to the other detectors.

5.1 Future work

As mentioned in the subsection of 4.1.3 *Comparison of the detectors in terms of images without any transformations*, there were many true interest points detected in the same "corner" or point area. This was unnecessary considering that only one true interest point is needed from each "corner" or point area. A solution to this is by making use of post processing of removing redundant true interest points and only keep one true interest point for each "corner" or point area. Thus, make it easier for distinguishing different interest points.

The precision of the detectors such as KAZE and SURF might be inaccurate in few of the experiments when detecting true interest points compared to SUSAN. Although most of the experiments are detected accurately as possible, there were some few experiments where the interest points were selected based on the green scale area rings rather than the middle cross of these rings. A possible future work, is to examine the precision of these detectors by comparing the centre points of these green rings with SUSAN in terms of determine the precision of the detectors.

Finally, an alternative approach that might be better to use instead of KAZE is AKAZE. AKAZE is a better alternative than KAZE in terms of detecting interest points as AKAZE increases both speed and performance by make use of Fast Explicit Diffusion (FED), which constructs the nonlinear scale space faster than using Additive Operator Splitting (AOS).

Bibliography

[1] M. Hassaballah, A. Abdelmgeid, and H. Alshazly, "Image features detection, description and matching," *Studies in Computational Intelligence*, vol. 630, pp. 11–45, Feb. 2016, visited on 2019-04-15. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-28854-3_2

[2] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer-Verlag Berlin, Heidelberg ©2010, 2010, visited on 2019-04-29.

[3] G. C. S. Linda G. Shapiro, *Computer Vision*. Prentice Hall PTR Upper Saddle River, NJ, USA ©2001, 2001, visited on 2019-04-29.

[4] Q. T. Yali Li, Shengjin Wang and X. Ding, "A survey of recent advances in visual feature detection," *Neurocomputing*, vol. 149, pp. 736–751, feb 2015, visited on 2019-04-29. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231214010121

[5] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: a survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, pp. 177–280, 2007, visited on 2019-04-29. [Online]. Available: http://homes.esat.kuleuven.be/~tuytelaa/FT_survey_interestpoints08.pdf

[6] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the Alvey Vision Conference*. Alvety Vision Club, 1988, pp. 23.1–23.6, doi:10.5244/C.2.23. [Online]. Available: http://www.bmva.org/bmvc/1988/avc-88-023.html

[7] Jianbo Shi and Tomasi, "Good features to track," *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, Jun. 1994, visited on 2019-02-04. [Online]. Available: http://ieeexplore.ieee.org/document/323794/

[8] S. Smith and J. Brady, "Susan – a new approach to low level image processing," Department of Engineering Science, Oxford University, Oxford, UK, Tech. Rep., 1995, visited on 2019-02-07. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.8270&rep=rep1&type=pdf

[9] Simultaneous localization and mapping. Visited on 2019-06-05. [Online]. Available: https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping

[10] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pp. 1508–1515, Okt. 2005, visited on 2019-02-07. [Online]. Available: https://ieeexplore.ieee.org/document/1544896

[11] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 105–109, Jan. 2010, visited on 2019-02-07. [Online]. Available: https://ieeexplore.ieee.org/document/4674368

[12] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 510–517, Jun. 2012, visited on 2019-02-09. [Online]. Available: https://ieeexplore.ieee.org/document/6247715

[13] D. Mistry and A. Banerjee, "Comparison of feature detection and matching approaches: Sift and surf," *GRD Journals- Global Research and Development Journal for Engineering*, vol. 2, no. 4, pp. 7–13, Mar. 2017, visited on 2019-02-09. [Online]. Available: https://pdfs.semanticscholar.org/5405/981f5ed9b16f1440476699793f09d874d123.pdf

[14] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004, visited on 2019-02-09. [Online]. Available: https://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94

[15] D. Jiang and J. Yi, "Comparison and study of classic feature point detection algorithm," *2012 International Conference on Computer Science and Service System*, pp. 2307–2309, Aug. 2012, visited on 2019-02-10. [Online]. Available: https://ieeexplore.ieee.org/document/6394890

[16] S. Tareen and Z. Saleem, "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk," *2018 International Conference on Computing, Mathematics and*

Engineering Technologies (iCoMET), pp. 1–10, Mar. 2018, visited on 2019-02-10. [Online]. Available: https://ieeexplore.ieee.org/document/8346440

[17] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, 2008, visited on 2019-02-18. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314207001555

