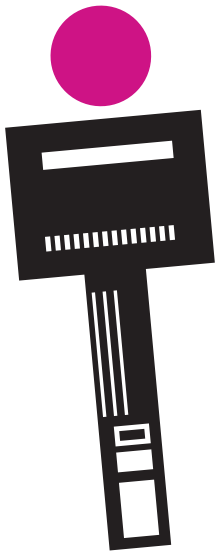
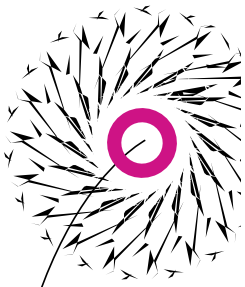




RAM ROBOTICS AND MECHATRONICS

COVARIANCE MODEL BASED KEYPOINT DETECTOR DEVELOPMENT

K. (Kevin Rafael) Indrawijaya

MSC ASSIGNMENT

Committee:
dr. ir. F. van der Heijden
dr. F.J. Siepel, MSc
prof. dr. ir. R.N.J. Veldhuis

July 2021

039RaM2021
Robotics and Mechatronics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Summary

Model-based approaches for image reconstruction, analysis and interpretation have been very popular in the past. However, in the last decade, with the availability of large amounts of imaging data and the increase in computing power, machine learning, in particular deep learning, has become more popular. Within a deep learning network, in standard descriptions of convolutional neural networks, the first layer is considered as an image feature extractor. Often, it is assumed that they extract features like edge, line, and spots. Despite this assumption, the kernels of these networks usually do not really resemble an edge, line, and spots detection. Therefore, the optimality of these layers can be questioned.

The model-based and data-driven approaches are not perfect; each has their strength and shortcomings. This thesis revisits a novel statistical approach to keypoint, specifically line and edge detection, using the covariance model based image feature detector. This research explores the implementation of covariance model-based image feature detection and any of its intermediate results in combination with deep learning approaches. The result shows that incorporating such prior information is useful in making more robust, non-spurious detections. Moreover, in the exploration of the CVM operator, a feature descriptor was designed that makes use of the CVM convolution kernels. This descriptor shows a rotation and limited scale invariant capability, which has shown to be useful for keypoint matching.

Contents

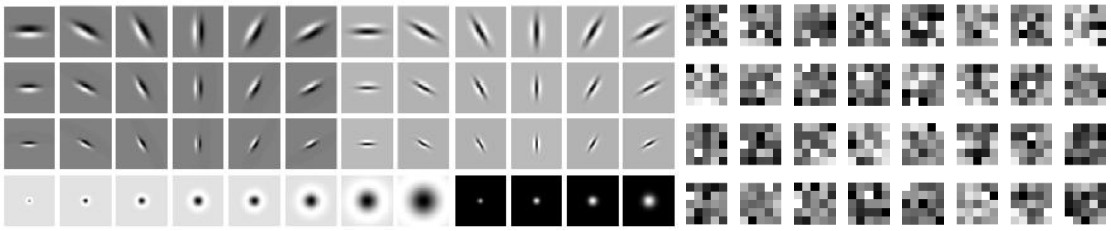
1 Introduction

Model-based approaches for image reconstruction, analysis and interpretation have been very popular in the past. Many of these approaches are based on mathematical, physical, or biological models. The challenge for the model-based approaches is the modelling of the problem with an appropriate level of details. However, in the last decade, with the availability of large amounts of imaging data and the increase in computing power, machine learning, in particular deep learning, has become more popular (Rueckert and Schnabel, 2019). Data-driven approaches have become more widespread for use in different tasks of reconstruction, analysis and interpretation.

A big part of image processing comes in image feature detection. It is used for semantic segmentation of the image and keypoint detection. Depending on the type of detection, these could be line, edge, corner, or points of interest. A found point of interests is often used further for the localisation of descriptors. The use of descriptors is often much related to keypoint matching, an operation that is the base for image registration, object tracking, and a lot more (Hassaballah et al., 2016).

Commonly, a keypoint detector has a raw image input. Some known model-based keypoint detectors are SIFT, SURF, FAST, BRISK, and KAZE (Lowe, 1999; Bay et al., 2006; Trajković and Hedley, 1998; Leutenegger et al., 2011; Alcantarilla et al., 2012). An alternative method that has been popular as of late is deep learning network trained for image segmentation, such as FCN, VGG16, and UNet (Long et al., 2014; Simonyan and Zisserman, 2015; Ronneberger et al., 2015). The model-based and data-driven approaches are not perfect; each has their strength and shortcomings. However, the shortcomings of one could possibly be overcome when used in conjunction with the other.

Within a deep learning network, in standard descriptions of convolutional neural networks, the first layer is considered as an image feature extractor (Lecun et al., 2015). This is similar to the concept of a front-end vision, a concept where the first image captured through the retina is the edge or line elements (ter Haar Romeny, 2003). Despite these assumptions, the kernels of these networks usually do not resemble an edge or line detection. An example of model-based kernels for edge and line detector is given in Figure 1.1. For comparison, the figure also shows the kernels of a first layer (front-end vision) of a semantic segmentation network. As a network is considered optimal after training, the effects of such kernels are questioned. Would a model-based filter in the front end vision have a better performance in comparison?



(a) Leung-Malik filter bank **(b)** Deep Learning first convolutional layer

Figure 1.1: Comparison of model-based filter and deep learning front end vision

This thesis revisits a novel statistical approach to image feature detection, specifically line and edge detection, using the covariance model, which was proposed by van der Heijden (1992). This research will explore the further implementation of covariance model-based feature detection and any of its intermediate result in combination with deep learning approaches.

Moreover, in the extension of exploring the covariance model, the possibility of making a descriptor out of the covariance model will also be explored.

1.1 Related Work

Rueckert and Schnabel (2019) did a survey on model-based and data-driven approaches in medical image computing. In the survey findings, the deep learning approaches are heavily emphasised to have revolutionised the field of medical image computing. This is especially so as more image databases are becoming available. However, it is often hard to generalise beyond the training data, and this generalisation is difficult to predict. Even so, deep learning can be very versatile, and often the same neural network architecture can be trained for different purposes. It mentions that despite this recent success of data-driven approaches, it is likely that the combination of data-driven and model-based approaches will be needed to have even better results. This is because the combination of both approaches is likely to advance in the challenges of generalisability, explainability, and data efficiency.

In the survey that looks into both model-based and data-driven approaches of deraining by Yang et al. (2019), data-driven methods generally perform better than the model-based methods. However, the paper still mentioned a few problems, specifically in the data-driven method. The data-driven methods are lacking in fusing physics models and real rain images, an accurate rain model, and the real application of deraining itself still needs further developments.

Therefore, the idea of combining model-based and data-driven approach is not new. This has been proposed before, and yet it is still in the exploratory stage. Andrade-Loarca et al. (2020) have previously used shearlet as the model-based methods and combine it with their own design convolutional neural network for semantic edge detection. The result of the paper shows an improvement in the performance of edge extraction and classification when using the shearlet transform for model-based feature extraction to feed the network. However, the paper also mentions that explorations in this combination of approaches are not in-depth yet, and it will likely be fruitful in the coming years.

The approach of Andrade-Loarca et al. (2020) is similar in idea to Clough et al. (2019) who use the concept of persistent homology to obtain topological characteristics of segmentation results. This is differentiable with respect to the pixelwise probability of being assigned to a given class. This acts as additional prior knowledge to the network, information that is often needed when dealing with small amounts of labelled data. The result shows that it improves segmentation performance in terms of topological correctness without sacrificing pixelwise accuracy.

Lee et al. (2019) introduce the concept of template transformer networks. This network takes the underlying structure of interest and deforms it. This is then incorporated into the pixelwise binary classification network, in this case, UNet. The template shape is given as an additional input channel, which acts as prior shape enforcement for the network where it will be incorporated. According to the paper, this approach reduces the number of false positives.

An interesting implementation of both model-based and data-driven approach was done by Oktay et al. (2018) to get super-resolution reconstruction. The approach was to incorporate prior anatomical knowledge into CNNs to do cardiac image enhancement and segmentation. The resulting super-resolution image reconstruction has better robustness. This is especially the case where the images are corrupted or contain artefacts.

Another approach was done by Uzunova et al. (2017) who made use of the known model to synthesise realistic ground truth training data for CNN based image registration. The paper also shows that the network learnt from the generative model outperform the one trained on classical image registration.

Jung and Sundström (2017) have a paper on a combined data-driven and model-based residual selection algorithm for fault detection and isolation. This is not specifically for images. However, it still holds significant conclusions. In the paper's findings, the importance of good training data is repeatedly stressed, and finding a suitable set is crucial to achieving satisfactory performance. For this particular case, this was achieved by generating the data-driven training data using the model-based approach.

In conclusion, data-driven methods benefit from having more information. This is especially the case for prior information that model-based methods can supply. This combination allows for a more robust network training, which shows better results in most cases compared to when the training is done on its own.

1.2 Research Question

There are two main research questions corresponding to the main goal and the secondary goal. These two research questions are:

1. To what extent can a data-driven approach, which is fed by information derived from a model-based approach, outperform the purely data-driven approach or purely model-driven approach?
2. In a feasibility study, to what extent are the features that are intermediate results of the CVM operator usable as a descriptor?

The proposed hypothesis is that the incorporation of a covariance model-based method will improve the performance of a data-driven method of feature detection. This hypothesis will be investigated by means of several experiments. There are several research sub-questions that can be answered which leads to this hypothesis.

Within a 1-dimensional model, there are already several aspects that will be addressed. The CVM operator is a model-driven approach to detect features within a record of a signal/image. The CVM operator is built with some input parameters (density, inhibition distance, etc.), which are called the design parameters. The CVM operator is tested with signals which are generated using the same model on which the CVM operator was designed. The values of these test parameters are not necessarily the same as the one which were used to build the CVM operator.

(i.a) Given the set of design parameters, which equals the set of test parameters, to what extent is it advantageous to numerically optimise the weights of the designed CVM operator?

(i.b) Given a set of test parameters with which test signals are generated, to what extent is it advantageous to numerically optimise the design parameters of the CVM operator?

(i.c) To what extent can a data-driven approach, such as a convolution neural network, outperform the model-driven approach?

(i.d) To what extent can a data-driven approach fed by signals derived from a model-based approach outperform the purely data-driven approach or purely model-driven approach?

In the 2-dimensional case, the CVM operator should be able to detect the spots, line, and edge elements.

(ii.a) How is the performance of the covariance model feature detector on spots, line, and edge detection compared to the literature?

(ii.b) To what extent can a data-driven approach, such as a convolution neural network, outperform the model-driven approach?

(ii.c) To what extent can a data-driven approach, which is fed by information derived from a model-based approach, outperform the purely data-driven approach or purely model-driven approach?

In addition to the experiment done on combining the covariance model with the deep learning method, further experimentation on covariance model output will be done. Keypoints and their descriptors are often used for geometrically associating one image with another. A displacement field is needed to describe the geometrical transform between the images. This geometrical transform can be a rigid affine transform or a projective transform.

(iii.a) To what extent are the features that are intermediate results of the CVM operator usable as a descriptor?

(iii.b) Can these descriptors be made rotational invariant?

(iii.c) Can these descriptors be made scale invariant?

(iii.d) To what extent can the features/descriptors of the CVM operator be used for keypoint matching?

1.3 Organisation of the Thesis

As there are several topics on the research questions, the thesis will be divided into several chapters. Each of the chapters will discuss a sub-topic containing one or more of the research questions. This will be started with some backgrounds. Then, the next few chapters are a series of experiments that leads to the main goal of exploring the CVM operator incorporation into a deep learning framework. Afterwards, the CVM feature descriptor is explored. This thesis will finally be closed with a showcase of the results and a general conclusion.

Chapter 2 shows the theoretical background of the covariance model-based keypoint detection.

Chapter 3 looks into Research Questions (i.a) and (i.b), where numerical optimisation in the 1-dimensional cases are explored.

Chapter 4 looks into Research Questions (i.c) and (1.d), discussing 1-dimensional data-driven feature detector performance in comparison to the model-based approach.

Chapter 5 looks into Research Questions (ii.a) and (ii.b), the extension of the covariance model into 2-dimensional feature detector.

Chapter 6 looks into Research Questions (ii.c) and (ii.d), discussing 2-dimensional data-driven feature detector performance in comparison to the model-based approach.

Chapter 7 looks into Research Questions (iii.a), (iii.b), (iii.c), and (iii.d), exploring the possibility of using the CVM features as a descriptor, along with the combination of keypoint matching.

Chapter 8 draws the general conclusion from this thesis. The hypothesis and the main goals will be re-addressed and discussed in this chapter.

2 Background

This section will go into the details on the theoretical background behind the work of this thesis. The key point detector in this section will be based on the covariance model as proposed by van der Heijden (1992). The proposed value of using this covariance model is that using conditional covariance matrices allows for detection and localisation of image discontinuities. This development will follow along with the development from 1-dimensional to a 2-dimensional case. The development of this key point detector starts with the detection of step and pulse in a 1-dimensional case.

2.1 1-dimensional Covariance Model Keypoint Detection

This section starts by introducing the waveform model, an inhibited generalised Poisson point process. This is a process in which events occur in a defined space. Events appear as a sequence of point $\xi_k \in \mathbb{R}$. The process is called inhibited if it is known that the distance between any pair of events is larger than a so-called inhibition distance. That is $|\xi_k - \xi_l| > r_i$ for any pair $k \neq l$. With each event k, an amplitude a_k is associated.

It is assumed that a_k has a zero mean and standard deviation σ_a. The process is observed by means of a point spread function $s(x)$. That is, an event ξ_k is observed as $a_k s(x - \xi_k)$. All functions, taken together, form shot noise. The whole composition is observed as a waveform,

$$w(x) = \sum_k a_k s(x - \xi_k) + b + n(x) \tag{2.1}$$

where $n(x)$ is observation noise, and b is an unknown offset. An example of a point spread function and the waveform model can be seen in Figure 2.1.

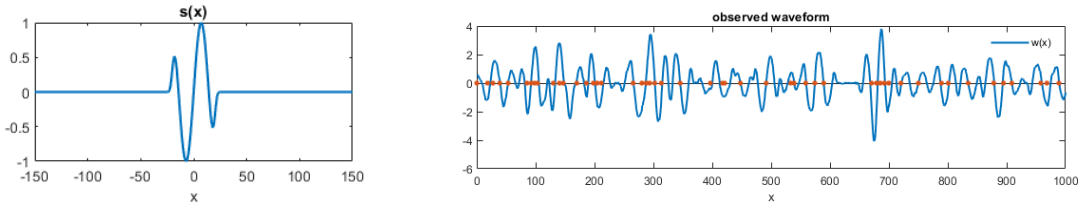


Figure 2.1: PSF and waveform of the model

The method to find the position starts with finding the autocovariance functions. This autocovariance function was derived in the book of Papoulis et al. (2002). The point process is assumed to have zero mean and is therefore described by the following autocovariance function

$$R_w(x_1, x_2) = R_p(x_1, x_2) + \sigma_b^2 + R_n(u) \tag{2.2}$$

where $u = x_2 - x_1$. $R_p()$ is associated with the point process. σ_b^2 reflects the unknown background b. The autocovariance function of the noise $n(x)$ is then given by $R_n(u)$.

The point process can then be defined as

$$R_p(x_1, x_2) = \sigma_a^2 \int_{\alpha=-\infty}^{\infty} \lambda(\alpha) s(x_1 + \alpha) s(x_2 + \alpha) d\alpha \tag{2.3}$$

If the density is constant, $\lambda(x) = \lambda_0$, this simplifies to:

$$\begin{aligned} R_p(x_1, x_2) &= \sigma_a^2 \lambda_0 \int_{\alpha=-\infty}^{\infty} s(x_1 + \alpha) s(x_2 + \alpha) d\alpha \\ &= \sigma_a^2 \lambda_0 \int_{\alpha=-\infty}^{\infty} s(x_1 - x_2 + \alpha) s(\alpha) d\alpha \end{aligned} \tag{2.4}$$

This leads to the autocovariance function for the case where density is constant, except for an interval in which it is zero:

$$R_p(x_1,x_2|\neg X) = \sigma_a^2\lambda_0\int_{\alpha=-\infty}^{\infty} s(x_1-x_2+\alpha)s(\alpha)d\alpha - \sigma_a^2\lambda_0\int_{\alpha=-X}^{X} s(x_1+\alpha)s(x_2+\alpha)d\alpha \tag{2.5}$$

The autocovariance itself is then separated into two cases. The first case, the null hypothesis, is for the case of no events within a small interval. The other case, the alternative hypothesis, is for the case of an event within a certain interval. The probability that there would be two or more events in a small interval is assumed to be zero due to the inhibition. This results in the following equation, where the visualisation can be seen in 2.2.

$$\begin{aligned} R_0(x_1,x_2) &= R_p(x_1,x_2|\neg\frac{1}{2}\Delta) + \sigma_b^2 + R_n(x_1-x_2) \\ R_1(x_1,x_2) &= R_s(x_1,x_2) + R_p(x_1,x_2|\neg r_i) + \sigma_b^2 + R_n(x_1-x_2) \end{aligned} \tag{2.6}$$

Where $R_s(x_1,x_2|\xi) = \sigma_a^2 s(x_1-\xi)s(x_2-\xi)$ and therefore $R_s(x_1,x_2) = \sigma_a^2 E_\xi[s(x_1-\xi)s(x_2-\xi)]$

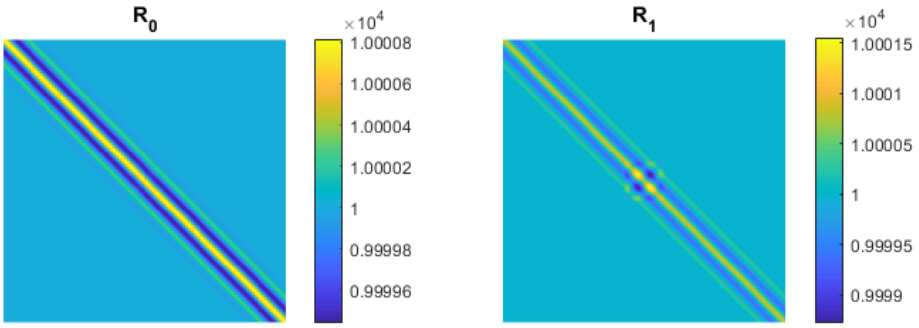


Figure 2.2: Covariance of the model

As it is a discrete model and not a continuous variable, the waveform is observed at a number of $N = 2K+1$ discrete positions $x_k = k\Delta$, called samples: $w_k = w(x_k)$ with $k = -K,\cdots,+K$. This defines an N-dimensional vector **w**. Therefore the covariance matrices of the discrete model can then be described as

$$\begin{aligned} \mathbf{R}_0(k,\ell) &= R_0(k\Delta,\ell\Delta) \\ \mathbf{R}_1(k,\ell) &= R_1(k\Delta,\ell\Delta) \end{aligned} \tag{2.7}$$

To test the hypotheses whether the pixel at $x = 0$ contains a point, ideally, the posterior probabilities under the two hypotheses are compared. However, this can also be done using the log-likelihood, which under Gaussian assumption is a sufficient statistic as it contains all the information of **w**. This can be seen in the following equation, where T is a threshold of a condensed constant parameter.

$$v(\mathbf{w}) \overset{>}{<} T \text{ with: } v(\mathbf{w}) = \mathbf{w}^T(\mathbf{R}_0^{-1} - \mathbf{R}_1^{-1})\mathbf{w} \tag{2.8}$$

A further simplification is introduced by application of a simultaneous decorrelation technique, applied to the vector **w** using the two covariance matrices. After this simultaneous decorrelation, the log-likelihood ratio becomes:

$$v(\mathbf{w}) = \mathbf{u}^T \left(\mathbf{I} - \Gamma^{-1}\right)\mathbf{u} \quad \text{with} \quad \mathbf{u} = \mathbf{T}\mathbf{w} \tag{2.9}$$

Elementwise, the equation can be noted down as

$$\mathbf{v}(\mathbf{w}) = \sum_{n=1}^{N} \left(1 - \frac{1}{\gamma_n}\right) u_n^2 \quad \text{where } \mathbf{u}^T = \begin{bmatrix} u_1 & \cdots & u_N \end{bmatrix} \tag{2.10}$$

where γ_n is the n-th diagonal element in the matrix Γ. The elements u_n are features derived from **w**.

