Edge Detection and Enhancement using Shunting Inhibitory Cellular Neural Networks

by

Carmine Pontecorvo, B.E. (Hons. I)

Thesis submitted for the degree of

Doctor of Philosophy

Department of Electrical and Electronic Engineering

Faculty of Engineering

The University of Adelaide

April 1998

Abstract

This thesis describes the application of shunting inhibitory cellular neural networks to the problems of edge detection and enhancement. Edge detection is needed in many mammalian and computer vision systems to reduce the vast amounts of incoming data to a relatively small number of features. Shunting inhibition describes the nonlinear interactions between sensory cells, and has been used to explain many nonlinear visual phenomenon as found in the mammalian visual system. A cellular neural network (CNN) is a grid of locally connected, nonlinear parallel processing elements, able to process information in both space and time. CNN with shunting inhibition as the nonlinear interaction are referred to as SICNN.

This thesis begins with a general investigation of recurrent and feedforward SICNN systems. By linearising the feedforward SICNN using perturbation analysis, the frequency and impulse response characteristics are derived and its response to random (noisy) inputs is investigated. We also discuss how the SICNN can be designed to perform edge detection, and how the factors of the output and the SICNN parameters affect the performance.

Following this, we investigate how the SICNN parameters can be chosen to maximise edge detection performance given some optimality criteria. The SICNN weight distribution affects the edge detection performance, in particular the edge standard deviation and the hit rate. Hence an optimal weight distribution is derived using constrained optimisation to simultaneously optimise a number of criteria, namely the statistical measures of hit rate and edge standard deviation.

We also derive the SICNN decay factor which gives zero edge bias in the 1-D edge detection performance, which also optimises the hit rate and edge standard deviation. For input edges with multiplicative noise, this optimal decay factor is zero, but for additive noise, it is proportional to the noise standard deviation and the sum of weights. The constant of proportionality is empirically derived and tabulated for various weight distributions for both 1-D and 2-D synthetic edges. The SICNN performance with a number of different nonlinear activation functions is also investigated.

Next we investigate a number of postprocessing methods for the SICNN output. Most of

the methods involve the combination of the outputs of different SICNNs to improve the edge detection performance. The edges in the SICNN output are tracked as the neighbourhood size and the width of the weight distribution are slowly reduced, resulting in an improvement in the 1-D and 2-D edge detection performance.

The outputs of SICNNs with different weights and thresholding schemes are combined to give significant improvement in the performance, particularly for 1-D edges. The outputs of SICNNs with reversed weight distributions are also combined to improve the performance, especially for edges with small contrast. Finally, by examining the number of edges in the local neighbourhood of each edge pixel, spurious edges which may arise from noise can be eliminated resulting in improved performance.

Having designed and optimised the SICNN for edge detection we compare it to a number of standard edge detectors on both synthetic 1-D and 2-D images, as well as real 2-D images. The results indicate that the SICNN has better performance than the linear operators, particularly for inputs with multiplicative noise and on real images with both multiplicative and additive noise.

Finally, we look at the edge enhancement capabilities and properties of the SICNN. We investigate a number of different enhancement measures, and propose a new measure for step edges, called the Edge Enhancement Product (EEP). The EEP measures not only the edge enhancement, but also the difference in the background intensity levels of the step edge. When solving for the SICNN steady-state, the solution can be found using an iterative approach using a recursive sequence. The SICNN decay factor that maximises the EEP after 1 iteration of the sequence is derived and was shown to be the largest for different number of iterations. Thus, the output after 1 iteration is always used for image enhancement. The EEP is also used to find optimal parameters for other edge enhancers when the value of their parameters is not clearly defined. Finally, we compare the enhancement of both synthetic and real images using the SICNN and a number of standard edge enhancement techniques, and show that the SICNN is in general superior to the others, particularly for inputs with multiplicative noise.

Statement of Originality

I hereby declare that this work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and that, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. I also give consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying.

Carmine Pontecorvo

Signed:

Dated: 22 April 1998

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Abdesselam Bouzerdoum. Without his guidance, enthusiasm, and support this thesis could never have been completed. His seemingly endless energy and optimism helped me to overcome many of the difficulties that I encountered over the years. His cheerful personality and funny stories will be missed. All in all, I couldn't have asked for a better supervisor and friend.

I would also like to thank Prof. Robert Bogner for his probing and incisive (and often perplexing!) questions and comments over the years.

I gratefully acknowledge the financial support provided by CSSIP, the Australian Research Council (ARC), and the Australian Government through the APA scholarship.