[18] J. Heinly, E. Dunn, and J. Frahm, "Comparative Evaluation of Binary Features," *Computer Vision - ECCV 2006*, vol. 7573, pp. 759–773, 2012, visited on 2019-02-20. [Online]. Available: http://link.springer.com/10.1007/978-3-642-33709-3_54

[19] P. Alcantarilla, "Kaze features," *Computer Vision – ECCV 2012*, vol. 7577, pp. 214–227, 2012, visited on 2019-03-04. [Online]. Available: http://link.springer.com/10.1007/978-3-642-33783-3_16

[20] B. Ramkumar, "GPGPU, Acceleration of the KAZE Image Feature Extraction Algorithm," *Computer Vision and Pattern Recognition*, Jun. 2017, visited on 2019-04-09. [Online]. Available: http://arxiv.org/abs/1706.06750

[21] T. Lindeberg, "Feature detection with automatic scale selection," Department of Numerical Analysis and Computing Science, KTH (Royal Institute of Technology), Technical report ISRN KTH/NA/P–96/18–SE, 1998. [Online]. Available: https://people.kth.se/~tony/papers/cvap198.pdf

[22] P. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces," in *Procedings of the British Machine Vision Conference 2013*. Bristol: British Machine Vision Association, 2013, pp. 1–11. [Online]. Available: http://www.bmva.org/bmvc/2013/Papers/paper0013/index.html

[23] G. Gilboa, "Nonlinear Scale Space with Spatially Varying Stopping Time," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 12, pp. 2175–2187, Dec. 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4441717/

[24] Wikipedia. Hessian matrix. Visited on 2019-04-11. [Online]. Available: https://en.wikipedia.org/wiki/Hessian_matrix

[25] J. Weickert and H. Scharr, "A Scheme for Coherence-Enhancing Diffusion Filtering with Optimized Rotation Invariance," *Journal of Visual Communication and Image Representation*, vol. 13, no. 1-2, pp. 103–118, Mar. 2002, visited on 2019-04-10. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S104732030190495X

[26] P. F. Alcantarilla. (2014) Feature detection and description in non-linear scale spaces. Visited on 2019-04-10. [Online]. Available: http://cis.eecs.qmul.ac.uk/2014SummerSchool/PabloAlcantarilla_CIS_SummerSchool2014.pdf

[27] G. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2008, visited on 2019-04-28.

[28] F. Crow, "Summed-area tables for texture mapping," -, vol. 18, pp. 207–212, 1984.

[29] Badgerati. Computer vision – the integral image. Visited on 2019-04-11. [Online]. Available: https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/

[30] P.-T. Y. Zhanlong Yang, Dinggang Shen, "Image mosaicking using surf features of line segments," -, vol. -, pp. 207–212, 2017. [Online]. Available: https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0173627&type=printable

[31] A. Neubeck and L. V. Gool, "Efficient Non-Maximum Suppression," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, Aug. 2006, pp. 850–855.

[32] M. Brown and D. Lowe, "Invariant Features from Interest Point Groups," in *Procedings of the British Machine Vision Conference 2002*. Cardiff: British Machine Vision Association, 2002, pp. 1–10. [Online]. Available: http://www.bmva.org/bmvc/2002/papers/92/index.html

[33] C. Claus, "Optimizing the SUSAN corner detection algorithm for a high speed FPGA implementation," *2009 International Conference on Field Programmable Logic and Applications*, pp. 138–145, 2009.

[34] S. Smith and J. Brady, "Susan – a new approach to low level image processing," Department of Engineering Science, Oxford University, Oxford, UK, Technical Report TR95SMS1c, 1997. [Online]. Available: https://users.fmrib.ox.ac.uk/~steve/susan/susan/node12.html

[35] Mathworks. Point feature types. Visited on 2019-04-20. [Online]. Available: https://se.mathworks.com/help/vision/ug/point-feature-types.html

[36] P. Alcantarilla. kaze_features.cpp. Visited on 2019-05-23. [Online]. Available: https://github.com/pablofdezalc/kaze/tree/master/src

[37] K. Yan. Corner detection using susan operator. Visited on 2019-04-23. [Online]. Available: https://se.mathworks.com/matlabcentral/fileexchange/30789-corner-detection-using-susan-operator

[38] Common objects in context. Visited on 2019-06-03. [Online]. Available: http://cocodataset.org/#download

[39] Z. Pusztai and L. Hajder, "Quantitative Comparison of Affine Invariant Feature Matching," in *VISIGRAPP*, 2017.