The effectiveness of these features can then be quantified by Bhattacharyya distance. Following this argument, the elements with no or negligible contribution can therefore be discarded. The calculation of Bhattacharyya distance and the visualisation of an example of this can be seen in Equation 2.11 and Figure 2.3.

$$J_b(N) = \sum_{n=1}^{N} J_n \qquad \text{with:} \qquad J_n = \frac{1}{2} log \frac{1}{2}\left(\sqrt{\gamma_n} + \frac{1}{\sqrt{\gamma_n}}\right) \tag{2.11}$$

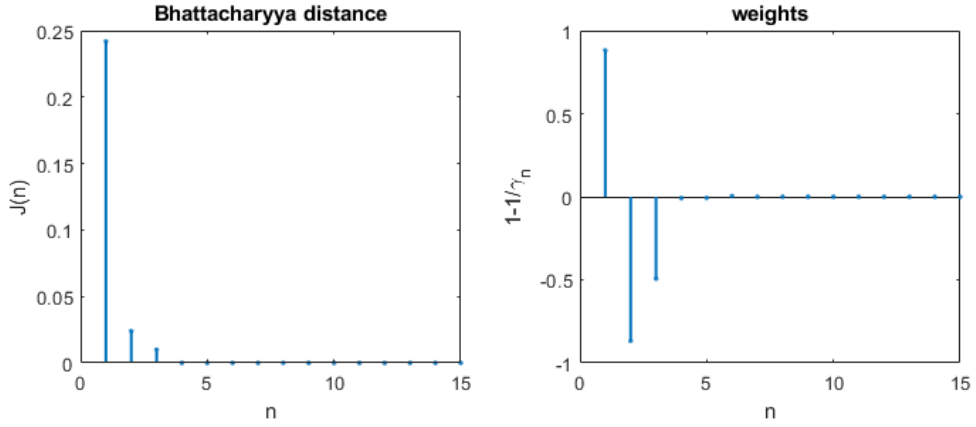


Figure 2.3: Distance and Weight of features

In the previous sections, it was assumed that the continuous waveform is observed as a finite sequence consisting of $N = 2K + 1$ samples that are enumerated either by $n = 1, \cdots, N$, or by $k = -K, \cdots, +K$. The hypothesis was only tested on the middlemost sample at k=0, or n=K+1. The procedure can be made applicable to all sample by modifying the $D \times N$ matrix **T** and the $D \times D$ diagonal matrix Γ. The transformation of the finite sequence **w** can then be rewritten by:

$$u_d = \sum_{n=1}^{N} \mathbf{T}_{d,n}\mathbf{w}_n \quad \text{with} \quad d = 1, \cdots, D \tag{2.12}$$

If **w** represents an infinite sequence, then the feature extraction is applied to finite selections of the sequence

$$u_d(m) = \sum_{n=1}^{N} \mathbf{T}_{d,n}\mathbf{w}_{m-K+n-1} \tag{2.13}$$

Up to now, the discussion about the position ξ of a point was restricted to only the middlemost pixel within a finite sequence. However, in a Poisson point process, many events are generated, and we want to detect them all. If at a certain sample m, the log-likelihood exceeds the

threshold, $v(m) > T$, then this is not a sufficient condition for detection since a neighbouring sample might have an even larger log-likelihood ratio. Hence, non-local-maximum suppression must be applied. A detection only takes place if the $v(m)$ exceeds the threshold and is larger than its two neighbours

$$\text{a point is detected at } m \text{ if}: \quad v_m > T \quad \text{and} \quad v_{m-1} < v_m \text{ and } v_{m+1} < v_m \tag{2.14}$$

Figure 2.4 shows the log-likelihood ratio in comparison to the waveform. It also includes the detected points in comparison to the true reference points. The error rate in this case is 12%, where there is 6 missed detection out of 50, where all 6 of them are spurious.

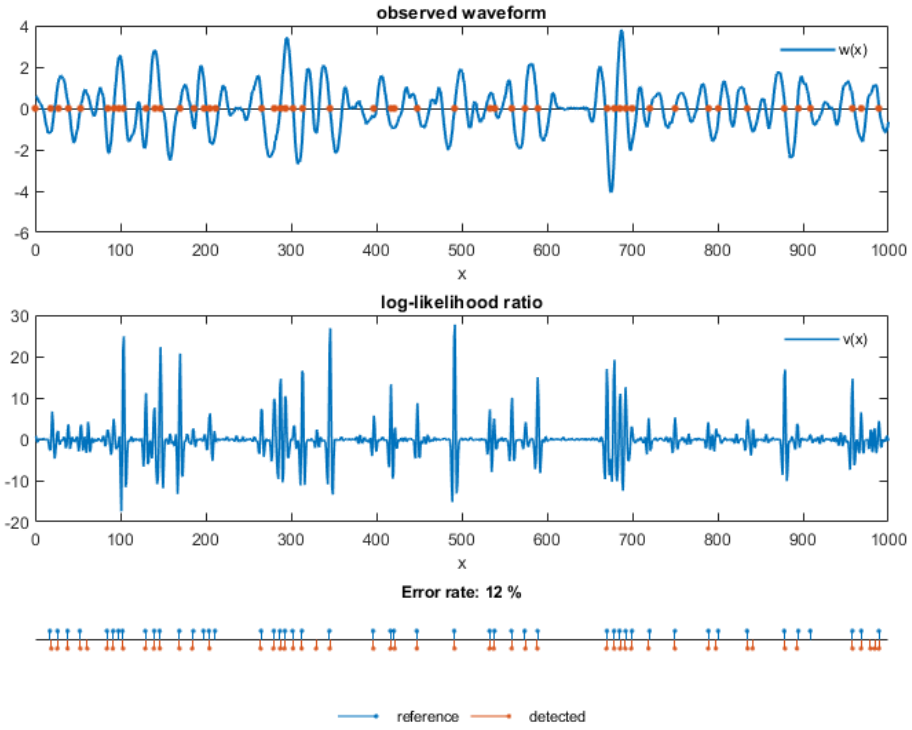


Figure 2.4: Result of the key point detector

2.2 Extension into 2-dimensional

The next thing is to extend it into a 2-dimensional case. This can be done by extending the domain of the Poisson point process into $\mathbb{R}^2$. The points θ_k on the real axis become 2D points $\theta_k = (x_k, y_k)$. The density $\lambda(x)$ becomes a 2D function $\lambda(x, y)$, and the autocovariance functions become 4D, where the integrals extend over the 2D space. For instance,

$$R_p(x_1, x_2, y_1, y_2) = \sigma_a^2 \int_{\beta=-\infty}^{\infty} \int_{\alpha=-\infty}^{\infty} \lambda(\alpha, \beta) s(x_1 + \alpha, y_1 + \beta) s(x_2 + \alpha, y_2 + \beta) d\alpha d\beta \tag{2.15}$$

The 1D neighbourhoods, e.g., $n = 1, \cdots, N$ become 2D $N \times N$ structures must be reshaped to 1D $N^2 \times 1$ structures to enable a description in terms of covariance matrices. This can be done by reshaping the N columns to just one large column with M^2 elements.

The shape of the psf also needs to be adapted to the 2-dimensional psf. Some of the example of these psf can be seen in Figure 2.5.

A more extensive discussion of this extension process will be discussed in the chapter itself.

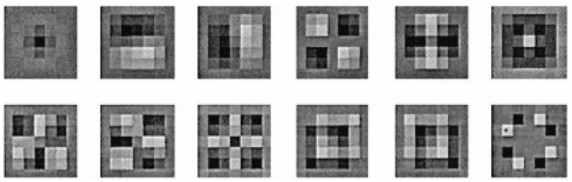


Figure 2.5: Examples of point spread function (van der Heijden et al., 1997)

2.3 Performance Evaluation

In order to evaluate the performance, there is a need to assess the detected keypoint. The detected keypoint can be classified as a correct detection, spurious, localisation error, or missed. In general, a localisation error is less bad than a spurious or a missed detection. Should a performance evaluation be based on incorrect detection, localisation error should therefore have less weight than a spurious or missed detection.

One way to do this performance evaluation is to do a Figure of Merit calculation, as proposed by van der Heijden (1992). The computational structure of this FOM method can be seen in the following diagram in Figure 2.6. This method works by first having the two signals, the detected signals and the reference signals. Then detected signals is subtracted from the reference signals. This will create a distinction between spurious and missed with the localisation error. After passing the signal through a low pass filter, the error should be calculated by taking the squared value then averaging them based on the number of reference points.

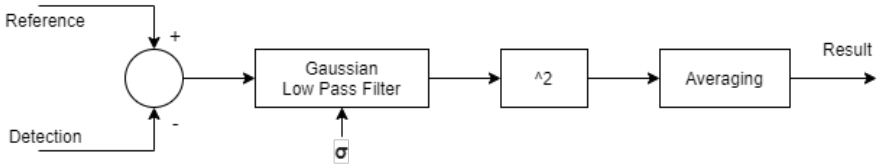
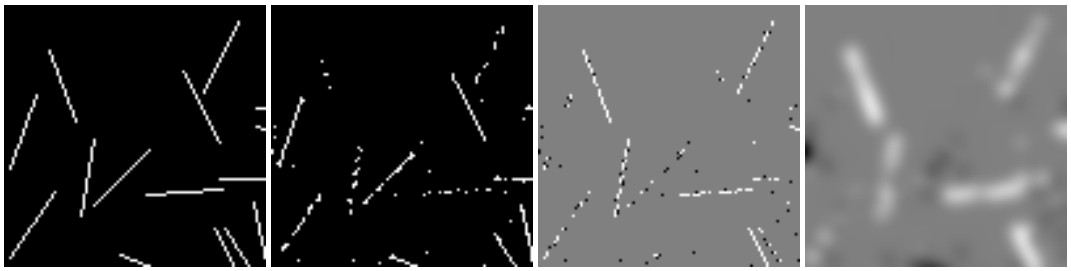


Figure 2.6: Computation structure of the figure of merit performance measure

The feature of this method is that the smearing caused by the low pass filter makes sure that a minor localisation error has a lower weight than a large localisation error. Localisation error still has less error weights than spurious or missed detection. All of these depend on the value of the σ of the Gaussian filter. This σ affects what would be considered a localisation error. The illustration of this method can be seen in Figure 2.7. It can be seen that the filtering caused some of the localisation error to mostly nullify each other.



(a) Reference **(b)** Detected **(c)** Error **(d)** Filtered Error

Figure 2.7: FOM Steps

3 Numerical Optimisation in 1-Dimensional CVM

3.1 Introduction

Although the feature detection has a decent error rate, it is still not optimal. Within a 1-dimensional model, there are still several questions that still need to be investigated. The CVM operator is a model-based approach to detect features within a record of a signal/image. The CVM operator is built with some input parameters (density, inhibition distance, etc.) called the design parameters. The CVM operator is tested with signals which are generated using the same model on which the CVM operator was designed. The values of these test parameters are not necessarily the same as the one which were used to build the CVM operator.

From this knowledge, this chapter would explore the extent to which the CVM operator can improve by optimising the weights and the parameter. The research question of this section, as mentioned in Chapter 1, are

1. Given the set of design parameters, which equals the set of test parameters, to what extent is it advantageous to numerically optimise the weights of the designed CVM operator?

2. Given a set of test parameters with which test signals are generated, to what extent is it advantageous to numerically optimise the design parameters of the CVM operator?

This chapter will start with the methods to answer the research questions. In this section, the experimental setups are provided. In the following section, the results of the experiments are presented. The discussion will follow, starting with the interpretation of the result. There will also be discussions on the limitation of the study along with a future recommendation. Finally, the conclusion will close this chapter.

3.2 Methods

This section presents the procedure on which the research question is answered. In this section, the analysis and the experimental setup is discussed. The research question will be investigated by means of experiments. These experiments are done on a 1D CVM operator with varying signals and varying parameters. The 1D CVM operator is the same one that has been discussed in the background. The performance measure here is FOM which has also been discussed previously.

One of the main reason that the performance is still not optimal is the limited amount of training resources. With only a length of 1000, the test signal is not able to be optimised properly. The numerical optimisation on a larger training set will have a more general and unbiased result. The numerical optimisation attempts to minimise the FOM value by taking the weights or parameters as a multivariable function. This operation makes use of `fminsearch` functions of MATLAB. This `fminsearch` functions makes use of Nelder-Mead simplex algorithm by Lagarias et al. (1998). This procedure can be seen in Figure 3.1 where the parameter adjustment can be either the weights of the CVM operator or the design parameter of the CVM.

3.2.1 Experimental Setup

Parameters Influence on CVM Operator Behaviour

Before the numerical optimisation is done, the first experiment will be conducted to observe the operator's behaviour. A varied design parameters will allow for more comparison and, consequently, insight into the operator's general behaviour. These insights can be useful to interpret the results of the numerical optimisation that will be performed later.

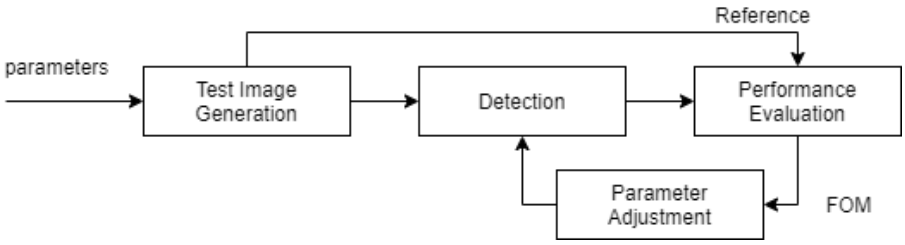


Figure 3.1: Structure of the numerical optimisation

The experiments will then see the resulting weights and response to varying parameters on a common signal. This signal will be a block shape signal which can be seen in Figure 3.2a.

The design parameters will be the variable in this experiment. These include the density, points amplitude, inhibition distance, and white noise. These values are given in Table 3.1. In this study, $\bar{x}$ denotes the original design value of a parameter. Variation of these values will be used to experiment. Points amplitude and noise are especially related as the ratio between them (σ_a/σ_n) forms the Signal to Noise Ratio used in the results.

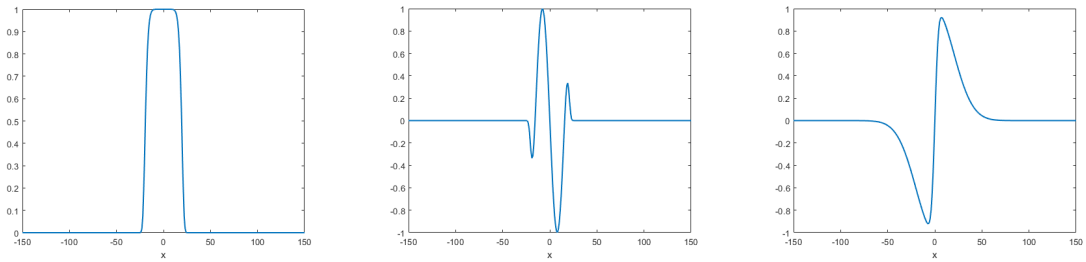
Table 3.1: Design Parameters and their Default Value

Parameters	Value
density $(\bar{\lambda(x)})$	0.05
standard deviation of points amplitude $(\bar{\sigma_a})$	1
inhibition distance $(\bar{R_i})$	4
standard deviation of white noise $(\bar{\sigma_n})$	0.05

Numerical Optimisation on Weights of CVM Operator

The next experiment will be to do numerical optimisation on the resulting weights. This experiment will be done on a much larger training set, with a length of 500000. The weights of the CVM operator will then be numerically optimised.

For the numerical optimisation on the weights of the CVM operator, the experiments will be performed on several types of signals. This experiment will use three types of signals: block shape, modulated shape, and edge shape. These three signals shape can be seen in Figure 3.2.



(a) Block Shape Signal **(b)** Modulated Signal **(c)** Edge Shape Signal

Figure 3.2: Examples of Common Signals

Moreover, to understand the influence that the parameters have on the FOM performance, the weights numerical optimisation will be performed on varying values of parameters. The experiment will be conducted by keeping all other parameters constant while modifying one parameter value. This will result in an overview of the parameter influence in the weights optimisation.

Numerical Optimisation on Parameters of CVM Operator

The last experiment in this section is numerical optimisation on the design parameter. Several parameters construct the CVM operator. These parameters can be different on the test signal when compared to the design parameter. This experiment attempts to minimise the FOM by modifying the value of the design parameter.

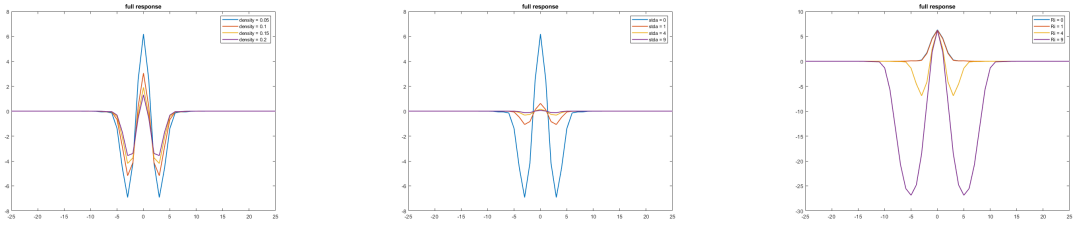
This experiment follows a similar approach as optimising the weights, however, with a signal of length 100000 instead of 500000 due to limited computation power and time efficiency. The experiment will be performed on the same three common signals, which can also be found in Figure 3.2.

This experiment will start by providing a default parameter as seen in Table 3.1. The wave will then be generated on the varied parameter value. This results in a measure of the numerical optimisation performance. The result's extension is the possibility of getting the unknown design parameter out of the numerical optimisation.

3.3 Results

3.3.1 Parameters Influence on CVM Operator Behaviour

The responses of the CVM operator when modifying a parameter are shown in Figure 3.3. These figures show the influence that parameter has on the CVM operator response. First, on the density of the points. Then the next one on the SNR (σ_a/σ_n) is done by modifying the value of the standard deviation of points amplitude. The third and final one is on the inhibition distance.



(a) density **(b)** SNR based on $sigma_a$ **(c)** inhibition distance

Figure 3.3: CVM operator responses with varied parameters on block shape signals.

From the figures, it seems that the density mainly affects the amplitude of the response. The SNR itself affects the amplitude and the width of the response. The inhibition distance affects the width of the response, where there is a 'valley' which means that there will be no response near to each other.

3.3.2 Numerical Optimisation on Weights of CVM Operator

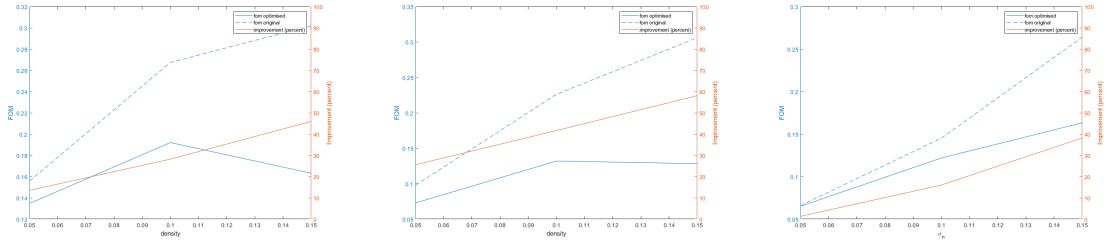
On the common case, the three signals with default design parameter value, the numerical optimisation can be seen in Table 3.2.

Table 3.2: Results of Weight Optimisation on Common Signals

	Block Shape	Modulated Shape	Edge Shape
Original FOM	0.156	0.098	0.063
Optimised FOM	**0.135**	**0.073**	**0.061**

The following results are on the influence that the parameters have on the optimisation. The first parameter that will be modified is density. The effect of this parameter on FOM and the optimisation can be seen in Figure 3.4. The figures show that with higher density, the detection

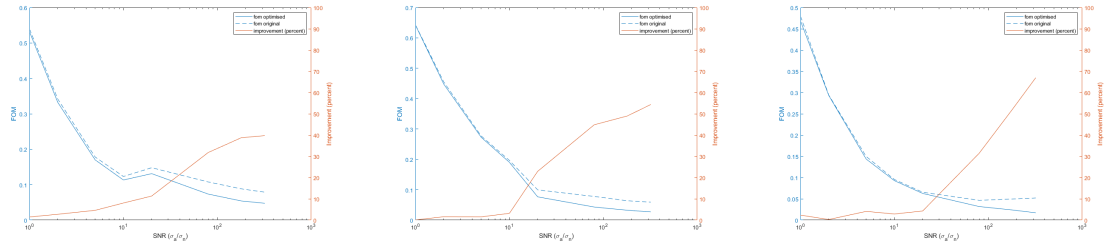
got worse. However, the optimisation compensates for that, as it shows a more stable FOM result.



(a) Block Shape **(b)** Modulated Shape **(c)** Edge Shape

Figure 3.4: FOM performance on various density value for weight optimisation

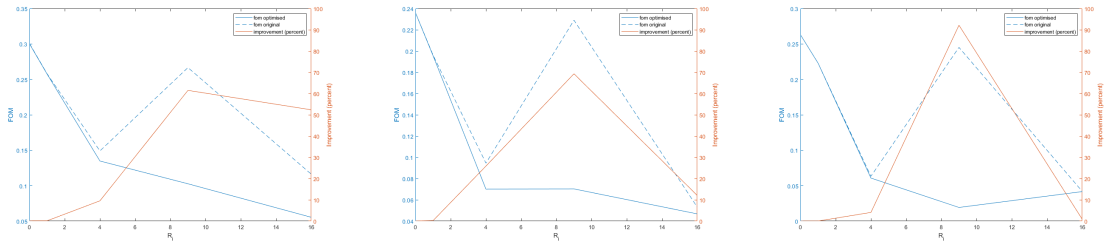
The following result comes from the varied value of SNR. Figure 3.5 visualise the results. This experiment shows that the variance of this parameter improves the FOM significantly until a value of around 20. After this range, optimisation does not improve the result as much anymore. However, there is still a trendline of improvement when the value of the parameter is higher.



(a) Block Shape **(b)** Modulated Shape **(c)** Edge Shape

Figure 3.5: FOM performance on various SNR value for weight optimisation

The FOM result for the various inhibition distance can be seen in Figure 3.6. The FOM might not necessarily decrease all the time without optimisation. However, after optimisation, the FOM have a trendline to go down as the inhibition distance increase.



(a) Block Shape **(b)** Modulated Shape **(c)** Edge Shape

Figure 3.6: FOM performance on various inhibition distance value for weight optimisation

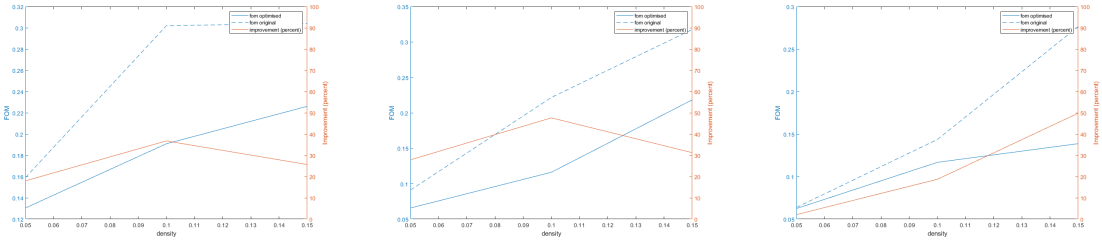
3.3.3 Numerical Optimisation on the Parameters of CVM Operator

In numerical optimisation of the design parameters, there might be differences in the parameter of the constructed wave to the design parameter. Table 3.3 show the optimisation done on the common example where the wave is constructed using the same design parameter as the initial input to the CVM operator. It can be seen that there is still some improvement that can be achieved even on the same design parameter.