To the people of the Signal Processing Lab., Jamie Sherrah, Habib Hosseini, Steve Wawryk, Anna Buzescu, Zhishun She, Ben Raymond, and Jihan Zhu, I thank them for all the fun and happy times, and of course the good coffee.

For computing matters, I am grateful to Jihan Zhu for tirelessly setting up my computing environment, particularly in the early stages of this thesis. Very special thanks also to Ninh Duong for his patient help and explanations with Framemaker during the write-up of my thesis. It was also great to have made such a good friend during my postgraduate years.

My friends Jijoong Kim and Christine Jang gave so much to me and initiated me on my path to my future. So many wonderful things have happened to me that can always be eventually traced back to our friendship.

To my family, and parents in particular, I thank them for their neverending support during all my years of study. They provided me with everything that I needed, and showed me what hardwork and unselfish love are.

Finally, I give praise and thanks to the Lord, Jesus Christ. I thank him for giving me everything that is dear to my heart, and for teaching me about love and humbleness. I know that where ever I will go in the future he will always be walking together with me.

Publications

Pontecorvo, C., & Bouzerdoum, A. (1995). "Edge Detection using a Shunting Inhibitory Cellular Neural Network." *Digital Image Computing: Techniques and Applications*, pp. 637-642, Brisbane, Australia, 6-8 December.

Pontecorvo, C., & Bouzerdoum, A. (1997). "Edge Detection in Multiplicative Noise using the Shunting Inhibitory Cellular Neural Network." *Proceeding of the 1997 International Conference on Engineering Applications of Neural Networks*, pp. 281-285, Stockholm, Sweden, 16-18 June.

Abbreviations

pdf	Probability Density Function
CNN	Cellular Neural Network
DoG	Difference-of-Gaussian
EB	Edge Bias
EEP	Edge Enhancement Product
ENR	Edge-to-Noise Ratio
ESD	Edge Standard Deviation
FA	False Alarm rate
FOM	Figure of Merit
GLE	Gradient-Like Enhancement
HR	Hit Rate
LoG	Laplacian-of-Gaussian
MOP	Measure of Performance
NF	Noise Figure
NSR	Noise-to-Signal Ratio
PD	Probability of Detection
PNR	Peak response to Noise Ratio
PSF	Point Spread Function
REE	Relative Edge Enhancement
RV	Random Variable
SAR	Synthetic Aperture Radar
SICNN	Shunting Inhibitory Cellular Neural Network
SNR	Signal-to-Noise Ratio
SSW	Sum of Squared Weights
1-D	1-Dimension
2-D	2-Dimensions
MAW	Maximum output of the SICNN with an asymmetrical weight distribution
MSW	Maximum output of the SICNN with a symmetrical weight distribution
ZCSW	Maximum zero-crossing in the output of the

	SICNN with a symmetrical weight distribution
UM	Unsharp Masking
n-MAW	The n largest maxima in the output of the SICNN with asymmetrical weights
n-MSW	The n largest maxima in the output of the SICNN with symmetrical weights
n-ZCSW	The n largest zero-crossings in the output of the SICNN with symmetrical weights
a_{ij}	Decay factor of excitation of ij^{th} SICNN cell
c_{ij}	Contrast at pixel ij
$C(i,j)$, C_{ij}	Cell ij of a Cellular Neural Network
f	SICNN's activation function
$h(n)$	Filter impulse response
$H(\omega)$	Fourier transform of the filter impulse response
H_j	Hypothesis that event "j" has occurred
I_0	Mean input intensity
I_{ij}	Input intensity at pixel ij
p	Weight attenuation factor
r	SICNN neighbourhood size
w_{ij}^{kl}	Weight on the output of cell kl to cell ij
W	Sum of SICNN weights, $\sum_{j \in N_r} w_j$
x_{ij}	State value of ij^{th} cell of the SICNN
X_0	Mean output intensity of the SICNN
y_{ij}	Output intensity of ij^{th} cell of the SICNN
Δy	The SICNN's peak response
$\delta(n)$	Dirac delta function
μ	Background intensity of the SICNN output
ν_{ij}	Input noise signal at ij^{th} cell of the SICNN
η	Threshold multiplier
τ	Output intensity threshold

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Vision and Inhibition

For many years now, both insect and mammalian visual systems have been a constant source of inspiration for computer vision researchers. The ability of these visual systems to successfully operate under a wide range of operating conditions has drawn both envy and admiration from neurobiologists, computer scientists, and engineers alike.