[40] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, Jul. 1990, visited on 2019-03-14. [Online]. Available: https://ieeexplore.ieee.org/document/56205

[41] J. Weickert, *Anisotropic Diffusion in Image Processing*. ECMI Series, Teubner-Verlag, Stuttgart, Germany, 1998, visited on 2019-03-24.

[42] M. Wielgus. (2010) Perona-malik equation and its numerical properties. Visited on 2019-03-24. [Online]. Available: http://students.mimuw.edu.pl/~mwielgus/files/PM_equation.pdf

[43] J. Weickert, "Efficient image segmentation using partial differential equations and morphology," *Computer Vision – ECCV 2012*, vol. 34, no. 9, pp. 214–227, Sep. 2012, visited on 2019-03-20. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0031320300001096

[44] L. Feng, W. Zhuangzhi, and X. Long, "Fast image diffusion for feature detection and description," *International Journal of Computer Theory and Engineering*, vol. 8, pp. 58–62, Feb. 2016, visited on 2019-04-10. [Online]. Available: https://pdfs.semanticscholar.org/507f/5c17ce908c1924d09aa08211f40b4ab7df78.pdf

[45] J. Weickert, B. Romeny, and M. Viergever, "Efficient and reliable schemes for nonlinear diffusion filtering," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 398–410, Mar. 1998, visited on 2019-03-22. [Online]. Available: https://ieeexplore.ieee.org/document/661190

A. Nonlinear diffusion filtering

This chapter will discuss how to fully understand how KAZE features enhance detectors and the necessity to know what Nonlinear Diffusion Filtering is and how it is applied in KAZE detection. Introduced by Perona and Malik in 1990 [40], Nonlinear Diffusion Filtering (NDF) makes the blurriness locally adaptive to the image data, thus blurs noise but details such as edges will remain the same (see figure A.1) [26]. More specifically, it reduces the diffusion at edges without the trade off of loosing important image details like natural boundaries of objects using efficient schemes, namely AOS techniques and conductivity functions. The reader should note that in the absence of these schemes that it will lead to poor efficiency and limitations to practical use as NDF is only stable for limited, small time steps. The common difficulties in the application of NDF are associated with blurring and localization of problems. The solution is a nonlinear diffusion method (equation A.1) named "the Perona-Malik model", which utilize an inhomogeneous diffusivity process in image data where there are a large likelihood to be edges [41] [42]. The expression is written as following:

$$\frac{\partial L}{\partial t} = div(c(x,y,t) \cdot \nabla L) \tag{A.1}$$

where div is the divergence operator, ∇ is the gradient operator and L is the image brightness. $c(x,y,t)$ is the conductivity function and vital to make the diffusion adaptive to the local image structure. Time parameter t is the scale parameter where larger values lead to a simpler image representation [19]. Figure A.1 shows the difference between Gaussian filtering and NDF. Gaussian filtering blurs the image and remove both noise and detail in the image, whereas NDF makes the blurring locally adaptive to the image.

Figure A.1: Blur with Gaussian filtering (left). Blur with nonlinear diffusion filtering (right) [26]

A.1 The conductivity functions

In 1990 Perona and Malik proposed to make the function c dependent on the gradient magnitude [19]. The magnitude of the image gradient controls the diffusion at each scale level, thus the conduction function $c(x, y, t)$ A.2 is defined as

$$c(x, y, t) = g(|\nabla L_\sigma(x, y, t)|) \tag{A.2}$$

where ∇L_σ is the gradient of the Gaussian smoothed version of the original image L. The KAZE features make use of three different types of the function g, where g_1 promotes high-contrast edges, g_2 promotes wide regions over smaller ones, and g_3 smoothes the internal area and retain boundary information. Perona and Malik defined g_1 and g_2 in *Scale-space and edge detection using anisotropic diffusion* [40] as:

$$g_1 = exp(-\frac{|\nabla L_\sigma|^2}{k^2}), \qquad g_2 = \frac{1}{1 + \frac{|\nabla L_\sigma|^2}{k^2}} \tag{A.3}$$

where the contrast factor k determines which edges to be corrected using smoothing. This factor is empirical estimated in KAZE features to be 70% of a gradient histogram calculated from a smoothed version of the original image. Note that the value can also

be calculated from the image-gradient estimates if necessary [19]. Figure A.2 illustrates the the conductivity coefficient g_1 and g_2 in the Perona and Malik equation as a function of the contrast factor parameter k. By observing the figure, it can be seen that only high gradients are preserved when the contrast factor parameter k is increased. This implies to that low k preserves most of the gradients and high k only preserves the strongest gradients.