Table 3.3: Results of Parameter Optimisation on Common Signals

	Block Shape	Modulated Shape	Edge Shape
Original FOM	0.159	0.091	0.063
Optimised FOM	**0.130**	**0.065**	**0.062**

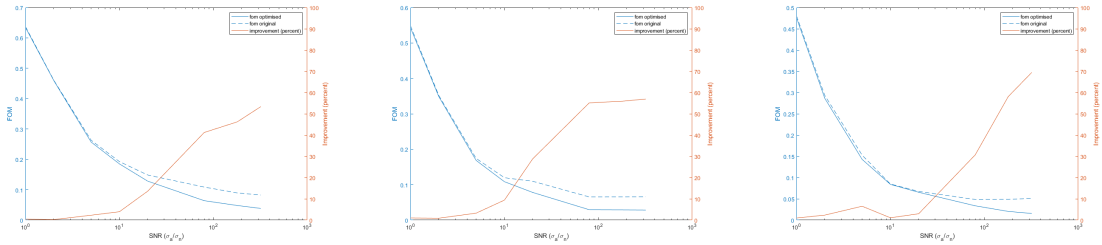
The following results are on optimisation possible when inputting default value parameters for the varied parameter value. The first parameter that will be modified is density. The FOM and optimised FOM on these varied values can be seen in Figure 3.7. As the density goes up, the FOM results also go up.



(a) Block Shape **(b)** Modulated Shape **(c)** Edge Shape

Figure 3.7: FOM performance on various density value for parameter optimisation

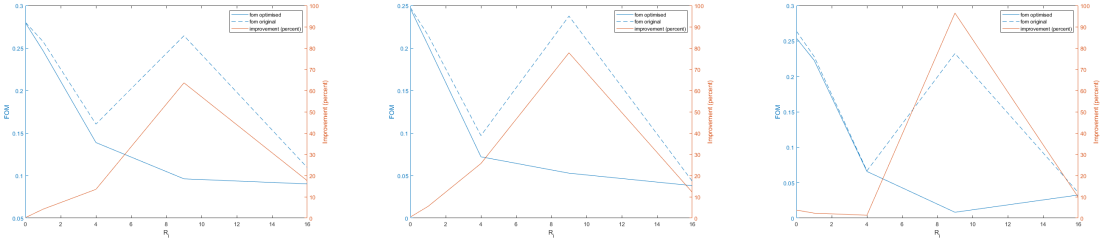
The following results are on the FOM results given varied value of the SNR. Figure 3.8 visualise the results. This parameter shows a better FOM on a higher parameter value. The figures also show greater improvement on the higher parameter value.



(a) Block Shape **(b)** Modulated Shape **(c)** Edge Shape

Figure 3.8: FOM performance on various SNR value for parameter optimisation

The original and optimised FOM result for the various inhibition distance can be seen in Figure 3.9. The optimised FOM shows improvement as the inhibition distance increase.



(a) Block Shape **(b)** Modulated Shape **(c)** Edge Shape

Figure 3.9: FOM performance on various inhibition distance value for parameter optimisation

The last result of the parameter optimisation is the resulting parameter value after optimisation. The summarised version containing the Root-Mean-Square of the difference between the design parameter value and the optimised parameter value can be seen in Table 3.4.

Table 3.4: RMS value between Design Parameter and Optimised Parameter

	Block Shape	Modulated Shape	Edge Shape
density (λ)	0.0203	0.0047	0.0259
standard deviation of amplitude (σ_a)	0.2216	0.1797	0.4014
inhibition distance (R_i)	1.3483	1.4185	0.8370
standard deviation of noise (σ_n)	0.0206	0.0295	0.0223

3.4 Discussion

There are several discussions to be made according to the results. The general result of the experiment comes as expected. The parameter influence helps in understanding the behaviour in the other experiments.

The result on the weight optimisation is also relatively expected as the improvement made on more complex signals such as the modulated shape are higher when compared to the already quite clear detection in edge shape signal. The same goes for the optimisation of the parameter.

The interesting observation from the experiment is that the general behaviour of the two methods of optimisation shows a similar result. This can be seen from the similar resulting behaviour between the weight and parameter optimisation. This could mean that a numerical optimisation could be done on either one of them to achieve a comparable result.

The observation on the density where the result gets worse over time corresponds to the founding in the response behaviour with higher density. Figure 3.3a shows that with higher density, the amplitude of the response gets lower. This makes for a harder decision and, therefore, a worse detection.

Similar things can be said regarding the SNR. As seen in Figure 3.3b as the amplitude gets higher, the response allows for an easier decision process. This corresponds to the founding in the weight and parameter optimisation, which shows as the amplitude goes up, it is possible to get a better result.

The inhibition distance prevents any events within a certain distance to another. Therefore, it can be expected that with a higher inhibition distance, points can be more easily detected as they are more likely to be in a certain space. This corresponds to the founding in the results. Moreover, it can be expected that with a default density value of 0.05 and inhibition distance of close to 20 would have an expected outcome of uniformly spread out events. This would probably result in a very deterministic detection. This might not be desired for the numerical optimisation as it would result in more missed detection.

Lastly, the optimisation on the parameter shows a very close optimised parameter when done on an unknown design parameter. The difference in the inhibition distance might look quite large. However, this is mostly caused by the larger inhibition distance, where it gets to a value of 16. Therefore this result is still considered quite small in comparison.

3.4.1 Limitations and Recommendations

Due to limited time and resource, the experiment done on this chapter can only be done on a small scale. The varied parameters were chosen on an exponential scale to show the general behaviour. There can be more combination in the experiment, which can generate more insight into the CVM operator behaviour.

According to the founding in this experiment, it seems that either the weight or parameter optimisation is enough to get into the optimal result. However, this is still not verified. For future study, the combination of weights and parameter optimisation can be explored.

Optimisation on the parameter on unknown input shows the possibility of doing the same thing into the parameter of the signal. This means that the parameter of an unknown signal that generates the wave can become known through optimisation on the signal parameter. Therefore another interesting future study would be to explore the possibility of determining an unknown input signal using numerical optimisation.

3.5 Conclusion

This experiment will be assessed based on the goals, which is how the CVM operator can improve by optimising the weights and the parameter. This will be answered by answering each of the research questions.

The improvement made on optimising the weights of the designed CVM operator depends heavily on the type of signals and the parameters of the CVM operator itself. In a common case scenario, 20% improvement can be expected out of the optimisation process. Meanwhile, in an extreme case, the gain can be as high as 70%.

The improvement made on optimising the design parameter of the CVM operator is quite similar to the improvement made on the weights optimisation. However, as expected, the amount of improvement in the parameter optimisation depends on the difference the default parameter has compared to the unknown parameter. A larger difference means that there is more room for improvement. This improvement can be as large as 77% improvement in some cases.

Moreover, in this experiments, many insights towards the behaviour of the CVM operator in a 1-dimensional case were made. The CVM operator's response on various design parameters allows for interesting observation and explanation towards the limits of optimisation.

4 1-Dimensional Data-Driven Keypoint Detection

4.1 Introduction

Deep learning is very commonly applied in 1-dimensional data. Examples of a 1-dimensional data-driven approach are speech recognition, waveform prediction, and waveform segmentation. Out of these examples, feature detection is part of waveform segmentation, where the features are segmented with a different label compared to the rest of the signals.

This chapter will explore the use of a data-driven method for feature detection in a 1-dimensional case. This can be a beneficial insight into the comparison that can be expected from a model-based and data-driven approach in the 2-dimensional case. As mentioned in Chapter 1, the research questions for this chapter are

1. To what extent can a data-driven approach, such as a convolution neural network, outperform the model-driven approach?

2. To what extent can a data-driven approach fed by signals derived from a model-based approach outperform the purely data-driven approach or purely model-driven approach?

This chapter will start with the methods to answer the research questions. In this section, the experimental setups are provided. In the following section, the results of the experiments are presented. The discussion will follow, starting with the interpretation of the result. There will also be discussions on the limitation of the study along with a future recommendation. Finally, the conclusion will close this chapter.

4.2 Method

There are several options of neural network that can be used for data-driven waveform segmentation. Convolutional Network, or a variation of it, is more commonly used in 2-dimensional data such as images (Albawi et al., 2018). However, it can be modified so that it accepts a more 1-dimensional input. In comparison, for comparison of performance with a type of network that is suitable for sequential or 1-dimensional input, the Long Short Term Memory (LSTM) network will also be part of the experiment (Hochreiter and Schmidhuber, 1997).

CNN is usually well known to be used in image segmentation, where the main usages revolve around training the network to find the weights that can output enough difference for classification. This will be done towards 1-dimensional data, which means that it will recognise a certain pattern in the waveform to recognise the events. LSTM network, meanwhile, was developed with sequential input as the primary purpose. It is well-suited for classification, processing, and predictions on time series data. It is also commonly used for ECG wave segmentation or prediction.

Both of these methods are constructed on MATLAB with further specifications listed in the following experimental setup.

4.2.1 Experimental Setup

The data for this experiment will be generated using the Poisson Point Process (PPP). This dataset will contain 1000 waveform of 5000 lengths each. This will be separated into 70% training data and 30% test data. Moreover, this is done with three different event signal shape. There are block shape, modulated shape, and edge shape signals. These signals are the same one that is used in Chapter 3 and can be seen in the Figure 3.2

The MATLAB code used to set up the network and the training options used can be seen in Appendix A.

The CNN networks are constructed on ten layers. However, there is a problem with MATLAB deep learning library regarding 1-dimensional reference labels. When parsing a 1-dimensional categorical reference, MATLAB recognises it as a vector that does not allow for image segmentation. This makes it impossible to have a purely 1-dimensional convolutional network. The option that has been done was to concatenate the wave into a $2 \times N$ data format to accommodates this network. The network then accepts a $2 \times N$ input with a $2 \times N$ categorical label to allow semantic segmentation.

The network is composed of the input layer, which goes into the convolution layer. Afterwards, there is a pooling section along with the ReLu layer. In the end, it makes use of a classification layer with custom weights according to the inverse frequency of the class labels.

The LSTM networks are constructed on five layers in MATLAB. The first is a sequence input layer, which allows for 1-dimensional input. The next is the bidirectional LSTM layer with 200 hidden units. Afterwards, there is a fully connected layer that has two outputs, denoting the labels. Later, a softmax layer is used to normalise the output to a probability distribution over the predicted output class. Finally, the classification layer computes the loss function.

The trained network will be used to do waveform segmentation. The resulting feature detection is assessed using the Figure of Merit method, which was also used in the previous chapter. This allows for performance comparison between only the CVM operator, only a 1-dimensional neural network, and a combination of the two.

There will be two types of experiments following the different research questions. The first one will be a network trained and tested on the Poisson point process generated waveform. This will be the baseline performance of the data-driven method on 1-dimensional data. The second one will be a network trained and tested on the log-likelihood ratio as the output of the CVM.

1-dimensional Neural Network on Generated Wave

The first experiment will be more straightforward. The LSTM and CNN will both be trained based on the original waveform and the reference. This training aims to let the neural network recognise the shape of the event and therefore detect the feature.

The result of this experiment will be the FOM performance on the trained network. This will be used to compare the FOM result in the previous chapter, allowing for a comparison between model-based and data-driven performance.

1-dimensional Neural Network on CVM Output

The second experiment will make use of the CVM operator to have the log-likelihood ratio output. This resulting log-likelihood ratio will be used to train and test the network. This incorporation of CVM into LSTM will be done twice, first on unoptimised CVM operator weights and then finally on optimised weights of the CVM operators.

Optimisation of CVM weights is done on a 500000 length wave following the procedure done in Chapter 3. This is to minimise the computation time in optimising 1000 generated waves. The resulting optimised weights and their corresponding transformation matrices are used to get the log-likelihood ratio on the dataset.

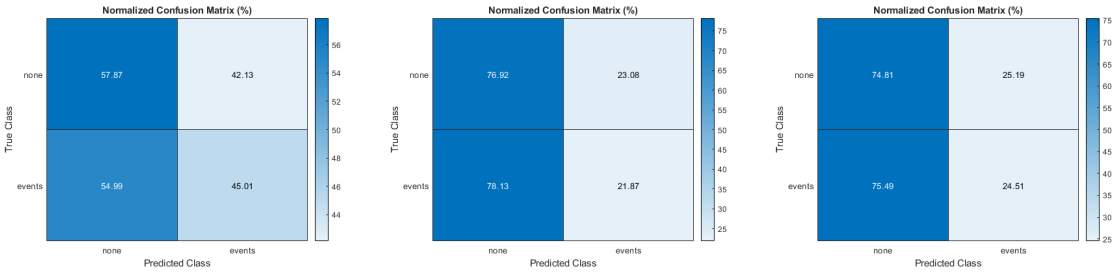
The experiment will result in FOM measures which will be used to assess the performance of model-based and data-driven method used together. It will also show the differences between a network trained and tested on unoptimised and optimised model-based output.

4.3 Results

There are two sets of results on the 1-dimensional network on the generated wave. For the CNN results, Figure 4.1 show the confusion matrix and Table 4.1 shows the FOM values on each of the signals tested. Confusion matrices allow for an easy overview of correct, spurious, or missed detections. The table shows the FOM results for each signal while also showing the standard deviation among the test data results.

Table 4.1: FOM results of CNN network on generated wave

Signal shape	mean FOM	std FOM
block shape	60.7048	8.238
modulated shape	30.7213	4.498
edge shape	16.7142	1.460



(a) Block Shape **(b)** Modulated Shape **(c)** Edge Shape

Figure 4.1: Confusion Matrices of CNN network on raw waveforms with different signals

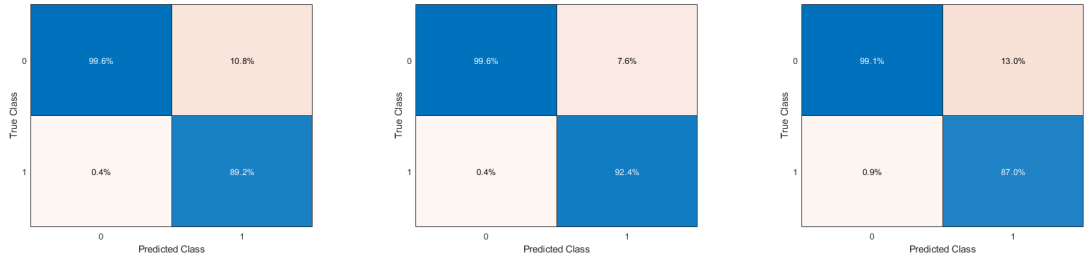
The second sets of results are on the LSTM network. Table 4.2 compares the results from the LSTM on raw signal, unoptimised CVM into LSTM, optimised CVM into LSTM, and finally the CVM operator itself for comparison. The table shows the FOM results for each signal while also showing the standard deviation for the test.

Table 4.2: FOM results of LSTM network on various input, with CVM operator for comparison

	Signal shape	mean FOM	std FOM
LSTM with raw image	block shape	**0.0442**	0.02
	modulated shape	**0.0482**	0.02
	edge shape	0.2445	0.035
LSTM with unoptimised LLR	block shape	0.1779	0.035
	modulated shape	0.0824	0.02
	edge shape	**0.0593**	0.02
LSTM with optimised LLR	block shape	0.1871	0.03
	modulated shape	0.0899	0.022
	edge shape	0.1154	0.028
optimised CVM	block shape	0.1356	0.026
	modulated shape	0.0757	0.02
	edge shape	0.0801	0.017

Figure 4.2 show the confusion matrix of the LSTM network detection on the raw test waveforms.

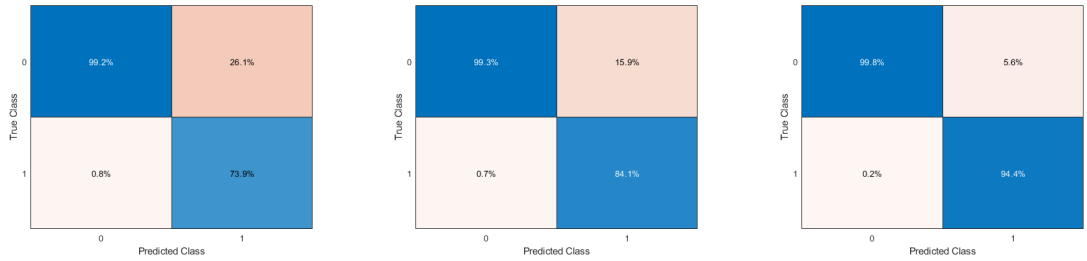
There are two sets of results on the 1-dimensional network on the incorporation of CVM into deep learning. The first one is on the unoptimised CVM into LSTM. The second one is on the optimised CVM operator weights into LSTM. Each sets will show the confusion matrices as



(a) Block Shape **(b)** Modulated Shape **(c)** Edge Shape

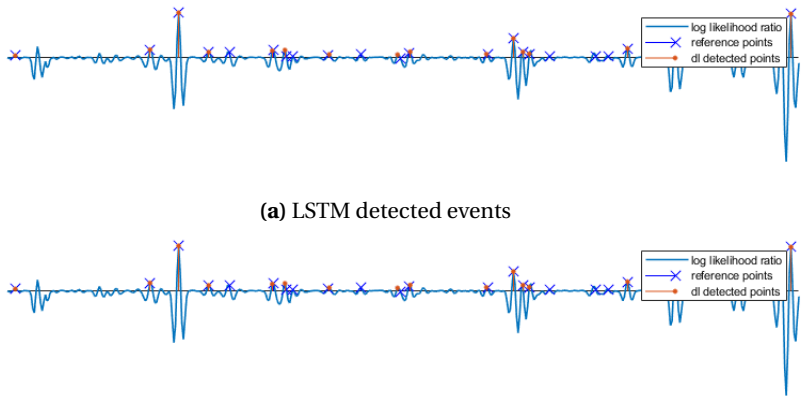
Figure 4.2: Confusion Matrices of LSTM network on raw waveforms with different signals

shown in Figure 4.3 and 4.5. Then there will be an example of a detection on the waveform for each of the network as shown in Figure 4.4 and 4.6.



(a) Block Shape **(b)** Modulated Shape **(c)** Edge Shape

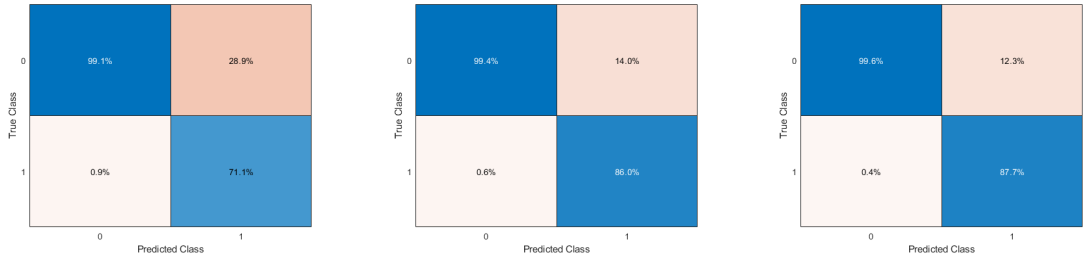
Figure 4.3: Confusion Matrices of LSTM network on CVM output with different signals



(a) LSTM detected events

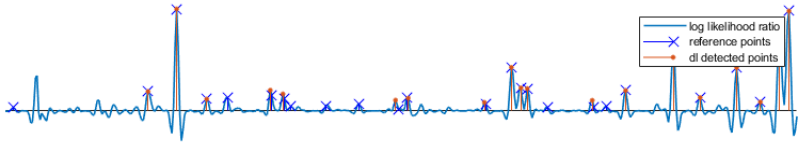
(b) CVM detected events

Figure 4.4: Detected events on log likelihood ratio of block shape PPP with unoptimised CVM weights

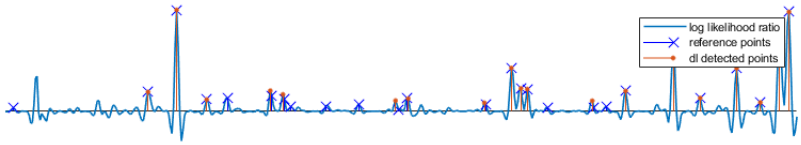


(a) Block Shape **(b)** Modulated Shape **(c)** Edge Shape

Figure 4.5: Confusion Matrices of LSTM network on optimised CVM output with different signals



(a) LSTM detected events



(b) CVM detected events

Figure 4.6: Detected events on log likelihood ratio of block shape PPP with optimised CVM weights

4.4 Discussions

4.4.1 Interpretation of Results

From the result, as seen in Figure 4.1 and Table 4.1, CNN does not perform very well on 1-dimensional data. There is too much spurious detection, as can be seen from the confusion matrices. This is possibly due to the filter size, as it is harder to recognise the signal's shape with only a small filter. Efforts have been made to fix this by changing the filter size, but even after varying the filter width from 2 to 20, this still does not improve the result.

Other efforts such as changing the CNN setup have also been made into other Fully Connected Network architecture. This, however, still does not give any remarkable result. The result that is presented for CNN is by no means the best result that was obtained. However, it is a general representation of the CNN result where it simply does not converge or simply decide to classify everything into one class. Basic hyperparameter tuning has also been attempted. However, this does not result in any significant improvement.

LSTM, in comparison, is performing well and gives out promising results. Looking at the comparison at Table 4.2, it does, in fact, exceeds the performance of the CVM operator. There is a minimal amount of error, which is an excellent result coming from a neural network. The method of using log-likelihood to train artificial neural network has advantages and disadvantages. The benefits being a faster training time. This is likely to be due to easier recognition of the peak of log-likelihood ratio. It converges to a relatively stable state within around 20-30% of the iteration of the PPP wave training.