Taking edge detection as an example, both the mammalian and insect visual systems are able to represent the vast amounts of incoming data in terms of edges, as well as other primitives (Marr & Hildreth, 1980). The neurons in the visual system are able to interact with each other in a complex or nonlinear way, known as inhibition. Linear inhibition has been shown to remove redundancy in the input, increase sensitivity, improve efficiency and resolution, and also enhance or deblur edges (Srinivasan et al., 1982). It can also explain many visual phenomena, such as Mach bands. However, for some visual phenomena, nonlinear or shunting inhibition is needed whereby each neuron in the visual system interacts in a nonlinear way with its neighbours.

In this section we begin by describing the basic structure of the mammalian eye, followed by a description of linear and nonlinear inhibition. We discuss its usefulness to the biological visual system, followed by a brief biological and electrical description of shunting or multiplicative inhibition. Finally, we discuss the role of inhibition in the early visual system.

1.1.1 The Structure of the Eye

Figure 1.1 shows a crude schematical representation of a cross-sectional slice of the human eye. The eye is nearly spherical in shape, and has a diameter of the order of 20mm.

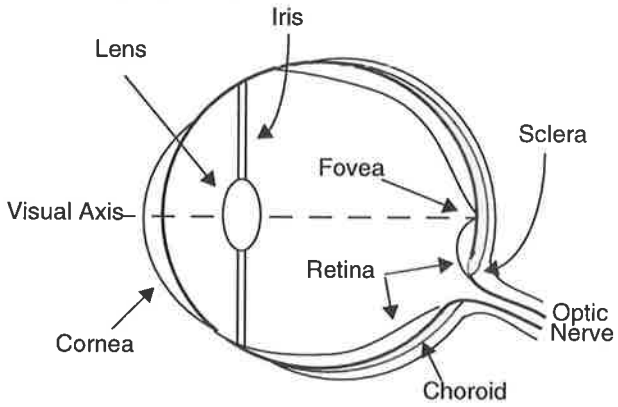


Figure 1.1 Simple cross-section of human eye.

The eye is enclosed by three membranes, the *choroid*, *sclera*, and *retina*. The sclera is an opaque membrane which covers part of the optic globe. The choroid lies directly beneath the sclera, and its main function is to give nutrition to the eye and to reduce the amount of back-scattered light within the eye by being heavily pigmented. The *iris* has a central opening, the *pupil*, which can be controlled to vary the amount of light entering the eye. For instance the iris contracts under conditions of dim-light to enable more light to enter the eye aiding visibility. Behind the pupil (and iris) is the *lens* consisting of layers of fibrous cells containing mostly water, suspended from *fibres*. Its role is to focus light onto the back part of the eye.

The innermost membrane is the *retina*, and when the optical system is properly focused the external object is imaged onto it. To detect the distribution, and intensities, of the incoming light the surface of the retina is covered with *discrete light receptors* called the *cones* and *rods*.

Cones are responsible for colour vision, and they number in the order of 6 to 7 million. Most cones are located in a portion of the retina called the *fovea*. As a result of this the eyeball usually orientates itself so that the object of interest falls onto the fovea. The resolution is very good since each cone is connected to only one nerve ending. Cone vision is usually called *photopic* or bright-light vision.

Rods, on the other hand, number in the order of 75 to 150 million. They are spread throughout the retina, with many rods connected to a single nerve ending, thus reducing their resolution. They give an overall picture of the field of view, and are sensitive to low-levels of illumination. This phenomena is known as *scotopic* or dim-light vision.

Electrical impulses from all the nerve endings exit the eyeball through the *optic nerve*. There are no receptors in the location of this exit, hence vision is poor when this exit is aligned with the visual axis. This region is called the *blind-spot* and occurs about 20 degrees away from the fovea.

1.1.2 Linear and Nonlinear Lateral Inhibition

Lateral inhibition is frequently encountered in the preliminary stages of sensory processes such as touch and vision (Deutsch & Deutsch, 1992). It is a concept that explains interactions and information processing between neighbouring sensory nerve cells. Linear lateral inhibition was first proposed by Ernst Mach to explain the border contrast effects commonly referred to as Mach bands (Mach, 1886a; Mach, 1886b). Figure 1.2 shows an example of Mach bands at the transitions between regions of different intensities. Mach bands are the illusory dark and light bands on either side of each transition. This phenomena is essentially due to inhibition causing the perceived brightness to differ from the actual luminance. Mach proposed a receptive field similar to that shown in Figure 1.3 which causes the brightness to undershoot and overshoot at the transition borders.