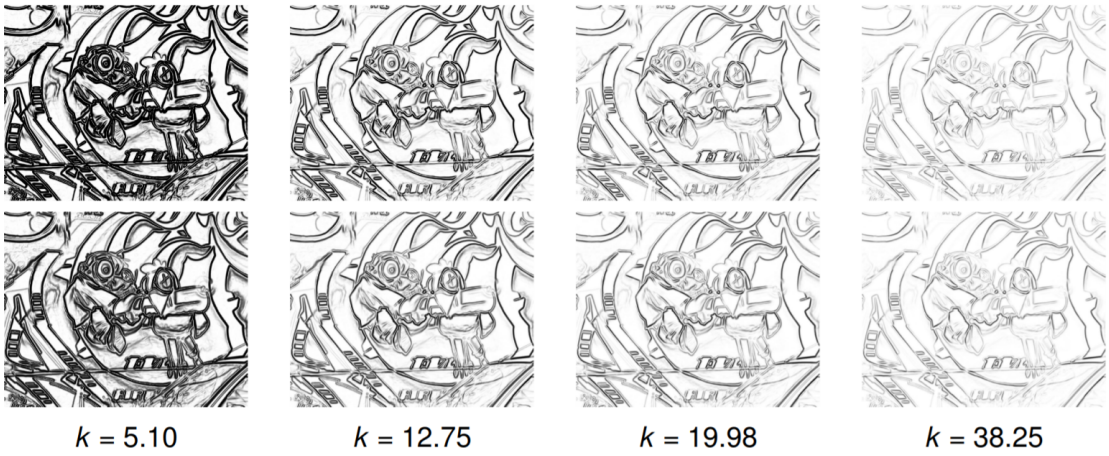


Figure A.2: First row: The conductivity coefficient g_1 using different values for the contrast factor parameter k Second row: The conductivity coefficient g_2 using different values for the contrast factor parameter k [26]

The last conductivity function is referred to the J. Weickerts diffusion function for rapidly decreasing diffusivities, slightly different from the g_1 and g_2 functions. Defined as g_3 (A.4), this function favors intraregional smoothing to interregional blurring, where smoothing on both sides of an edge is much stronger than across it [43].

$$g_3 = \begin{cases} 1, & |\nabla L_\sigma|^2 = 0 \\ 1 - exp(-\frac{3.315}{(|\nabla L_\sigma|^2/k)^8}), & |\nabla L_\sigma|^2 > 0 \end{cases} \tag{A.4}$$

B. AOS schemes

As PDE's inclusion in nonlinear diffusion filtering return no analytical solutions, it is imperative to utilize numerical methods to obtain iterative solutions [19]. The methods are calculated by approximating the differential equations similar to equation A.1. There are several possible approaches in diffusion equations in terms of discretization using various techniques such as *the semi-implicit scheme* or *the explicit scheme*. A common preference is to use *the explicit scheme* due to its ability to be utilized using a direct implementation. However, it does come with a major drawback being computationally intense as a result of the required number of iterations necessary to reach a desirable scale level following stability issues.

An alternative method is utilizing *the semi-implicit scheme* with *the AOS scheme*. This method provides stable solutions at any step size with efficient performance, additionally creating nonlinear scale spaces for description problems and feature detection [22]. The discretization of equation A.1 using *the AOS scheme* is expressed as following:

$$\frac{L^{i+1} - l^i}{\tau} = \sum_{l=1}^{m} A_l(L^i) L^{i+1} \tag{B.1}$$

where A_l represents a matrix that encodes the image conductivities for each dimension and τ is a constant time step in order to respect stability conditions [44]. The AOS schemes produces the nonlinear scale space iteratively using the solution L^{i+1}, in which I is the identity matrix. This solution can be expressed as following:

$$L^{i+1} = \left(I - \tau \sum_{l=1}^{m} A_l(L^i) \right)^{-1} L^i \tag{B.2}$$

According to the article *Efficient and reliable schemes for nonlinear diffusion filtering* [45], the performance efficiency is the result of utilizing *the Thomas Algorithm*. This variation of the Gaussian elimination algorithm allow *the semi-implicit scheme* to efficiently solve a linear system of equations. Note that the system matrix is tridiagonal and diagonally dominant.

C. Source code

A overview of the code used in this project can be found here.