However, there might be information loss due to the imperfections of the CVM models. This might cause missed detection in pure CVM operator looking for local maxima; however, this

problem can sometimes be solved with the deep learning method. This can be seen in Figure 4.4 where a missed detection in pure CVM is still detected in the combined method. Looking at the final FOM result shown in Table 4.2, it shows a relatively good result, however, it is still not perfect. This might or might not become better through more training.

An interesting observation seems to be that optimised weights on the CVM operator do not seem to make the result better. This can be seen from the result in Table 4.2. The same reason might cause this as before; optimising the weights allows for sharper responses. This would cause the network to assume that peaks or local maxima should be categorised as an event, which causes spurious detections. In this case, looking at Figure 4.4 and 4.6, it seems that it is quite likely to cause double events detected in a single events. Therefore, it seems that there are no real differences in doing the weights optimisation for the CVM operator if combined with the deep learning network.

4.4.2 Limitations and Recommendations

There are some limitations to the study found in this chapter. The most obvious is that MATLAB does not allow image segmentation using a vector-shaped matrix, forcing the input and output of the network to be $2 \times N$ where N is the length of the signal. This is already getting into 2-dimensional data instead of 1-dimensional data even though it highly simulates it.

The next things is the amount of training and the optimality of such training done on the network. It is possible that with further training or optimisation on the hyperparameters, both CNN and LSTM network will be able to have a better result.

Another thing that can be explored is the validation of the results in comparison to the other state of the art feature detection. This could be done as a comparison of different state-of-the-art model-based feature detection methods aside from CVM.

4.5 Conclusions

The goal of this experiment was to see the extent of the data-driven approach compared to the model-based approach in a 1-dimensional case. This was done in the hope of more significant insights for further experiments in the 2-dimensional case. The performance and limitation of the data-driven method have been explored in this chapter.

The data-driven approach on PPP shows outperforms the performance of the model-based approach in some cases. While CNN performs poorly, LSTM has better FOM results even when compared to the CVM. It exceeds the CVM performance. However, in return, it takes time to train a good network.

The combination of model-based and data-driven does not show improvement in terms of FOM values. However, this method shows improvement in the training time of the network. It reduces the training time by around 70% when compared to the training on the PPP. From the experiment, the limitation seems to be on the double detection of a single event. An optimisation effort on the weights also does not seem to improve the result, as it only causes a sharper peak on the log-likelihood ratio.

This experiment does not fully cover the extent of the 1-dimensional data-driven approach. Nevertheless, for the scope of the study, these experiments show insights into the advantages and disadvantages that can be expected in the 2-dimensional data-driven approaches later.

5 2-Dimensional CVM Feature Detection

5.1 Introduction

A big part of image processing comes in image feature detection. It is used for semantic segmentation of the image and keypoint detection. Depending on the type of detection, these could be line, edge, corner, or points of interest. Some known model-based feature detectors are Canny, Sobel, and Harris (Canny, 1986; Kanopoulos et al., 1988; Harris and Stephens, 1988).

This chapter will explore the extension of the covariance model-based feature detector, which has been discussed in Chapter 2 into a 2-dimensional case. This method has previously been explored by van der Heijden (1992). The CVM operator will first be designed for the spots, line, and edge detector. As mentioned in Chapter 1, the research questions for this chapter are

1. How is the performance of the covariance model feature detector on spots, line, and edge detection compared to the literature?

This chapter will start with the methods to answer the research questions. In this section, the analysis and experimental setups are provided. In the following section, the results of the experiments are presented. The discussion will follow, starting with the interpretation of the result; there will also be discussions on the limitation of the study along with a future recommendation. Finally, the conclusion will close this chapter.

5.2 Method

5.2.1 The CVM Operators for Spots Detection

The extension into 2-dimensional cases starts with the extension into the spots detector. The events in the 1-dimensional case can be seen as a spot in an image in the 2-dimensional case. This also means that such events can be modelled in the CVM operator. The image model for this condition can also be modelled with a 2D Poisson Point Process. This PPP can be defined as follows,

$$w(x,y) = \Sigma_k a_k s(x-\xi_k, y-\eta_k) + b + n(x,y) \tag{5.1}$$

Where k denotes the events, s defines the specific events, a 2D Gaussian function, at a certain location specified with ξ and η. These events are further modified by random Gaussian amplitude a. The whole image is then affected by unknown offset b and observation noise n. The event itself is defined as a 2D Gaussian function, as seen in the following equation.

The autocovariance function needs to be modified such that it accommodates a 2-dimensional case such as this. Following the methods used by van der Heijden (1992), the autocovariance functions for the presence and absence of a spots in A_{nm} such as shown in Section 2, 1D CVM operator, can be defined. In these equations, vectors will be used to denotes coordinates such as $\vec{x}_n = (x_n, y_n)$, $\vec{\xi}_n = (\xi_n, \eta_n)$, and $\vec{\alpha} = (\alpha, \beta)$.

$$R_w(\vec{x}_1, \vec{x}_2 | \vec{\xi}_k \notin A_{nm}) = R_p(\vec{x}_1, \vec{x}_2 | \vec{\xi}_k \notin A_{nm}) + \sigma_b^2 + R_n(\vec{x}_1 - \vec{x}_2) \tag{5.2}$$

with

$$R_p(\vec{x}_1, \vec{x}_2 | \vec{\xi}_k \notin A_{nm}) = \sigma_a^2 \lambda \int_{\vec{\alpha}=-\infty}^{\infty} s(\vec{x}_1 - \vec{x}_2 + \vec{\alpha}) s(\vec{\alpha}) d\vec{\alpha} - \sigma_a^2 \lambda \int_{\vec{\alpha} \in A_{nm}} s(\vec{x}_1 + \vec{\alpha}) s(\vec{x}_2 + \vec{\alpha}) d\vec{\alpha} \tag{5.3}$$

$$R_w(\vec{x}_1, \vec{x}_2 | \vec{\xi}_k \in A_{nm}) = R_s(\vec{x}_1, \vec{x}_2 | \vec{\xi}_k \in A_{nm}) + R_p(\vec{x}_1, \vec{x}_2 | \vec{\xi}_k \in A_{nm}) + \sigma_b^2 + R_n(\vec{x}_1 - \vec{x}_2) \tag{5.4}$$

with

$$R_s(\vec{x}_1,\vec{x}_2|\vec{\xi}_k \in A_{nm}) = \sigma_a^2 s(\vec{x}_1-\vec{\xi}_k)s(\vec{x}_2-\vec{\xi}_k) \tag{5.5}$$

$$R_p(\vec{x}_1,\vec{x}_2|\vec{\xi}_k \in A_{nm}) = \sigma_a^2\lambda\int_{\vec{\alpha}=-\infty}^{\infty} s(\vec{x}_1-\vec{x}_2+\vec{\alpha})s(\vec{\alpha})d\vec{\alpha} - \sigma_a^2\lambda\int_{|\vec{\alpha}-\vec{\xi}|<R_i} s(\vec{x}_1+\vec{\alpha})s(\vec{x}_2+\vec{\alpha})d\vec{\alpha} \tag{5.6}$$

The measurement vector then need to be defined. This measurement vector is 2-dimensional for the as it is a 2-dimensional observation, yet it needs to be 1-dimensional for the autocovariance function. This can be done by rearranging the $M \times M$ neighbourhood into a $1 \times N$ neighbourhood. Elements of the measurement vector are then defined by,

$$\begin{aligned} \underline{\vec{w}}_{nm_i} &= w(n\Delta + x_i, m\Delta + y_i) \qquad i = 0,1,2,...,N-1 \\ x_i &= (i/M - M/2)\Delta \\ y_i &= (i\%M - M/2)\Delta \qquad N = M \times M \qquad \text{where M is odd} \end{aligned} \tag{5.7}$$

In the 2-dimensional case of the CVM operator, a 2D neighbourhood has to be arranged in such a way that it can fit into 1D data so that the autocovariance function can be calculated. As known from the autocovariance function input, the autocovariance function of a 2D case is a 4D input, and this is not usable in the CVM operator. Therefore, block-toeplitz is used to arrange the vectors such that it can be treated as a 2D input. The 1-dimensional $1 \times N$ neighbourhood will be a block-toeplitz in the autocovariance function. This is explained more in the Appendix B. The detector for a single event can then be defined with the hypotheses as follows.

$$\begin{aligned} R_{0_{ij}} &= R_w(\vec{x}_1,\vec{x}_2|\vec{\xi}_k \notin A_{nm}) \\ R_1(0,0)_{ij} &= R_w(\vec{x}_1,\vec{x}_2|\vec{\xi}_k = \vec{0}) \end{aligned} \tag{5.8}$$

The rest of the steps in the construction of the CVM operator follows from the 1-dimensional CVM operator in Section 2. In short, it goes through simplification through principal component analysis. The resulting log-likelihood ratio is then used for the statistical test. For these spot detections, it is sufficient to check if the log-likelihood ratio is a local maximum as well as above a certain threshold. This is seen in the following equation.

$$(n\Delta, m\Delta) \text{ is a spot element iff} \begin{cases} v_{nm} > \text{threshold} & \text{AND} \\ v_{n,m-1} < v_{nm} \text{ AND } v_{n,m+1} < v_{nm} & \text{AND} \\ v_{n-1,m} < v_{nm} \text{ AND } v_{n+1,m} < v_{nm} & \end{cases} \tag{5.9}$$

5.2.2 The CVM Operator for Line Detection

The line detection follows through the spots detection. Some modification to the shape of the events and the autocovariance functions are needed. A forgiving window function $p(x)$ is introduced in the shape model of a line, representing the uncertainty of existing line elements nearby in a particular direction. The shape of the line is then defined as follows.

$$s(x,y,\phi) = s_l(x\sin\phi + y\cos\phi)p(x\cos\phi - y\sin\phi) \tag{5.10}$$

$$p(x) = exp\left[\frac{-x^2}{2\sigma_p^2}\right] \tag{5.11}$$

The autocovariance functions are also affected by the changes in the model. Most notably due to the addition of the integration for the ϕ for the rotational invariant line detection. Then, for example, the knowledge that a line element with $\phi \in \Phi$ exists in the region A_{nm} will result in the autocovariance function in the following equation.

$$R_s(\vec{x}_1, \vec{x}_2, \vec{\xi}_k) = \frac{\sigma_a^2}{|\Phi|} \int_{\phi \in \Phi} s(x_1 - \xi_k, y_1 - \eta_k, \phi) s(x_2 - \xi_k, y_2 - \eta_k, \phi) d\phi \tag{5.12}$$

where $|\Phi|$ refers to the length of the interval Φ. This changes can also be applied towards the $R_p(\vec{x}_1, \vec{x}_2 | \vec{\xi}_k \notin A_{nm})$ and $R_p(\vec{x}_1, \vec{x}_2 | \vec{\xi}_k \in A_{nm})$.

$$\begin{aligned} R_p(\vec{x}_1, \vec{x}_2 | \vec{\xi}_k \notin A_{nm}) = \frac{\sigma_a^2 \lambda}{|\Phi|} \int_{\vec{\alpha}=-\infty}^{\infty} \int_{\phi \in \Phi} s(\vec{x}_1 - \vec{x}_2 + \vec{\alpha}) s(\vec{\alpha}) d\phi d\vec{\alpha} \\ - \frac{\sigma_a^2 \lambda}{|\Phi|} \int_{\vec{\alpha} \in A_{nm}} \int_{\phi \in \Phi} s(\vec{x}_1 + \vec{\alpha}) s(\vec{x}_2 + \vec{\alpha}) d\phi d\vec{\alpha} \end{aligned} \tag{5.13}$$

$$\begin{aligned} R_p(\vec{x}_1, \vec{x}_2 | \vec{\xi}_k \in A_{nm}) = \frac{\sigma_a^2 \lambda}{|\Phi|} \int_{\vec{\alpha}=-\infty}^{\infty} \int_{\phi \in \Phi} s(\vec{x}_1 - \vec{x}_2 + \vec{\alpha}) s(\vec{\alpha}) d\phi d\vec{\alpha} \\ - \frac{\sigma_a^2 \lambda}{|\Phi|} \int_{|\vec{\alpha} - \vec{\xi}| < R_i} \int_{\phi \in \Phi} s(\vec{x}_1 + \vec{\alpha}) s(\vec{x}_2 + \vec{\alpha}) d\phi d\vec{\alpha} \end{aligned} \tag{5.14}$$

The rest of the steps can then follow through the spots detection. However, on determining the line elements based on the loglikelihood ratio, the condition needs to be adjusted. Instead of a local maxima in both directions, the line elements is considered when it is a local maxima in at least 1 direction. This can be seen in the following conditions.

$$(n\Delta, m\Delta) \text{ is a line element iff} \begin{cases} v_{nm} > \text{threshold} & \text{AND} \\ v_{n,m-1} < v_{nm} \text{ AND } v_{n,m+1} < v_{nm} & \text{OR} \\ v_{n-1,m} < v_{nm} \text{ AND } v_{n+1,m} < v_{nm} & \end{cases} \tag{5.15}$$

5.2.3 The CVM Operator for Edge Detection

The edge detector can be done as an extension of the line detection. The main differences are the step edges need two window functions. The additional window function is used to express the uncertainty concerning the exact course of a boundary segment that passes through the hypothesised edge. This change to the model shape can be seen in the following equations.

$$s(x, y, \phi) = s_e(x \sin\phi + y \cos\phi) p_r(x \sin\phi + y \cos\phi) p_p(x \cos\phi - y \sin\phi) \tag{5.16}$$

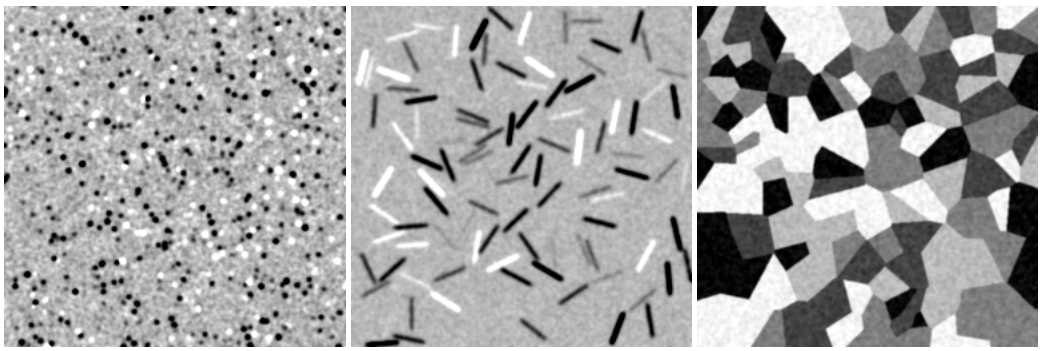
The rest of the steps follow the same methods as the line detection.

5.2.4 Experimental Setup

The experiment is done four times for spots, line, and edge detection. For each of the detector, a set of 100 images are used as a test set. This ensures a statistically significant result. 2D Poisson Point Process generates the data set for spots detection, Mikado test for line detection, and Voronoi test for edge detection. Examples of these datasets can be seen in Figure 5.1.

The parameters for the CVM operator can be seen in the following Table 5.1.

The performance measure used in this experiment is the FOM. After obtaining the results with the CVM operator, optimisation will then be done on the weights of the CVM operator. The



(a) 2D PPP **(b)** Mikado **(c)** Voronoi edge

Figure 5.1: Example of the Dataset Image

Table 5.1: Parameter Value of Various CVM Operators Feature Detector

	Spots	Line	Edge	Corner
density (λ)	0.02	0.04	0.04	0.04
SNR (σ_a/σ_n)	20	20	10	20
inhibition distance (R_i)	2	3	3	3
width of projection function (σ_r)		4	4	4
width of forgiving function (σ_p)			4	
Neighbourhood size ($M \times M$)	11x11	11x11	11x11	9x9
Orientation range (Φ)		[0 π]	[0 π]	[0 π]
Second orientation range (Θ)				[π/6 5π/6]

choice of only optimising the weights of the CVM operator follows from the conclusion in Chapter 3 that optimisation on the weights and design parameter results in the same behaviour. Weights optimisation is more efficient to do compared to design parameter. This optimisation follows the same methods like the one used in Chapter 3. The weights will be the variable while trying to look for the minimum FOM value.

The result that can be expected out of this experiment is the validation of the spots, line, and edge detector performance in comparison to the one found in the literature using the same FOM evaluation. These will all be compared to the results of the optimised weight.

5.3 Results

The main results from the experiments are the validation of the performance of the CVM operator on various tests. This result can be seen in the following Table 5.2.

Table 5.2: FOM value for various detections over 100 test data.

	Spots	Line	Edge	Corner
mean FOM	0.2090	0.3619	0.1640	0.98
std FOM	0.0140	0.0269	0.0085	0.0040
mean FOM (optimised)	**0.2086**	**0.3248**	**0.1181**	**0.98**
std FOM (optimised)	0.0147	0.0247	0.0047	0.0040

Examples of detection for each of the shape are shown in Figure 5.2. All of these images are only a partial image of 100 × 100 pixels instead of the full images. The first column shows the test images, the second one shows the reference, the third one shows the detection, and the last one shows the detection based on the optimised weight.

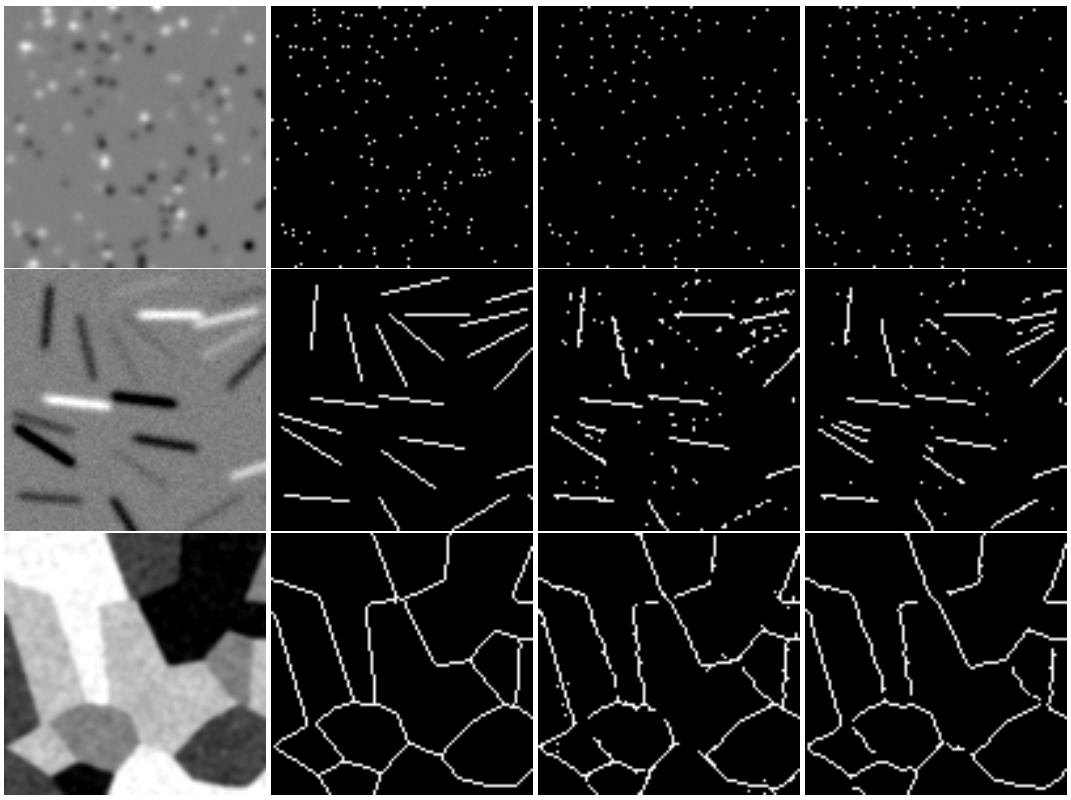


Figure 5.2: Example of the test results for spots, line, and edge (top to bottom) showing test image, reference, CVM detection, and CVM detection with optimised weights (left to right)

Figure 5.3 show some examples of the detections on real life scenario. The left column shows the image where the detections are performed. This is followed by the log-likelihood ratio and the detections on unoptimised CVM operator. Then on the right would be the log-likelihood and the detections on the optimised CVM operator.

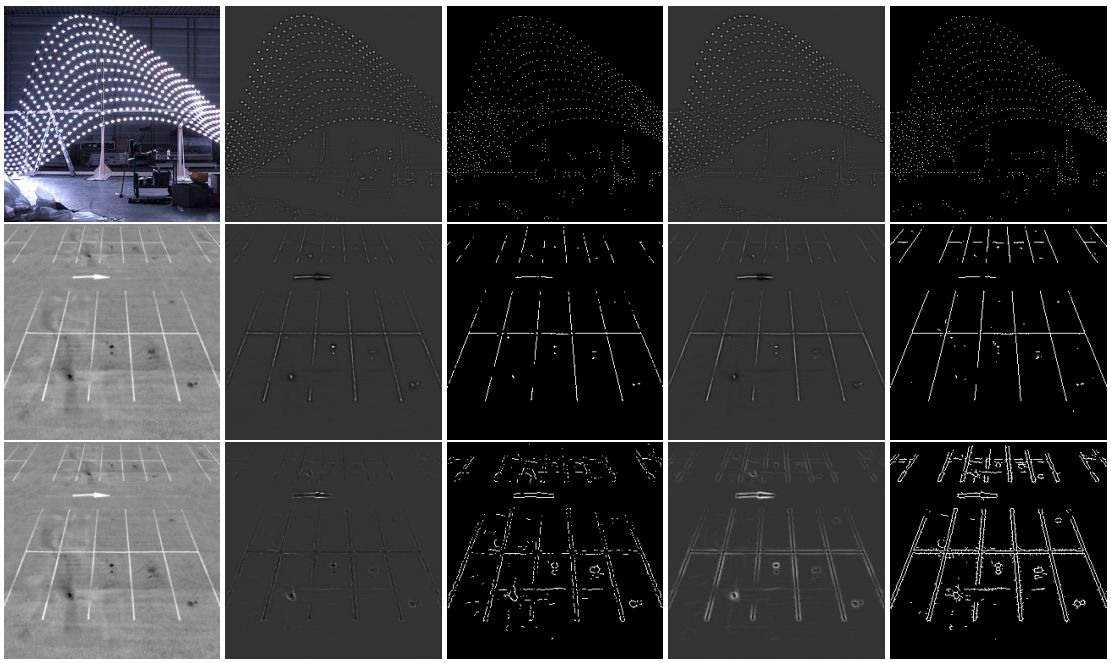


Figure 5.3: Example of real image spots, line, and edge detections (top to bottom). left to right: test image, log-likelihood (unoptimised), detection (unoptimised), log-likelihood (optimised), and detection (optimised).

5.4 Discussions

The result of the CVM operator on the spots detection falls shorts of the one found in the literature by van der Heijden (1992). The performance measure is different where this one uses FOM while the one in literature uses error rate. However, it still does not justify the difference in performance. This difference might be due to the difference in parameter while constructing the CVM operator in addition to not optimising the design parameter.

The line and edge detection, however, seems to roughly have the same performance as found in the literature. The literature uses FOM being divided over the size of the image, while the one used here is divided over the number of reference points. Should this be done in the same way, the line detection, for example, has a density of 0.04. This means that the result should then be divided by 25, which results in a very close result to the one found in the literature.

Most of the problem with the detection seems to come from spurious detection. Based on observations, it appears that the responses have some positive values after the inhibition distance. This caused a feature with a large amplitude to sometimes have a higher artefacts value than other small amplitude features. Optimisation on the weights fixed some of these issues, as seen in the better result on the optimised weights.

The real images application of the CVM operator goes mostly as expected with the results of the testing. This is shown in Figure 5.3 The spots detection are able to detect most of the spots, however, similar to the testing, optimisation does not seem to improve the detection. The line and edge detections perform as expected from the other testing where the optimised one perform better.

Some other discussions on the edge detector comparison can be seen in Appendix D

5.4.1 Limitations and Recommendations

Optimisation on the spots, line, and edge detection has only been done on the weights of the CVM operator. Similarly to the 1-dimensional case, optimisation can also be done on the para-

meter of the CVM. However, this is very computationally intensive, as it took several minutes to calculate all the integration needed for the CVM operator. The result in the 1-dimensional case shows that optimisation on weights and parameter results in approximately the same performance. This can still be validated by further experiment following the same procedure as the 1-dimensional case.

line and edge detections has been done, to extend this detection techniques further, the corner detection can be detected next. An idea for an approach would be to change the model shape functions into a corner shape. This possibility has been explored and can be seen in Appendix C, however it does not seems to be very promising.

Another idea for the approach, taking inspiration from popular corner detection method such as Harris corner detection (Harris and Stephens, 1988), is to make use of two-directional line detection in x and y direction to detect the corner. An experiment has also been done by van der Heijden (1992) which shows the possibility of using two directional CVM operator in x and y direction to get a complete edge detection.

The log-likelihood ratio of each directional line detection can be multiplied with each other, which theoretically will result in a peak at the corner location. The result of this operation is then filtered with a Gaussian filter. The result should have a peak value at roughly the location of the corner. Early experimentation of this can be seen in Figure 5.4. However, this produces a lot of localisation errors and still requires a lot more refinement on the approach.

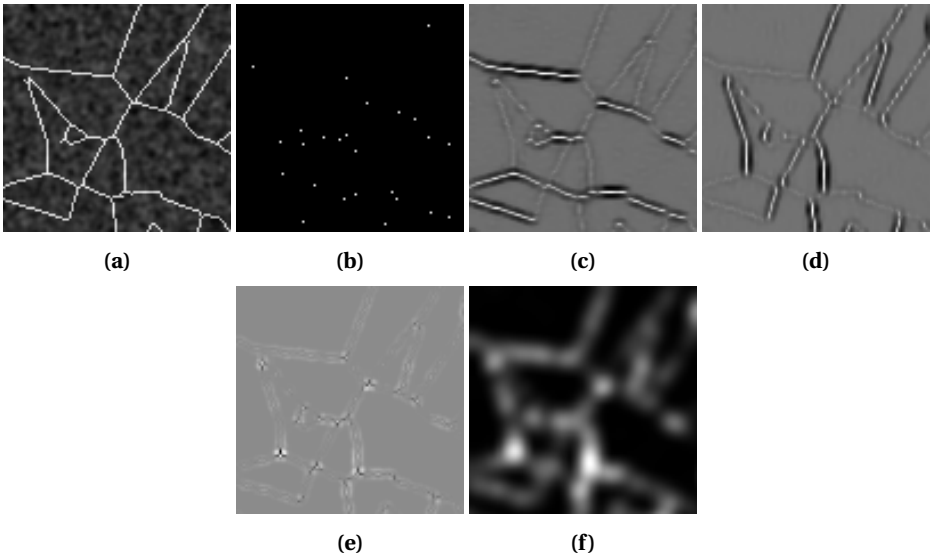


Figure 5.4: Early experiment on two directional line detector to find corner. (a) image, (b) reference, (c) x direction line detection, (d) y direction line detection, (e) combined llr, (f) gaussian filtered

5.5 Conclusions

This chapter aims to recreate and re-validate the result of the CVM operator on spots, line, and edge detection. The second goal is to see if the CVM operator can be moulded into corner detection. Both of these goals have been explored in this chapter.

The spots, line, and edge detection has been recreated, and it performs in roughly the same order of magnitude as found in the thesis of van der Heijden (1992). It has an excellent performance in general. The main problem found is when the convolution kernel leaves behind some artefacts, which became detected when the amplitudes of the other events are not high enough. However, this problem is commonly found in feature detection when the amplitude of a feature is not high enough. Numerical optimisation on the weights of the CVM operator

seems to improve this condition. The results show a significant improvement in performance compared to the detection without optimisation.

In the end, this chapter shows that the CVM operator still has good performance on the spots, line and edge detection. Numerical optimisation performed on the weights of this operator shows improvement to the performance. Several suggestions for further explorations have been mentioned in the discussion section.

6 2-Dimensional Data-Driven Feature Detection

6.1 Introduction

This chapter will explore the use of a data-driven method for feature detection in a 2-dimensional case. This relates to the main goal of this thesis, where the effects of combining model-based and data-driven approach were asked. As mentioned in Chapter 1, the research questions for this chapter are

1. To what extent can a data-driven approach, such as a convolution neural network, outperform the model-driven approach?

2. To what extent can a data-driven approach fed by information derived from a model-based approach outperform the purely data-driven approach or purely model-driven approach?

This chapter will start with the methods to answer the research questions. In this section, the analysis and experimental setups are provided. In the following section, the results of the experiments are presented. The discussion will follow, starting with the interpretation of the result. There will also be discussions on the limitation of the study along with a future recommendation. Finally, the conclusion will close this chapter.

6.2 Method

UNet is a fully convolutional network model, with a structure similar to shape U by Ronneberger et al. (2015). UNet is built with image segmentation and feature localisation in mind. It requires fewer training sets and has high segmentation accuracy. The architecture of UNet is shown in Figure 6.1. As seen from the figure, it is composed of an encoder and decoder, which are symmetrical with the symmetry axis of the intermediate layer. The encoder uses a convolutional layer to extract image feature, followed by a down-sampling with a pooling layer. The decoder conducts up-sampling of the feature images with the addition of a cross-layer connection, which helps to recover the details of the image.

In this particular implementation, the encoder extracts the information through 5×5 convolutional layer, ReLU function, and 2×2 max-pooling layer. The choice of 5×5 filter size instead of 3×3 is chosen as it provides a larger amount of information to be captured. The advantage is that it is likely to be more accurate, with the training time as a trade-off. Logically, a 5×5 with two depth should capture the same information as 3×3 UNet with three depths. The depth chosen for this experiment is two depth for 5×5 filter size. The decoder performs up-sampling by 2×2 transposed convolution layer and gradually recovers the image information. Finally, a softmax along with a classification layer is used for the decision making of the network.

The CVM operator results in a log-likelihood ratio, which is then used by placing a threshold to determine the location of the features. However, the log-likelihood ratio is only the final result of the process. The image is convoluted with each of the CVM operator convolution kernels to get the log-likelihood ratio. This convolution produces responses, which, when added up according to its weights, results in the log-likelihood ratio. In other words, the convolution kernels and the responses are also part of the CVM operator results.

An approach that can be taken aside from using the log-likelihood ratio is to make uses of the responses of the image to the kernels. These responses can be used as a multi-channel input to the UNet. This is likely to give more information and context to the network.

Another approach is to have the convolution kernels as pre-trained convolution network weights. This idea comes from the fact that convolution network weights often is of unrecog-

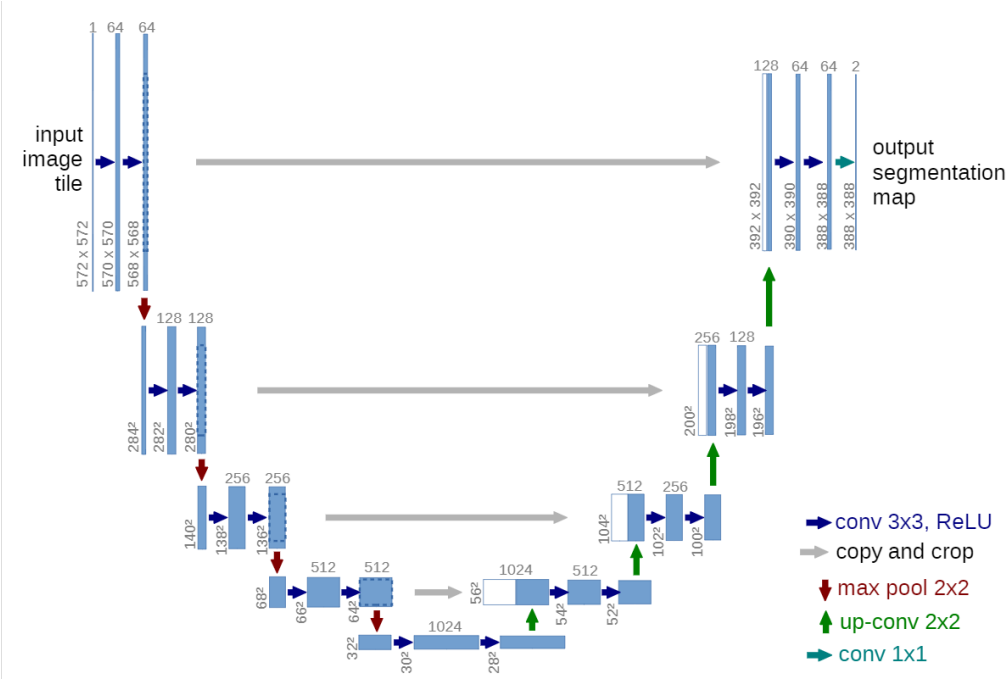


Figure 6.1: Architecture of UNet (Ronneberger et al., 2015)

nisable shape. This will explore the result if a known shape is used as a pre-trained parameter. This approach will hopefully make the training time faster, have better accuracy, and be more explainable.

6.2.1 Experimental Setup

The data set used in these experiments is based on the same one as used in 5.1. There are 2D Poisson Point Process, Mikado, and Voronoi for spots, line and edge detection. While a network can be trained to detect all three at once, the networks will be trained independently for each type of detection for this experiment. This is done so that the networks can have a comparable result to the CVM operator.

There will be several experiments conducted in this exploration. The first is the experiment done on the raw data trained with UNet. In this experiment, the images along with the references will be used to train the network to recognise the features. This will be the baseline performance of feature detection using deep learning methods.

Afterwards, experiments are conducted in implementing the CVM operator as part of the neural network. There will be three approaches to this. The first is done directly using the log-likelihood ratio of CVM as the input for the UNet. This is where the network will be trained using the log-likelihood ratio output of the CVM operator instead of the raw data.

The next one is done with responses of the CVM as the multi-channel input of the UNet. The dataset and UNet need to be modified to allow for this experiment. The data set will now contain nine-channel, where the first is the original test image, the last one is the log-likelihood ratio from CVM. Between them will be seven responses from seven of the CVM convolution kernels, which have the highest Batthacharya distance. An example of this data set on the line detection experiment is given in Figure 6.2. The network will also need to be adjusted so that it accepts nine-channel input. This can be easily done on the input layer of the network.

Finally, the convolution kernel of the CVM operator will be used as the weights of the first convolution network in the UNet. To do this, the convolution kernels are preloaded as the parameter for the convolution network layer. An example of the convolution kernels that will be

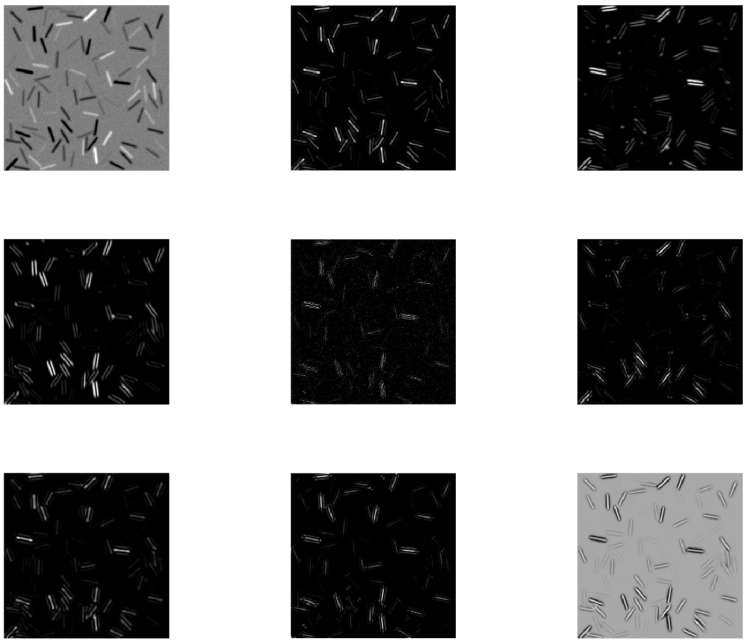


Figure 6.2: Multiple responses of line test image as multi-channel input to the UNet

used in the line detection experiment is presented in Figure 6.3. The rest of the experiment can be done in the same way as normal UNet training.

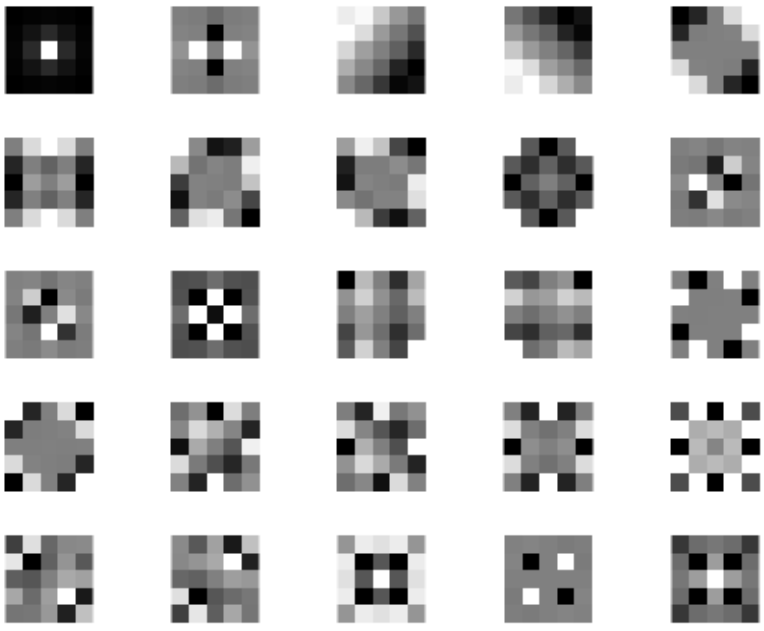


Figure 6.3: Convolution kernels of line CVM operator

Further details on the generation of the training data for all of these experiments can be seen in Appendix E.

The performance will be measured with FOM. This is done so that it can be compared with the CVM operator feature detector. In the deep learning method, the result will be skeletonised so that it shows only the most important component in case of multiple detections of a feature. This allows for a fairer comparison with the CVM operator.

The expected result of this experiment is the comparison of the results produced with these data-driven methods with the CVM operator for feature detection.

6.3 Results

The overview of the performance of various UNet approaches with all the feature detection can be seen in Table 6.1. There is also the result from the model-based approach, CVM operator, listed as a comparison.

Table 6.1: FOM value for various UNet and types of feature detections with additions of CVM performance

Approach		Spots	Line	Edge
UNet raw data	mean FOM	1.0766	0.3137	0.0707
	std FOM	0.0369	0.0275	0.0033
UNet log likelihood ratio	mean FOM	1.8822	0.4054	0.1115
	std FOM	0.1196	0.0509	0.0102
UNet multi-channel input	mean FOM	0.9003	**0.2040**	0.0899
	std FOM	0.0595	0.0321	0.0044
UNet pretrained weights	mean FOM	1.0003	0.2841	**0.0683**
	std FOM	0.0036	0.0241	0.0035
CVM	mean FOM	0.2090	0.3619	0.1640
	std FOM	0.0140	0.0269	0.0085
Optimised weights CVM	mean FOM	**0.2086**	0.3248	0.1181
	std FOM	0.0147	0.0247	0.0047

This result can be better seen in boxplots. This can be seen in Figure 6.4, 6.5, and 6.6 each corresponding to spots, line, and edge detection.

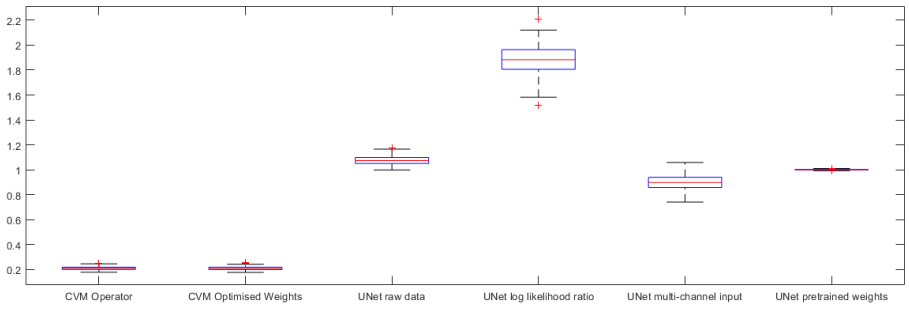


Figure 6.4: Boxplots of spots detection results

The examples of each of the approaches for spots, line, and edge detection respectively can be seen in Figure 6.7, 6.8, and 6.9. Each of the Figure contain eight subfigures, where each of them are (a) test image, (b) reference point, (c) CVM detection, (d) CVM optimised detection, (e) UNet raw image detection, (f) UNet log likelihood ratio detection, (g) UNet multi-channel input detection, and (h) UNet pretrained weight detection

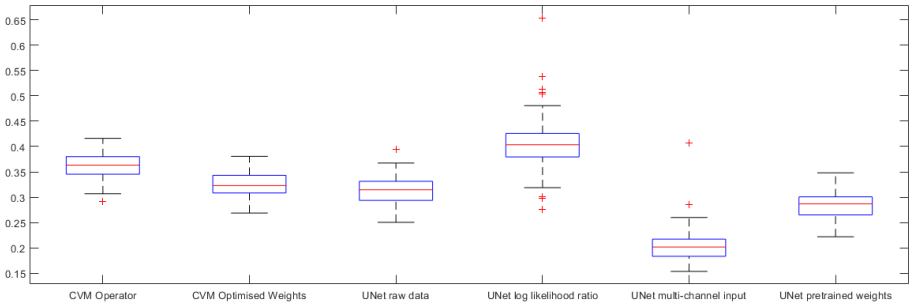


Figure 6.5: Boxplots of line detection results

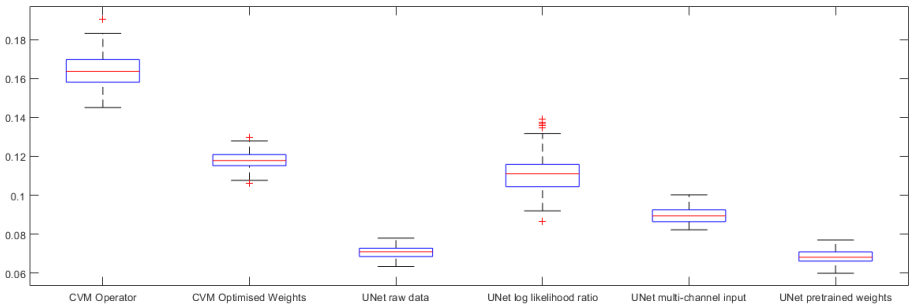


Figure 6.6: Boxplots of edge detection results

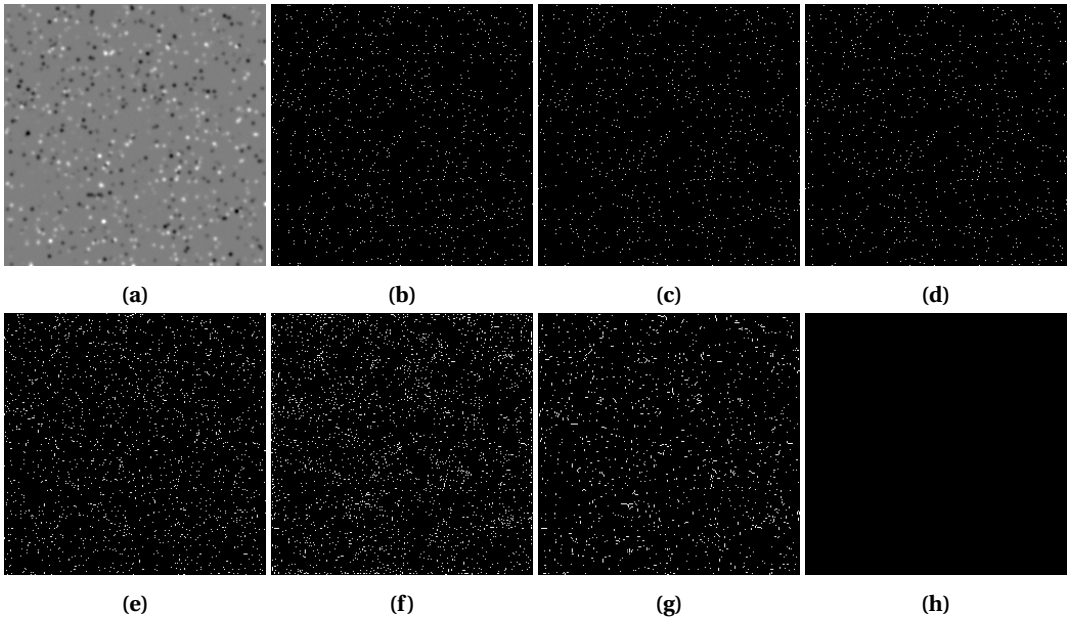


Figure 6.7: Result of spots detection with various approach, (a) test image, (b) reference point, (c) CVM detection, (d) CVM optimised detection, (e) UNet raw image detection, (f) UNet log likelihood ratio detection, (g) UNet multi-channel input detection, and (h) UNet pretrained weight detection

The pre-trained weights change from the original convolution kernel. These changes can be seen in Figure 6.10, 6.11, and 6.12 for spots, line, and edge respectively. Each of them contains two subfigure, the pre-trained weights which came from the convolution kernel of the CVM operator and then the weights after the network is trained. The figure shown has each of the weights value re-scaled to be displayed here.