- **Main.m**
- **KAZEdetect.m**
- **SURF.m**
- **SUSAN.m**

C.1 Main.m

```
% Author: Niroshan Thangalingam
% Date: 13.06.2019
close all;
clear;
clc;
%% Interest point detection
%img = imread('eksperiment1/shapes.png');
%KAZEdetect(img);
%SURF(img);
%SUSAN(img);
```

C.2 KAZEdetect.m

```
function [] = KAZEdetect(image)
if size(image, 3)==3
    image_gray = rgb2gray(image);% Converting image from rgb
        to grayscale
else
    image_gray = image; %If image is already in grayscale
end
kaze_img = uint8(255*mat2gray(image_gray)); %Normalising
   gray values in image

tic %Starting the timer
points = detectKAZEFeatures(kaze_img,'Diffusion','region','
   threshold',0.01,'NumOctaves', 3, 'NumScaleLevels', 4); %
   KAZEdetector
t=toc; %Ending the timer
num_points = length(points); %Finding number of stored
   points
strongestPoints = points.selectStrongest(num_points); %
   Select strongest points

%plotting the feature detector points and displaying the
   time and number
%of keypoints
figure;
imshow(image_gray);
hold on;
plot(strongestPoints);
annotation('textbox', [0.05, 0, 0.1, 0.1], 'String', "Number
    of keypoints: "+ strongestPoints.Count)
annotation('textbox', [0.7, 0, 0.1, 0.1], 'String', "Elapsed
    time: " + t)
end
```

C.3 SURF.m

```
function [] = SURF(image)
if size(image, 3)==3
    image_gray = rgb2gray(image);% Converting image from rgb
        to grayscale
else
    image_gray = image; %If image is already in grayscale
end
tic %Starting the timer
points = detectSURFFeatures(image_gray,'MetricThreshold'
   ,5000,'NumOctaves', 3, 'NumScaleLevels', 4); %SURF
   detector
t=toc; %Ending the timer
num_points = length(points); %Finding number of stored
   points
strongestPoints = points.selectStrongest(num_points); %
   Select strongest points

%plotting the feature detector points and displaying the
   time and number
%of keypoints
figure;
imshow(image_gray);
hold on;
plot(strongestPoints);
annotation('textbox', [0.05, 0, 0.1, 0.1], 'String', "Number
    of keypoints: "+ strongestPoints.Count)
annotation('textbox', [0.7, 0, 0.1, 0.1], 'String', "Elapsed
    time: " + t)
end
```

C.4 SUSAN.m

```
function [] = SUSAN(image)
%SUSAN corner detection inspired by
%https://se.mathworks.com/matlabcentral/fileexchange/30789-corner-detection-using-susan-operator
%Modified by Niroshan Thangalingam
if size(image, 3)==3
    image_gray = rgb2gray(im2double(image));% Converting image from rgb to grayscale
else
    image_gray = im2double(image); %If image is already in grayscale
end

tic; % Starting the timer
map = nlfilter(image_gray,[7 7],@func);
[r,c] = find(map);
t = toc; %Ending the timer
%plotting the feature detector points and displaying the time and number
%of keypoints
figure;
imshow(image_gray);
hold on;
plot(c,r,'g+')
annotation('textbox', [0.05, 0, 0.1, 0.1], 'String', "Number of keypoints: "+ length(c))
annotation('textbox', [0.7, 0, 0.1, 0.1], 'String', "Elapsed time: " + t)
end

function USAN = func(img)
% SUSANFUN  Determine if the center of the image patch IMG
%       is corner(res = 1) or not(res = 0)
mask = [...
```

```
        0 0 1 1 1 0 0
        0 1 1 1 1 1 0
        1 1 1 1 1 1 1
        1 1 1 1 1 1 1
        1 1 1 1 1 1 1
        0 1 1 1 1 1 0
        0 0 1 1 1 0 0];
% Geometric threshold for seperating corners and edges
thresholdGeo = 7;
thresholdGeo2 = 12;

% Intensity threshold
threshold_t = 0.2;
threshold_t2 = 0.01;

image_size = size(img,1);
nucleus = ones(image_size)*img(round(image_size/2),round(
   image_size/2));

similar = (abs(img-nucleus)<threshold_t).*mask;
USAN = sum(similar(:));

if USAN < thresholdGeo
        dark = nnz((img-nucleus<-threshold_t2).*mask);
        bright = nnz((img-nucleus>threshold_t2).*mask);
        USAN = min(dark,bright)<thresholdGeo2 && max(dark,
           bright)>thresholdGeo2;
else
        USAN = 0;
end
end
```