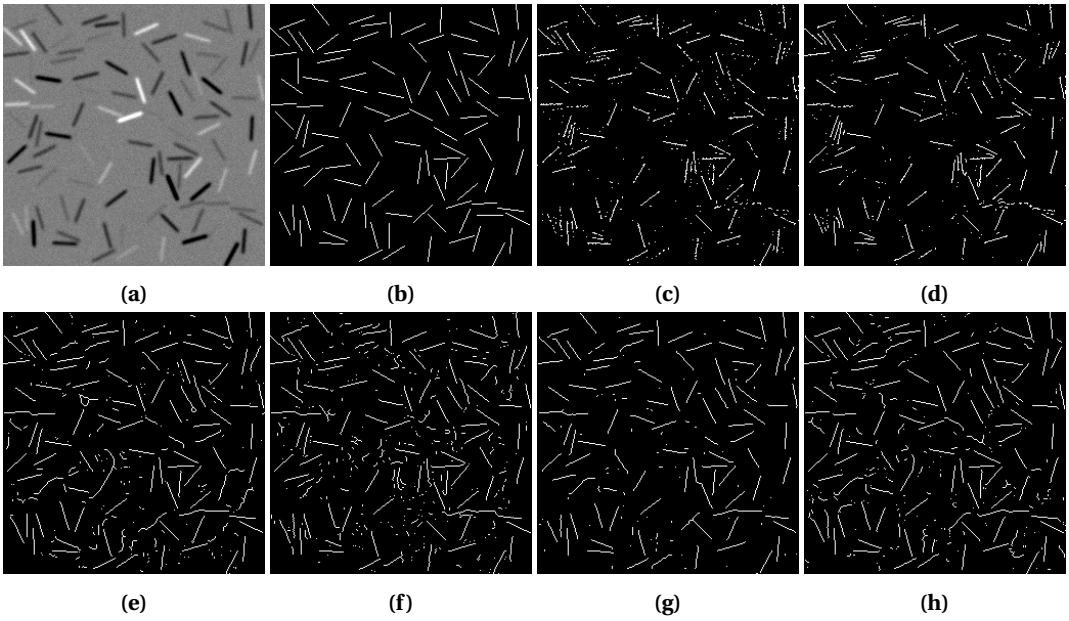


Figure 6.8: Result of line detection with various approach, (a) test image, (b) reference point, (c) CVM detection, (d) CVM optimised detection, (e) UNet raw image detection, (f) UNet log likelihood ratio detection, (g) UNet multi-channel input detection, and (h) UNet pretrained weight detection

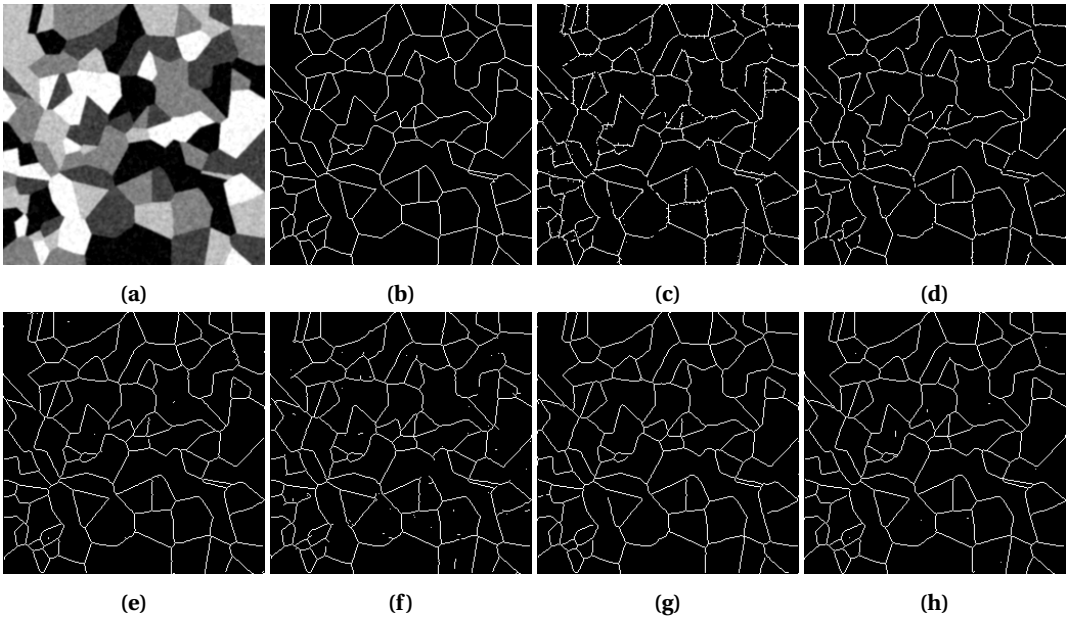
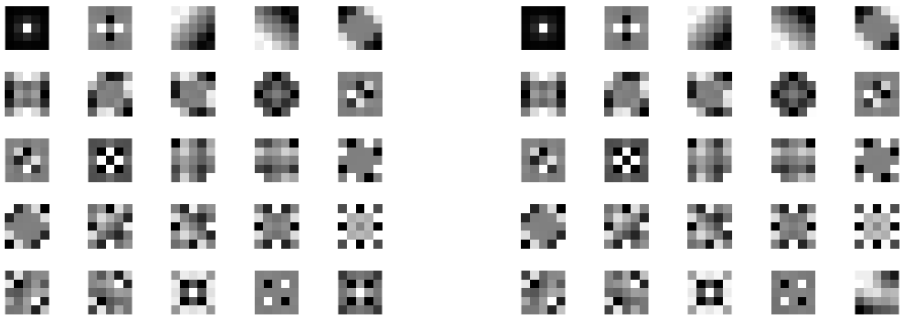


Figure 6.9: Result of edge detection with various approach, (a) test image, (b) reference point, (c) CVM detection, (d) CVM optimised detection, (e) UNet raw image detection, (f) UNet log likelihood ratio detection, (g) UNet multi-channel input detection, and (h) UNet pretrained weight detection

6.4 Discussions

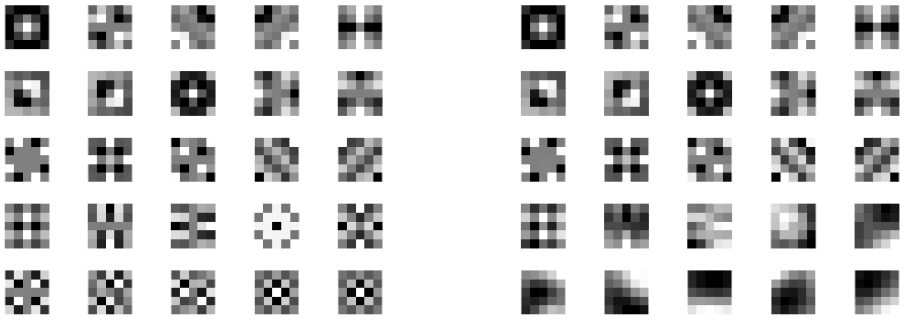
UNet spots detection are not successful. The result shows either spurious detections or simply no detection. This behaviour has previously been observed in the 1-dimensional CNN. The primary cause of this is likely to be the large gap between the class label amount. Improvement was not achieved even when the weights of the classification layer were adjusted.

The performance of line and edge UNet is comparable or better than the CVM operator, whether it is optimised or unoptimised. This can be seen in Figure 6.5 and 6.6. The result



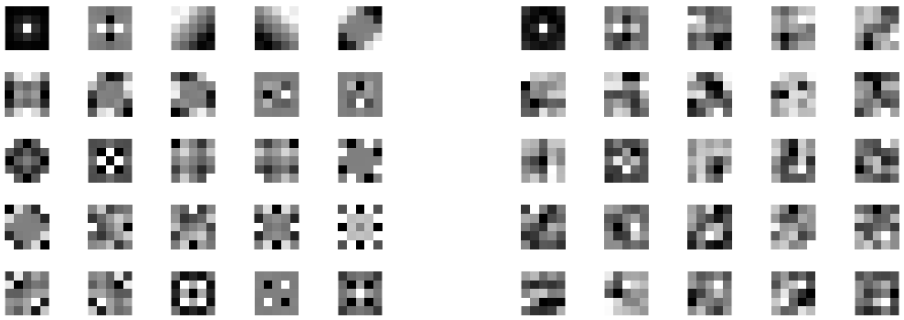
(a) pretrained weights **(b)** weights after training

Figure 6.10: Comparison of spots pretrained weights before and after network training



(a) pretrained weights **(b)** weights after training

Figure 6.11: Comparison of line pretrained weights before and after network training



(a) pretrained weights **(b)** weights after training

Figure 6.12: Comparison of edge pretrained weights before and after network training

shows that, especially in edge detection, UNet outperform the CVM operator. Meanwhile, in UNet line detection, the result is more comparable.

Incorporating CVM operator intermediate results into UNet has shown improvement in terms of more robust, decisive results. This can be seen in Figure 6.8 and 6.9 where (g), the multi-channel input approach has way less spurious detection compared to (e), the UNet on raw data. The multi-channel input approach has less spurious detection compared to the rest of

the approaches. This is likely to be the direct consequences of more prior information given to the network when compared to the rest of the network.

The pre-trained weights on the first convolution mean that result of the first convolution is similar to the log-likelihood ratio. However, as the weights can change, the result shows that it can perform better than simply feeding the log-likelihood ratio from the CVM operator to the network. This is due to the logical belief that the log-likelihood ratio approach causes some information loss due to model inaccuracies of the CVM operator itself. This is validated in Figure 6.5 and 6.6 where the performance of the pre-trained weights outperform the log-likelihood ratio approach.

One of the main ideas of using the pre-trained weights approach is to make the UNet more understandable. However, observing the changes in the weights, the results are not very recognisable. The spots detection weights stay mostly the same in Figure 6.10, but the classification layer decided to classify everything into one class. As shown by no detection in the result. The weights of line and edge detection changes a lot as seen in Figure 6.11 and 6.12. However, the result is not really recognisable and, unfortunately, is not understandable, which does not help in further understanding of the UNet.

The approach of multi-channel input seems to perform the best when considering that it has considerably fewer spurious detections. In the line detection it clearly has a better result as the boxplot in figure 6.5 shows. Although the FOM performance is slightly lacking in edge detection, it is within a reasonable range as all of them actually perform quite well. It has less spurious detection than the rest of the approach, so if a slight localisation error is better, this approach can be considered better than the other approaches. This type of improvement when incorporating a model-based approach into a data-driven approach similar to the findings in several papers, such as Oktay et al. (2018) and Lee et al. (2019).

6.4.1 Limitations and Recommendations

The spots detection with UNet did not works. As the main reason is believed to be the large gap in the number of the two labels, further exploration could be done to circumvent this issue. One such investigation could be done with blob detection, where multiple spot labels are clustered together instead of stand-alone spots that it is currently. After the detection, a shrink operation could be done to the result to get the spots. Further tuning on the hyperparameters could also be done to see if it could resolve such issues.

The current experiment was done only on 5×5 UNet with a depth of only two. This is not a very deep network. The maximum size of the filter in this method is only 7×7, which sometimes might not be enough for wider line segments or bigger spots. Therefore, further exploration could be done on UNet architecture with more depths. This will also further increase the possible tuning as there is more convolution layer within the network. However, this exploration comes at the cost of computation time.

A basic tuning of hyperparameters was done in this experiment. However, it is believed that it is still not optimal. Further exploration of the performance limits could be done using optimisation of the hyperparameters using grid search or bayesian optimisation.

6.5 Conclusions

This chapter aims to explore the performance of the data-driven approach compared to the CVM operator. The performance of pure data-driven and incorporation of CVM operator has been explored, and the results have been discussed.

In general, the performance of the data-driven method, in this case, UNet, is comparable to the model-based approach, the CVM operator. The spots detection does not perform well with UNet and causes spurious detections or no detection; this problem has been discussed in the

previous section. The line and edge detection of UNet is slightly better, although not very significant, compared to the CVM operator.

Incorporation of CVM results into the UNet shows improvement in performance. The main approach that works well is the multi-channel input approach where the image, multiple responses out of CVM convolution kernels, and the log-likelihood ratio of the CVM is concatenated into multi-channel data. With this approach, the spurious detection is observed to be much less compared to the other approaches. This good result is likely due to the amount of prior information available to the network.

The other approach of incorporating the CVM operator into UNet seems to be comparable to the performance of UNet on its own. Training the network on just the log-likelihood ratio appears to not be the optimal choice in this case, likely to be caused by the information loss due to the inaccuracy of the CVM operator model. The pre-trained weights perform well and perform comparable, if not slightly better, to the UNet on the raw image. However, some of the resulting weights after training have changed into unexplainable shapes, which does not help understand the UNet further.

7 Covariance Model as a Feature Descriptor

7.1 Introduction

Key points detectors are often used in conjunction with descriptors to associate an image to another geometrically. This process is called feature matching. CVM has been able to produced keypoints, especially in terms of line and edge. This chapter explores the possibility of using the results, including any intermediate results of the CVM Operator, to be used as a descriptor for feature matching.

This chapter attempts to explore the possibility of using CVM operator output to construct a descriptor. These are the research questions of this chapter,

1. To what extent are the features that are intermediate results of the CVM operator usable as a descriptor?
2. Can these descriptors be made rotational invariant?
3. Can these descriptors be made scale invariant?
4. To what extent can the features/descriptors of the CVM operator be used for keypoint matching?

This chapter will start with the methods to answer the research questions. This method section contains the analysis and experimental setups. The results of the experiments are presented in the following section. The discussion will follow, starting with the interpretation of the result, then discussions on the limitation of the study along with a future recommendation. Finally, the conclusion will close this chapter.

7.2 Method

7.2.1 Feature Descriptor and Matching

There are generally two methods to represent images, global descriptor and local descriptor. In this experiment, the local descriptor will be used. Feature descriptors encode interesting information about a feature into a series of numbers. That is, the descriptor is a vector characterising local visual appearance or local structure of image's patches (Hassaballah et al., 2016). The descriptor acts similarly to a numerical 'fingerprint' to differentiate one feature from another.

In the most simple situation, a descriptor could be the neighbourhood of the local feature. However, a more versatile descriptor in terms of scale and rotation invariant often used a derivation of the image on the neighbourhood or vectors representing the region's neighbourhood.

Feature matching is an act of establishing a relation between two images of the same scene. This is done by making use of the previously mentioned feature descriptors. Feature descriptors from two images are extracted to be compared with each other. This results in matching features that can further be used, such as image registration, image stitching, camera calibration, and object recognition.

Feature matching can be done in many ways. It is often optimised for specific descriptors to ensure the best result. In general, the approach of feature matching can be divided into two main approaches, brute force matcher and nearest neighbour matching (Hassaballah et al., 2016). Brute force matcher can be quite simple in that it mainly makes use of the sum of absolute difference or sum of squared difference. Meanwhile, nearest neighbour matching has some most efficient algorithm: the randomised k-d forest and the Fast Library for Approximate

Nearest Neighbours (FLANN). Moreover, RANSAC optimisation can be used to filter the outlier in the results.

7.2.2 Scale and Rotation Invariant

From the literature, scale invariant was seemingly only achievable when the descriptors are in scale with the information of scale-space extrema (Hassaballah et al., 2016). The neighbourhood in the descriptor needs to represent a magnitude in the same scale of the scale-space extreme so that it can be compared to another scale-space extrema detection and still produce a match. This is not something that can be directly applied using the intermediate result of the CVM operator. Therefore, the descriptor is likely not to be scale invariant. An attempt to resolve this issue is to have multiple features at a location containing information of different scale.

Rotation invariant is often achieved by making use of the gradient of the image descriptor, such as in HOG. However, rotation-invariant can be quite easily achieved by orientating the descriptor into the dominant gradient in a general use case. For example, a gradient of black to white can be rotated so that the white always face upwards. If done to all the descriptors, all descriptor should be rotation invariant. Therefore, the descriptor is likely possible to be made into rotation invariant.

This, however, has to be accompanied by a process that is also rotation invariant. If the image is convoluted with a non-symmetrical shape, then the resulting convolution might not be rotation invariant. This can be seen in the following example shown in Figure 7.1. The top row contains the convolution kernels, the left columns are the image and the rotated image. The rest of them shows the convolution results. It can be seen that with the convolution kernel two and three, when the image is rotated, the convolution result differs. This can be seen in the last row, where, ignoring the convolution at the edge, the non-symmetrical convolution results in a different value when the convolution kernel is non-symmetrical.

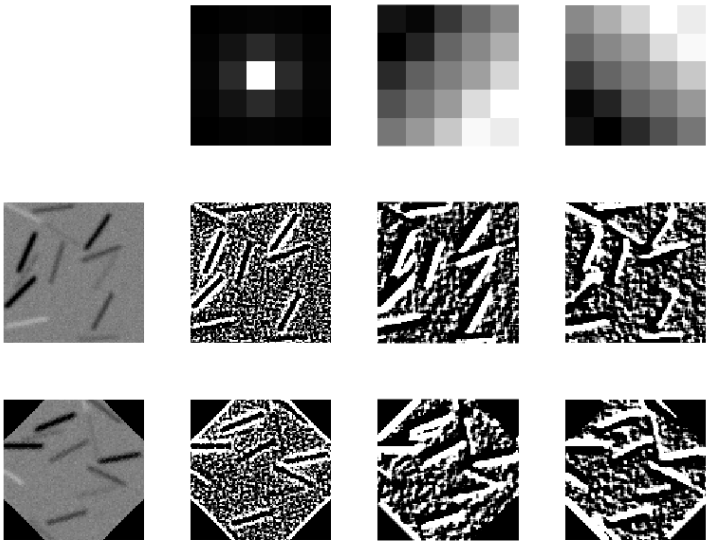


Figure 7.1: Examples of non rotation invariance with non symmetrical kernel. top row: convolution kernels, left column: original image

Therefore, when applied globally, convolution kernels need to be symmetrical. Another option would be to only apply the convolution on the local image patch that has been rotated accord-

ing to the local image patch. This ensures that the convolution response is the same, which also allow for a more diverse convolution shape to be used.

7.2.3 CVM Feature Descriptor

The CVM feature descriptor approach can be defined as bins of magnitudes, making use of convolution results as a bin of information. This method was inspired by the use of bins in SIFT descriptor (Lowe, 1999). However, instead of storing gradients in the descriptor as in SIFT, the convolution results are arranged into feature vectors which later are concatenated to each other, making the complete feature descriptor.

The computational structure of the CVM feature descriptor can be seen in Figure 7.2. It starts with sampling the points around the keypoint detected. Afterwards, a limited scale invariant should be able to be achieved by re-sampling the image patch at a different scale. Then, to achieve rotation invariance, the local image patch needs to be aligned to the dominant gradient. This is first achieved by getting the orientation by making use of a histogram of the gradient within a circle fitted into the image patch. This is to make sure that the process of alignment on other orientation results in the same histogram. With the histogram, the dominant gradient orientation is identified. Should there be more than one dominant gradient, it can be converted into a new feature, each with its respective orientations.

Afterwards, sampling is done at the centre of the aligned image to prevent artefacts from rotation. The resulting image patch will go through a set of convolution. This is where the CVM operator contributes, the CVM operator convolution kernels are used for the convolution. Each convolution kernel will act as a bin, where the convolution result will then be resized/pooled into a 4×4 patch. This will then be reshaped into a 1×16 feature vector. Should there be eight convolution kernels, this will result in 128 element vector. The result would then be normalised so that it can be invariant to illumination.

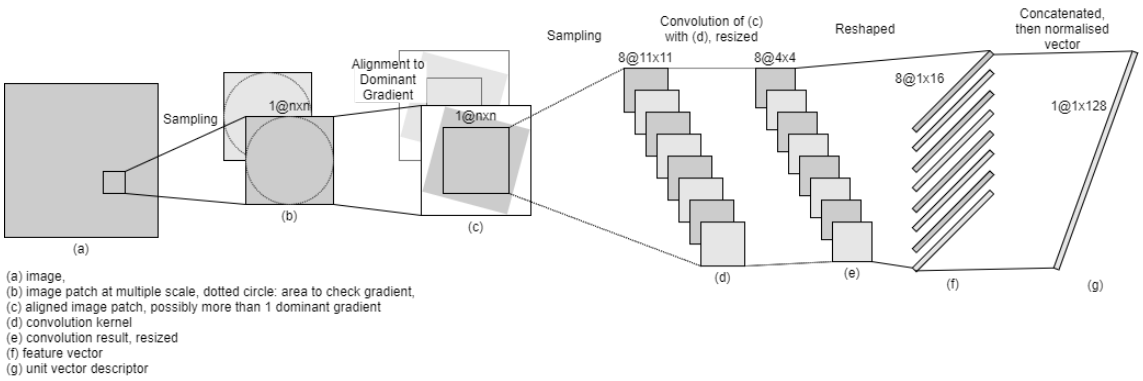


Figure 7.2: Computational Structure of CVM Feature Descriptor

The kernels that will be used for the CVM feature descriptor is from the 11×11 CVM operator. This can be seen in Figure 7.3. It was chosen based on the most distinct shape from the line CVM operator kernel.

7.2.4 Experimental Setup

Invariant Descriptor

The feature descriptor's experiment will use a set of test images constructed by a 2-dimensional multivariate Gaussian shape. Figure 7.4 shows various example of such test image. The test will be done by comparing just one feature descriptor original Gaussian shape to the rest of the varied Gaussian shape. This will allow for checks on rotational and scale invariant of the

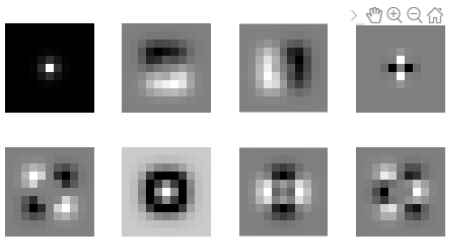


Figure 7.3: CVM kernels used for feature descriptor

descriptor due to the varied parameter of the Gaussian shape. This can be seen in the computational structure of the test in Figure 7.5.

Figure 7.4: Gaussian shape with varied parameters. left : original, the rest has varied rotation, scale, and illumination

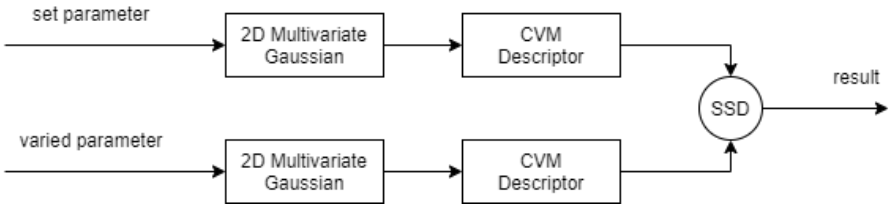


Figure 7.5: Descriptor test computational structure

The illumination transformation will be a change of amplitude with a range from 0.5 to 2.5. The rotation transformation will range from 0 to 180 degrees since it is symmetrical. The scale transformation will range from 0.3 to 2.8. The randomisation here is done with uniform distribution. The performance will be evaluated on the sum of squared differences on various Gaussian parameter. This will show the similarity and difference of them when they are at different angle and scale. The test will be done with 100 test images for each parameter and 100 for any combination of them. The expected result will be in the form of a table showing the sum of squared differences of illumination, rotation and scale transformation, and the combination of them.

Keypoint Matching

Keypoint matching can be used to assess the usefulness of the feature descriptor by using it for keypoint matching. This experiment makes use of the existing image, in this case, the Voronoi as seen in Figure 7.6. The image will be rotated and scaled at random to check the performance of the descriptor for keypoint matching. The keypoint detection will make use of Harris corner detection (Harris and Stephens, 1988).

A good descriptor for keypoint matching has a high number of correct matches on the operation. This can be tested by counting the number of correct matches, a similar performance metrics to that of Bekele et al. (2013). However, automating this on its own is hard to do. A

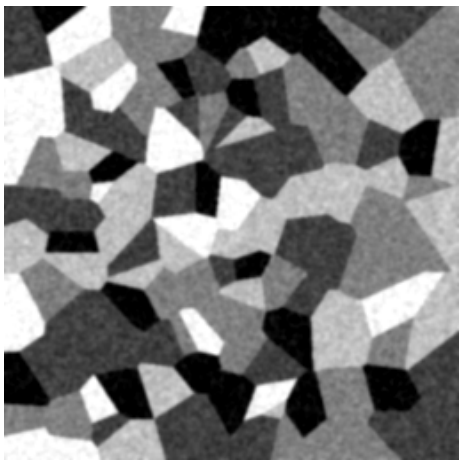


Figure 7.6: Example of figure to be used for keypoint matching

great number of matches can happen, while it might be all incorrect matches. There is a need to make sure that the matches found are correct matches.

Keypoint descriptors can also be used for the calculation of the displacement field where the image can be rotated back to be the original image. The resulting image of this operation can be evaluated by making use of the FOM evaluation. In this way, the number of correct matches can be compared by assessing the correct matches as well as the resulting transformation. The computational structure of this test can be seen in Figure 7.7.

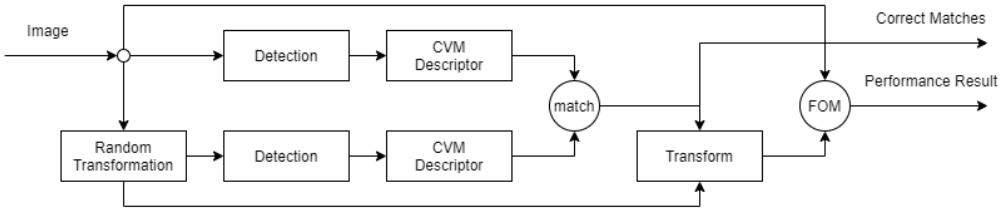


Figure 7.7: Keypoint matching test computation structure

In this experiment, initial detection of 50 corner keypoints is recorded using Harris corner detection (Harris and Stephens, 1988). The detected keypoint will be transformed along with the test image so that the keypoint locations stay the same relative to the images. The number of correct matches will be the amount of inlier match, that is, the matches on the same displacement field compared to the original image.

The transformation range is 0.3 to 2.8 for scale transformation and 0 to 180 degrees for the rotation transformation. The test will be done 100 times for each of the transformation types and the combination of the transformation.

7.3 Results

The overview of the performance of feature descriptor on various invariance testing can be seen in Table 7.1. This result can be better seen in a boxplot which is provided in Figure 7.8.

Table 7.1: SSD of various feature descriptor invariance testing

	mean	std
Illumination	1.0e-31	3.0e-32
Rotation	0.0117	0.0067
Scale	0.0076	0.0062
Illumination and Rotation	0.0094	0.0057
Illumination and Scale	0.0073	0.0063
Scale and Rotation	0.0198	0.0125
Illumination, Rotation, and Scale	0.0177	0.0121

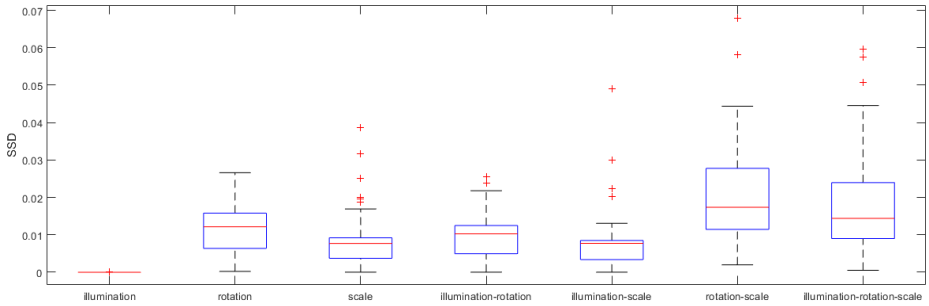


Figure 7.8: Boxplot of various feature descriptor invariance testing

Furthermore, details on the rotation and scale performance can be seen in Figure 7.9 and 7.10. These figures shows the difference of the descriptor at a certain rotation or scale interval.

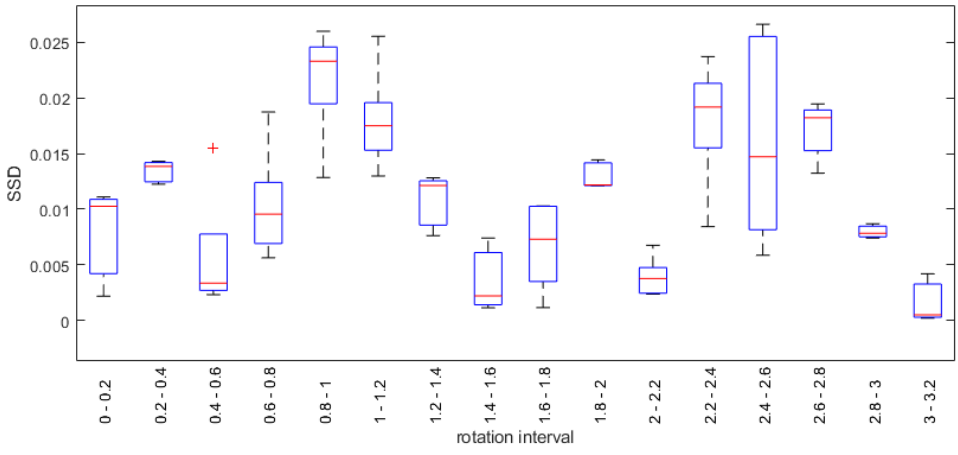


Figure 7.9: Boxplot of feature descriptor at a certain rotation interval $(0 - \pi)$

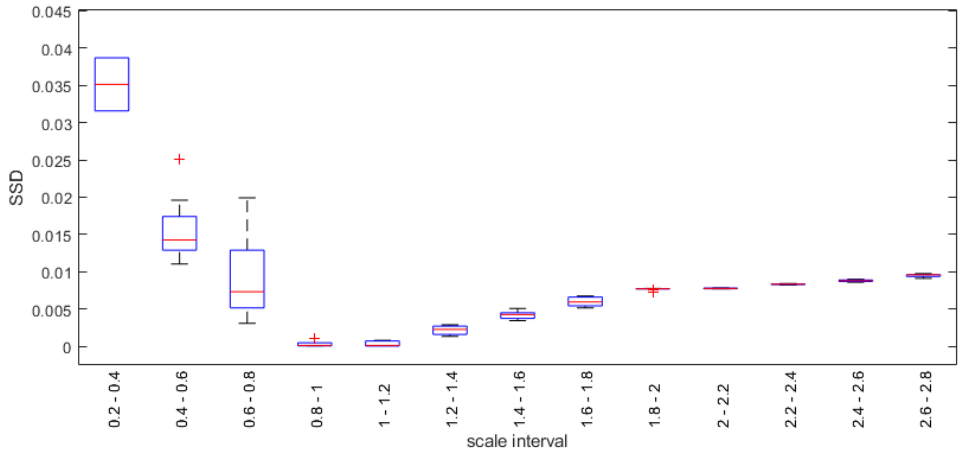


Figure 7.10: Boxplot of feature descriptor at a certain scale interval (0.3 – 2.8)

Next, is the result on the keypoint matching, the overview of the performance can be seen in Table 7.2. The boxplot of the results is also provided to provide more insights into the result of this experiment. This boxplot can be seen in Figure 7.11

Table 7.2: FOM of various keypoint matching tests

	mean	std
Rotation	0.0106	0.0031
Scale	0.0190	0.0846
Scale and Rotation	0.0131	0.0155

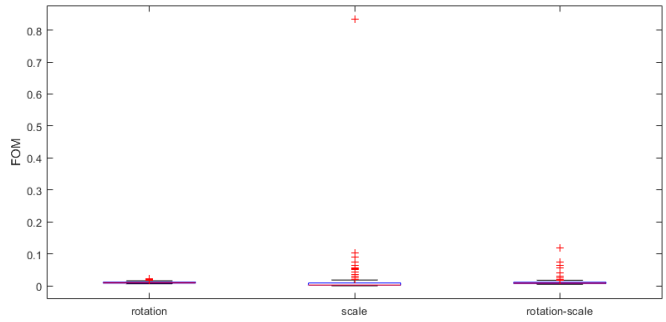


Figure 7.11: Boxplot of keypoint matching performance with various parameter change

Furthermore, the results of the scale and rotation transformation in details can be seen in Figure 7.12 and 7.13. This figure shows the performance of the keypoint matching at a certain scale interval.

Some examples of these keypoint matching can be seen in Figure 7.14. The figure shows the scaling of the test image in text above them. The resulting FOM performances are also listed above each figures.

The number of matches found that is used for the displacement field can be seen in Table 7.3. Based on Figure 7.14, it is reasonable to say that a FOM value of up to 0.1 is a localisation error, and afterwards, it is just a wrong classification. In this table, n refers to the amount of test that falls in the category.

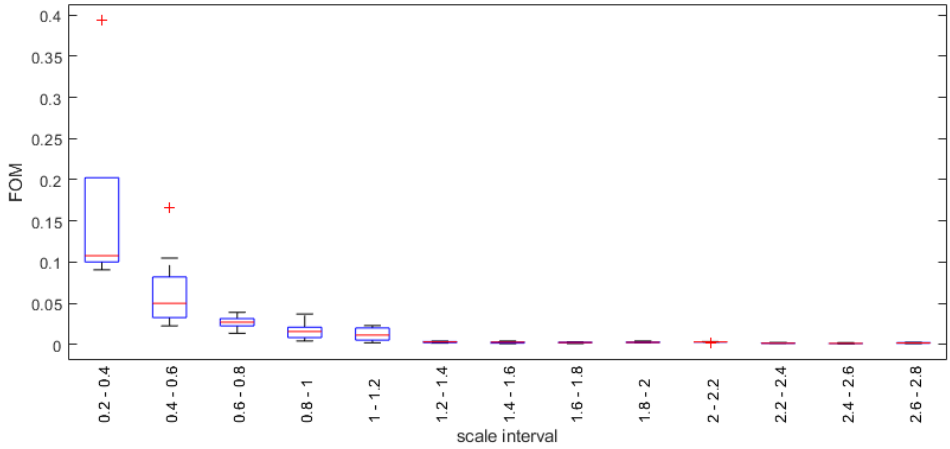


Figure 7.12: Boxplot of scale transformation keypoint matching performance

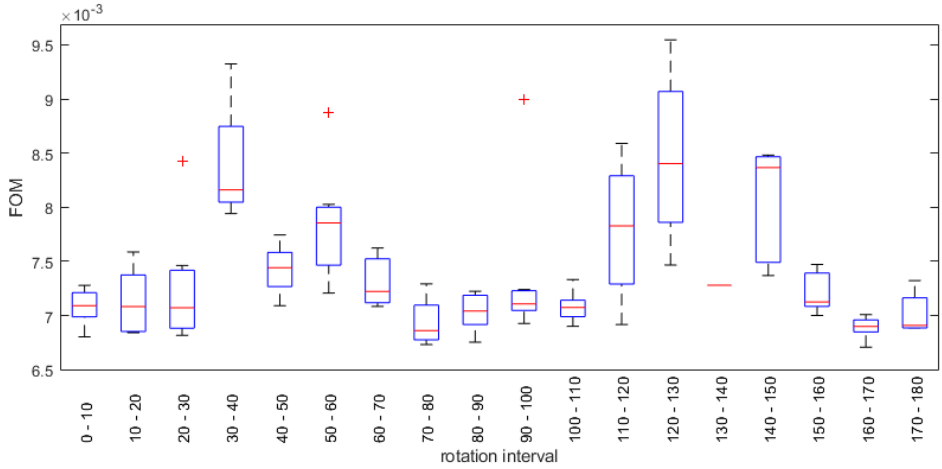


Figure 7.13: Boxplot of rotation transformation keypoint matching performance

Table 7.3: Amount of matches (mean) on the same displacement field for the transformation

	All	FOM≤ 0.1		FOM> 0.1	
	mean	amount	mean	amount	mean
Rotation	49.71	100	49.71	0	
Scale	43.13	98	43.87	2	7
Scale and Rotation	39.46	99	39.78	1	8

Finally, there are some keypoint matching examples on real images. The first one is done by taking two part of the same image and do keypoint matching on them. The second and third examples were done by taking a perspective change. These can be seen in Figure 7.15.

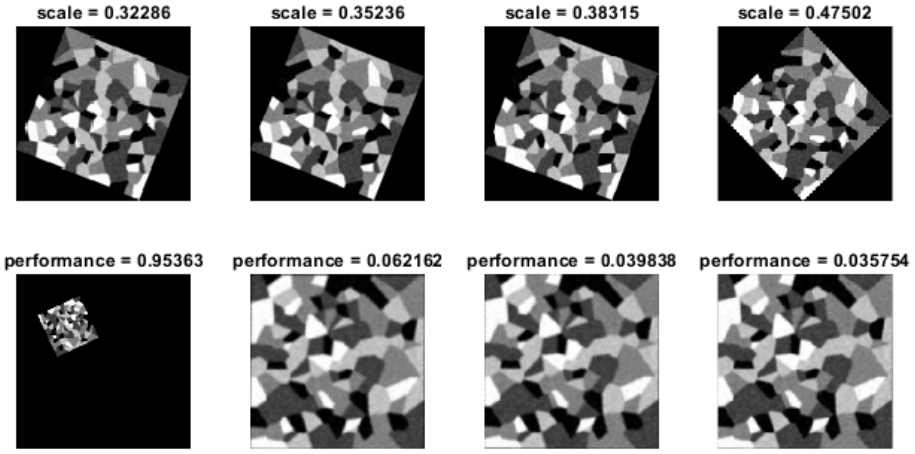
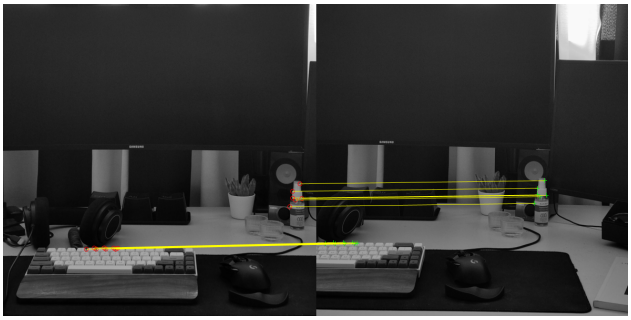
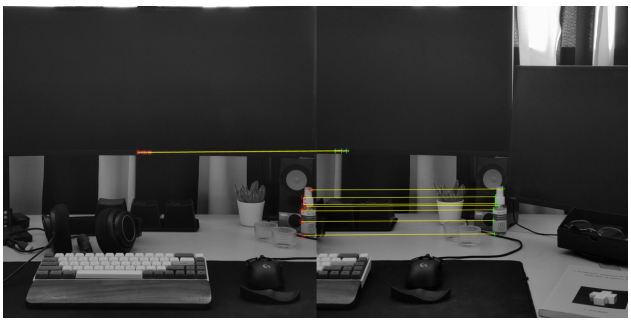


Figure 7.14: Keypoint matching results examples, top: transformed test image, bottom:resulting transformation back

(a) keypoint matching on parts of same image.



(b) keypoint matching with perspective change 1



(c) keypoint matching with perspective change 2

Figure 7.15: Keypoint matching real images examples

7.4 Discussions

From the result, as seen in Figure 7.8, it can be seen that the attempts to have illumination, rotation, and scale invariant feature descriptor is successful. The alignment to the dominant gradient and the normalisation to the feature descriptor has allowed this to be possible. Some results show as an outlier in the rotation transformation. This might be due to the low pixel amounts, which causes some inaccuracies while aligning to the dominant gradient. This can be seen in Figure 7.9, where rotation at approximately 45 degrees angle has worse performance compared to 90 degrees. This is likely due to the pixelation of the image, rotation at an irregular angle results in loss of information due to the discrete nature of pixels.

The limited scale invariant seems to work well in the test. This is still within the assumption that the scale difference is at most two as designed. This can be seen in Figure 7.10 when the scale is above two, the descriptor starts to differ again. However, on a scale of one to two, the differences between the descriptor are small. This is most likely because the resize into a 4×4 bin size of the feature descriptor compressed the differences in the simple shape. In actual image testing, there might be more differences in this range.

Artefacts hinder the experiment on keypoint matching due to several reasons. First, the inaccuracies brought in by the rotation done to the image might cause localisation error on the keypoint detection. This leads to the second reason: the slightest shift in the image will cause some differences in the images as all of the image will be detected as not localised properly. However, this is hugely improved when the FOM applies the low pass Gaussian filter to help with the localisation error.

The general results of the keypoint matching as seen in Figure 7.11 shows a good transformation into the original image. The keypoint matching on scale transformation indicates that there are some outliers. Even though everything is still in relatively good value, Figure 7.12 shows that the keypoint matching on a scale below one results in worse results. This might also be due to the pixelation problem of reverting a smaller image into a bigger image. The combination of the scale and rotation transformation also seems to work well, showing a good transformation back into the original image.

The amount of matches from the descriptor is good. The worst performance on the combination of scale and rotation transform still result in approximately 79% matches found. This is a very good performance on the test. The real-world scenario might differ, however, this is still a very good indication of the possibility that it can work well.

From the real images examples, it can be seen that it works well on the patches of a same image, however, perspective changes decrease the performance. Although it is worse, the amount of keypoint matched is still enough to do image registration or stereo vision on them. The performance on different perspectives is likely to be affected by the different corner point found as well. A different keypoint detection method might results into a different performance measure as well.

7.4.1 Limitations and Recommendations

The test on the CVM feature descriptor has been stand-alone. For further validation of these results, a further test should be done comparing the performance of the CVM feature descriptor compared to other descriptors in the same kind of rotation and scale invariant test. Further testing could also include the performance comparison in keypoint matching.

The current feature descriptor only encodes information in grayscale. An exploration into encoding colour information could also be explored further. Informations from colour could benefit the keypoint matching as it would allows for a more distinct descriptors.

The common method to construct a scale invariant descriptor is to have the magnitude of the feature in the scale-space extrema. The CVM feature descriptor can further be explored to accept such input and determine the image patch size automatically such that the resulting descriptor can be scale invariant.

7.5 Conclusions

The goal of this chapter was to explore the possibility of constructing a feature descriptor based on CVM. The design of the CVM feature descriptor has been done, and results have been discussed.

The designs of the CVM feature descriptor can be finalised. The resulting CVM feature descriptor has been tested and shows illumination, rotation, and limited scale invariant. This has been achieved through normalisation and alignment to the dominant gradient. A limited scale invariant has been achieved by taking multiple scaled descriptor at one location. Some ideas on achieving a more general scale invariant have been discussed for further exploration.

CVM feature descriptor has also shown the possibility of being used as a descriptor for keypoint matching. When used with K-d trees, it allows for sufficiently robust keypoint matching with a good amount of the descriptors matched correctly. This can work reliably with scale and rotation transformation. These matched descriptors can transform the test image into the original state correctly by calculating the displacement field.

Finally, to conclude, this chapter has shown the design of a feature descriptor that can use convolution as a filter for added information. In this chapter, the CVM operation kernel was used to do this exploration. Although it has shown some results as rotation and limited scale invariant descriptor, this is still a very early experiment. Further explorations can still be done on this subject, as mentioned in the discussions.

8 Conclusions

The main goal of this thesis is to investigate the incorporation of covariance model-based image feature detection and any of its intermediate results in combination with deep learning approaches. The hypothesis was this incorporation of covariance model-based method within a deep learning framework will improve the performance of feature detection. Then there is a secondary goal of a feasibility study of feature descriptor based on covariance model operator. All of these have been explored in this thesis, with details of these explorations discussed in the respective chapters.

The incorporation of covariance model-based within a deep learning framework has been discussed in Chapter 6. Other experiments in the previous chapters lead to the results and conclusions of this chapter. The result shows that the deep learning networks can take advantage and make use of additional prior information. The incorporation of the CVM operator into the deep learning framework makes for a more robust, less spurious line and edge detections. This finding is in line with the hypothesis that the performance of the deep learning framework is improved by incorporating CVM.

The secondary goal of exploration of CVM based feature descriptor has been discussed in Chapter 7. The design of the feature descriptor has been discussed and constructed. The resulting descriptor has shown that it is illumination, rotation, and limited scale invariant. Furthermore, keypoint matching using the descriptor has shown good results, where geometric transform of scale and rotation can be estimated between the two images.

To conclude, this thesis has presented the improvement that can be achieved by incorporating the CVM into a deep learning framework. This thesis also proposed a feature descriptor design that makes use of the CVM operator kernel. However, there are still many further explorations that can be explored following both directives. Within the incorporation of the CVM into the deep learning framework, explorations can be done on different network architectures. Meanwhile, the feature descriptor can be further explored by validating its performance to state-of-the-art. The design of the feature descriptor itself is still in the early stage, and further optimisation should be possible.

Acknowledgements

This thesis would have never been possible without the support and contribution of other people. Therefore, I would like to take some time to thanks these people.

Foremost, I would like to express my sincere gratitude to my thesis supervisor, dr.ir. Ferdinand van der Heijden, for the guidance and support throughout this thesis. He has been very accommodating and patient during this period, with constant help and suggestions on the methods and approaches found in this thesis. He has also been very flexible on the topics of this thesis. When there were unexpected changes in the situation due to Covid-19, he had provided alternatives. I have been deeply inspired by his ingenuity and work ethics in our time working together. It was a great privilege to have him as my supervisor.

I want to thank my fellow students on the 3DHAP projects, Mike, Stijn, Gideon, Ruud en Saket. Their meaningful discussions and feedbacks have been very valuable for the completion of this thesis.

I would also like to extend my gratitude towards my parents and families, who have supported me throughout my academic life. Also, I express my thanks to my friends, Renske, Gijs, Kevin, and Michael, who have been very valuable in keeping me motivated and helping me keep a sane mind through the challenging pandemic and thesis times.

A Setup for 1D Neural Network

The MATLAB code to construct one of the convolution architecture tried is provided in the following snippet. For the convolution network, important thing to note is that MATLAB does not allows semantic segmentation using 1-dimensional data, therefore, unfortunately, the architecture must be modified to accept $2 \times N$ data. The training data is simply the dataset concatenated to itself.

```
convnet = [
    imageInputLayer([2 5000 1],"Name","imageinput",...
            "Normalization","none")
    convolution2dLayer([2 20],20,"Name","conv1",...
            "Padding","same")
    reluLayer("Name","relu1")
    maxPooling2dLayer([2 2],"Name","maxpool", ...
            "Padding","same","Stride",[2 2])
    convolution2dLayer([2 20],20,"Name","conv2","Padding","same")
    reluLayer("Name","relu2")
    transposedConv2dLayer([2 2],20,"Name","trans_conv",...
             "Stride",[2 2])
    convolution2dLayer([1 1],2,"Name","finalconv")
    softmaxLayer("Name","softmax")
    pixelClassificationLayer("Name","classoutput",...
        "ClassNames",tbl.Name,"Classweights",inverseFrequency)];
```

The MATLAB code to construct the LSTM network is provided in the following snippet.

```
LSTMnet = [
    sequenceInputLayer(1,"Name","sequenceinput")
    bilstmLayer(200,"Name","biLSTM")
    fullyConnectedLayer(2,"Name","fc")
    softmaxLayer("Name","softmax")
    classificationLayer("Name","classoutput")];
```

B Block Toeplitz in the Autocovariance Function

A block toeplitz is a toeplitz matrix consisting of smaller block matrixes which are also a toeplitz matrix. A visualisation of such matrix can be seen here, where each of the A_{nm} is also a toeplitz on its own.

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \dots & A_{1n} \\ A_{21} & A_{11} & A_{12} & \dots & A_{1,n-1} \\ A_{31} & A_{21} & A_{11} & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & A_{12} \\ A_{n1} & A_{n-1,1} & \dots & A_{21} & A_{11} \end{bmatrix} \tag{B.1}$$

The autocovariance function is based on the neighbourhood measurement. Assuming this neighbourhood measurement is a $M \times M$ matrix, it will be an $M^2 \times M^2$ matrix in the autocovariance function. This is because the 2D matrix becomes a 1D vector as mentioned in Section 5. The neighbourhood matrix index can be seen in the following equation.

$$N = \begin{bmatrix} N_{11} & N_{21} & \dots & N_{i1} \\ A_{12} & N_{22} & \dots & A_{i2} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1j} & A_{2j} & \dots & A_{ij} \end{bmatrix} \tag{B.2}$$

i and j are indexes of the neighbourhood and has the same length M which forms a square matrix. This can be set into the block toeplitz form using the following visualisations, where $acf(C_x, R_x, C_y, R_y)$ is the input function. and C and R refers to the column and row of the neighbourhood index.

This can be better seen in the following visualisation.

$$A = \begin{array}{c} \\ C_y = 1 \\ C_y = 2 \\ \vdots \\ C_y = i \end{array} \begin{array}{c} \begin{array}{cccc} C_x = 1 & C_x = 2 & \dots & C_x = i \end{array} \\ \begin{bmatrix} [A_{1,1}] & [A_{2,1}] & \dots & [A_{i,1}] \\ [A_{1,2}] & [A_{2,2}] & \dots & [A_{i,2}] \\ \vdots & \vdots & \ddots & \vdots \\ [A_{1,i}] & [A_{2,i}] & \dots & [A_{C_x=i,C_y=i}] \end{bmatrix} \end{array} \tag{B.3}$$

and for each of the block matrix,

$$A_{C_x,C_y} = \begin{array}{c} \\ R_y = 1 \\ R_y = 2 \\ \vdots \\ R_y = j \end{array} \begin{array}{c} \begin{array}{cccc} R_x = 1 & R_x = 2 & \dots & R_x = j \end{array} \\ \begin{bmatrix} \ddots & \ddots & & \\ \ddots & \ddots & & \\ & & & A_{C_xR_x,C_yR_y} \end{bmatrix} \end{array} \tag{B.4}$$

Therefore, the rows and columns from the neighbourhood matrix is represented in the block toeplitz.

C CVM corner detection efforts

The corner detection can be done by making use of the line detection as the base. A corner can be defined as an intersection of lines. The first approach makes use of shape information. This can be applied to the CVM operator by modifying the model shape as well as the autocovariance function that comes with it. This is done by introducing another window function that limits the likelihood when there are no two line elements with different directions connected to the corner elements. This corner element is then modelled by the following equations.

$$s(x,y,\phi) = s_1(x\sin\phi + y\cos\phi)\,p(x\cos\phi - y\sin\phi)\,p(x\cos(\phi+\theta) - y\sin(\phi+\theta)) \tag{C.1}$$

Where θ refers to the difference of directions between the two lines. The autocovariance function is then affected by the model's changes, which results in the following sets of equations.

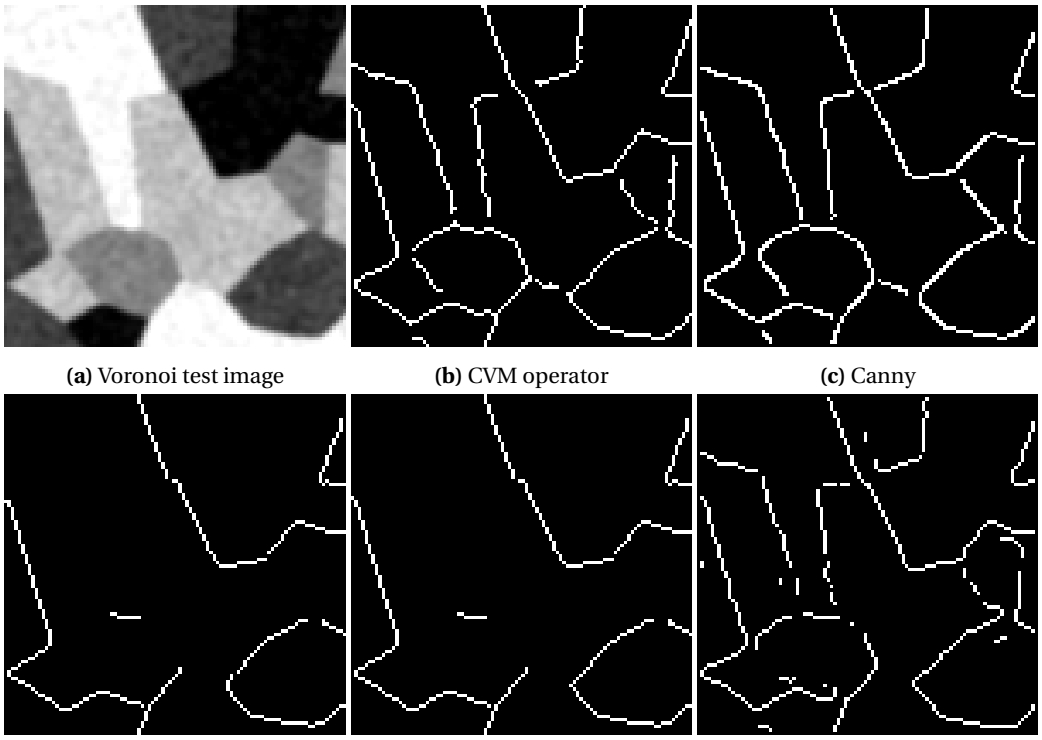
$$R_p(\vec{x}_1,\vec{x}_2|\vec{\xi}_k \notin A_{nm}) = \frac{\sigma_a^2\lambda}{|\Phi||\Theta|}\int_{\vec{\alpha}=-\infty}^{\infty}\int_{\phi\in\Phi}\int_{\theta\in\Theta} s(\vec{x}_1-\vec{x}_2+\vec{\alpha})\,s(\vec{\alpha})\,d\theta\,d\phi\,d\vec{\alpha} - \frac{\sigma_a^2\lambda}{|\Phi||\Theta|}\int_{\vec{\alpha}\in A_{nm}}\int_{\phi\in\Phi}\int_{\theta\in\Theta} s(\vec{x}_1+\vec{\alpha})\,s(\vec{x}_2+\vec{\alpha})\,d\theta\,d\phi\,d\vec{\alpha} \tag{C.2}$$

$$R_p(\vec{x}_1,\vec{x}_2|\vec{\xi}_k \notin A_{nm}) = \frac{\sigma_a^2\lambda}{|\Phi||\Theta|}\int_{\vec{\alpha}=-\infty}^{\infty}\int_{\phi\in\Phi}\int_{\theta\in\Theta} s(\vec{x}_1-\vec{x}_2+\vec{\alpha})\,s(\vec{\alpha})\,d\theta\,d\phi\,d\vec{\alpha} - \frac{\sigma_a^2\lambda}{|\Phi||\Theta|}\int_{|\vec{\alpha}-\vec{\xi}|<R_i}\int_{\phi\in\Phi}\int_{\theta\in\Theta} s(\vec{x}_1+\vec{\alpha})\,s(\vec{x}_2+\vec{\alpha})\,d\theta\,d\phi\,d\vec{\alpha} \tag{C.3}$$

The rest of the steps follow through the spots and line detection. The detection of corner elements is decided by the local maxima of the log-likelihood ratio for which it is above a certain threshold value. This is the same classification as the spots detection.

D Edge detector showcase

This appendix shows the detections done by various methods. This is done for qualitative evaluation on the differences between the detectors. The detectors used are CVM operator, Canny, Prewitt, Sobel, and Zerocross. Figure D.1 shows the detections on a Voronoi image.



(a) Voronoi test image **(b)** CVM operator **(c)** Canny

(d) Prewitt **(e)** Sobel **(f)** Zerocross

Figure D.1: Example of various edge detectors.

E 2-Dimensional Deep Learning Training Data

The log likelihood ratio of CVM operator does not need to much further processing. It is rescaled to 0 to 1 value, with the zeros of the log likelihood ratio set to a 0.5 with the max value being the extend of the scale. This is done to prevent any error detections caused by the differences in the background level. This procedure can be seen in the following equation, where LLR represents the log likelihood ratio.

$$\text{rescale the } LLR \text{ to} \begin{cases} [0.5 \quad 1] & \text{for } LLR \text{ with values of } \quad [0 \quad max(|LLR|)] \\ [0 \quad 0.5) & \text{for } LLR \text{ with values of } \quad [-max(|LLR|) \quad 0) \end{cases} \tag{E.1}$$

For the multi-response input network, each of the results of the convolution kernels result are also scaled using the same equation. They are then concatenated on the third dimension with the test image and the log likelihood ratio. Below is the list of the concatenation.

1. Test Image
2. Convolution kernel 1 result
3. Convolution kernel 2 result
4. Convolution kernel 3 result
5. Convolution kernel 4 result
6. Convolution kernel 5 result
7. Convolution kernel 6 result
8. Convolution kernel 7 result
9. Log likelihood ratio

Finally, for the experiment of using CVM operator kernel as the convolution layer weights, the kernel is then saved as a struct that can be loaded as the layer weights. The bias of the weights need to be set to 0. This struct will then be used as an initial weights of the first convolution layer of the UNet.

Bibliography

Albawi, S., T. A. Mohammed and S. Al-Zawi (2018), Understanding of a convolutional neural network, in *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, volume 2018-January, Institute of Electrical and Electronics Engineers Inc., pp. 1–6, ISBN 9781538619490, doi:10.1109/ICEngTechnol.2017.8308186.

Alcantarilla, P. F., A. Bartoli and A. J. Davison (2012), KAZE features, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7577 LNCS, Springer, Berlin, Heidelberg, pp. 214–227, ISBN 9783642337826, ISSN 03029743, doi:10.1007/978-3-642-33783-3_16. https://link-springer-com.ezproxy2.utwente.nl/chapter/10.1007/978-3-642-33783-3_16

Andrade-Loarca, H., G. Kutyniok and O. Öktem (2020), Shearlets as feature extractor for semantic edge detection: the model-based and data-driven realm, doi:10.1098/rspa.2019.0841.

Bay, H., T. Tuytelaars and L. Van Gool (2006), SURF: Speeded up robust features, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3951 LNCS, pp. 404–417, ISBN 3540338322, ISSN 03029743, doi:10.1007/11744023_32.

Bekele, D., M. Teutsch and T. Schuchert (2013), Evaluation of binary keypoint descriptors, in *2013 IEEE International Conference on Image Processing, ICIP 2013 - Proceedings*, pp. 3652–3656, ISBN 9781479923410, doi:10.1109/ICIP.2013.6738753.

Canny, J. (1986), A Computational Approach to Edge Detection, **vol. PAMI-8**, no.6, pp. 679–698, ISSN 01628828, doi:10.1109/TPAMI.1986.4767851.

Clough, J. R., I. Oksuz, N. Byrne, J. A. Schnabel and A. P. King (2019), Explicit Topological Priors for Deep-Learning Based Image Segmentation Using Persistent Homology, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11492 LNCS, Springer Verlag, pp. 16–28, ISBN 9783030203504, ISSN 16113349, doi:10.1007/978-3-030-20351-1_2. https://doi.org/10.1007/978-3-030-20351-1_2

Harris, C. and M. Stephens (1988), A COMBINED CORNER AND EDGE DETECTOR, doi:10.5244/C.2.23.

Hassaballah, M., A. A. Abdelmgeid and H. A. Alshazly (2016), Image features detection, description and matching, *Studies in Computational Intelligence*, **vol. 630**, pp. 11–45, ISSN 1860949X, doi:10.1007/978-3-319-28854-3_2. https://link-springer-com.ezproxy2.utwente.nl/chapter/10.1007/978-3-319-28854-3_2

van der Heijden, F. (1992), *A Statistical Approach to Edge and Line Detection in Digital Images*, Ph.D. thesis, University of Twente.

Hochreiter, S. and J. Schmidhuber (1997), Long Short Term Memory, *Neural Computation*, **vol. 9**, pp. 1735–1780. http://bioinf.jku.at/publications/older/2604.pdf

Jung, D. and C. Sundström (2017), A Combined Data-Driven and Model-Based Residual Selection Algorithm for Fault Detection and Isolation, , no.99, p. 1, doi:10.1109/TCST.2017.2773514. http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-14958215.https://doi.org/10.1109/TCST.2017.2773514http://www.ieee.org/index.html

Kanopoulos, N., N. Vasanthavada and R. L. Baker (1988), Design of an Image Edge Detection Filter Using the Sobel Operator, **vol. 23**, no.2, pp. 358–367, ISSN 1558173X, doi:10.1109/4.996.

Lagarias, J. C., J. A. Reeds, M. H. Wright and P. E. Wright (1998), Convergence properties of the Nelder-Mead simplex method in low dimensions, **vol. 9**, no.1, pp. 112–147, ISSN 10526234, doi:10.1137/S1052623496303470.

Lecun, Y., Y. Bengio and G. Hinton (2015), Deep learning, doi:10.1038/nature14539. https://www.nature.com/articles/nature14539

Lee, M. C. H., K. Petersen, N. Pawlowski, B. Glocker and M. Schaap (2019), TeTrIS: Template Transformer Networks for Image Segmentation with Shape Priors, **vol. 38**, no.11, pp. 2596–2606, ISSN 1558254X, doi:10.1109/TMI.2019.2905990.

Leutenegger, S., M. Chli and R. Y. Siegwart (2011), BRISK: Binary Robust Invariant Scalable Keypoints, Technical report. http://www.asl.ethz.ch/people/lestefan/personal/

Long, J., E. Shelhamer and T. Darrell (2014), Fully Convolutional Networks for Semantic Segmentation, **vol. 39**, no.4, pp. 640–651. http://arxiv.org/abs/1411.4038

Lowe, D. G. (1999), Object Recognition from Local Scale-Invariant Features, Technical report.

Oktay, O., E. Ferrante, K. Kamnitsas, M. Heinrich, W. Bai, J. Caballero, S. A. Cook, A. De Marvao, T. Dawes, D. P. O'Regan, B. Kainz, B. Glocker and D. Rueckert (2018), Anatomically Constrained Neural Networks (ACNNs): Application to Cardiac Image Enhancement and Segmentation, **vol. 37**, no.2, pp. 384–395, ISSN 1558254X, doi:10.1109/TMI.2017.2743464.

Papoulis, A., U. Pillai, B. Burr Ridge, I. Dubuque, I. Madison, Y. San Francisco St Louis Bangkok Bogota Caracas Kuala Lumpur Lisbon London Madrid Mexico City Mila and M. New Delhi Santiago Seoul Singapore Sydney Taipei Toronto (2002), PROBABILITY, RANDOM VARIABLES, AND STOCHASTIC PROCESSES FOURTH EDITION, Technical report. www.mhhe.com

Ronneberger, O., P. Fischer and T. Brox (2015), U-Net: Convolutional Networks for Biomedical Image Segmentation, Technical report. http://lmb.informatik.uni-freiburg.de/

Rueckert, D. and J. A. Schnabel (2019), Model-Based and Data-Driven Strategies in Medical Image Computing, Technical report.

Simonyan, K. and A. Zisserman (2015), VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, Technical report. http://www.robots.ox.ac.uk/

ter Haar Romeny, B. M. (2003), *Front-End Vision and Multi-Scale Image Analysis*, volume 27 of *Computational Imaging and Vision*, Springer Netherlands, Dordrecht, ISBN 978-1-4020-1503-8, doi:10.1007/978-1-4020-8840-7. http://link.springer.com/10.1007/978-1-4020-8840-7

Trajković, M. and M. Hedley (1998), Fast corner detection, **vol. 16**, no.2, pp. 75–87, ISSN 02628856, doi:10.1016/s0262-8856(97)00056-5.

Uzunova, H., M. Wilms, H. Handels and J. Ehrhardt (2017), Training CNNs for image registration from few samples with model-based data augmentation, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10433 LNCS, Springer Verlag, pp. 223–231, ISBN 9783319661810, ISSN 16113349, doi:10.1007/978-3-319-66182-7_26. https://link-springer-com.ezproxy2.utwente.nl/chapter/10.1007/978-3-319-66182-7_26

van der Heijden, F., W. Apperloo and L. J. Spreeuwers (1997), Numerical optimisation in spot detector design, **vol. 18**, no.11-13, pp. 1091–1097, ISSN 01678655, doi:10.1016/S0167-8655(97)00086-X.

Yang, W., R. T. Tan, S. Wang, Y. Fang and J. Liu (2019), Single Image Deraining: From Model-Based to Data-Driven and Beyond, Technical report.