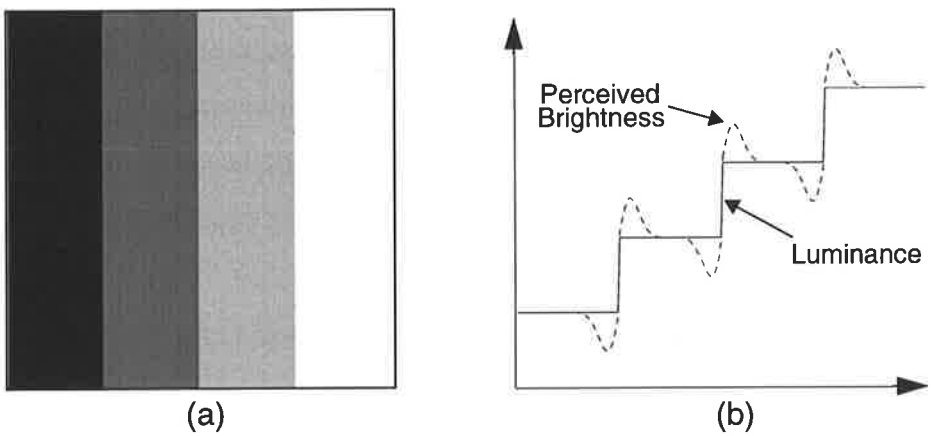


Figure 1.2 (a) An image consisting of regions of increasing intensity. Mach bands can be seen to the left and to the right of each transition. (b) Shows the corresponding plot of luminance and perceived brightness.

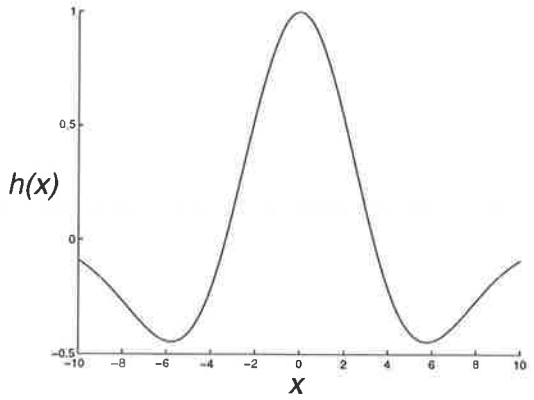


Figure 1.3 An impulse response depicting lateral inhibition.

The first extensive studies on inhibition, however, were performed by Hartline and Ratliff on the compound eye of the horseshoe crab, *Limulus* (Hartline & Ratliff, 1957; Hartline & Ratliff, 1958; Ratliff et al., 1963). The function, or role, of linear lateral inhibition has been concisely outlined by Srinivasan et al. (1982). They include:

- redundancy removal to improve the efficiency on the supply of information through the optic nerve (Barlow, 1981).
- removal of the DC bias in the input - to increase sensitivity (Brodie et al., 1978).
- deblurring and edge enhancement - e.g. the Mach bands phenomena.
- predictive coding to improve efficiency and resolution (Srinivasan et al., 1982).

For many neural cells and visual phenomena, linear lateral inhibition is an insufficient model. A nonlinear model is more appropriate in these cases. When the membrane conductance is controlled by the synaptic voltage of neighbouring cells, the mathematical equation describing the lateral inhibitory neural network becomes nonlinear, or more commonly referred to as *multiplicative*. The system of state equations is

$$\frac{dx_i}{dt} = I_i(t) - a_i x_i - \sum_{j=-n}^{n} w_{ij} f_{ij}(x_j)\, x_i \qquad i = 1, 2, \ldots, n$$

where x_i is the state of the i^{th} neuron, I_i is the input to the i^{th} neuron, a_i is the decay factor of the i^{th} neuron activity, w_{ij} is the coupling strength of postsynaptic activity of the j^{th}

neuron to the i^{th} neuron, and f_{ij} is the activation function which determines the inhibitory action exhibited on the i^{th} neuron by the j^{th} neuron.

Nonlinear, or multiplicative lateral inhibition has been used to explain selectivity of visual units in the ventral nerve of insects for small objects (Pinter, 1983a; 1983b), and to explain adaptation of the receptive field spatial organisation and the spatial modulation transfer function (Pinter, 1984; 1985). Nonlinear lateral inhibitory neural networks have also found application in image processing, mainly for image enhancement (Jernigan & McLean, 1992; Paradis & Jernigan 1994).

1.1.2.1 Biological and Electrical Description of Shunting Inhibition

The *neuron* is the fundamental processing unit in the human nervous system. Each neuron is connected to many thousands of other neurons through nerve fibres called *dendrites*. The dendrites "connect" to the *cell body*, or soma, via *synapses*. The neuron itself communicates with other neurons through the *axon*, which is connected in turn to many dendrites. Figure 1.4 shows a simple diagram of a neuron to illustrate qualitatively the principles involved in shunting inhibition.

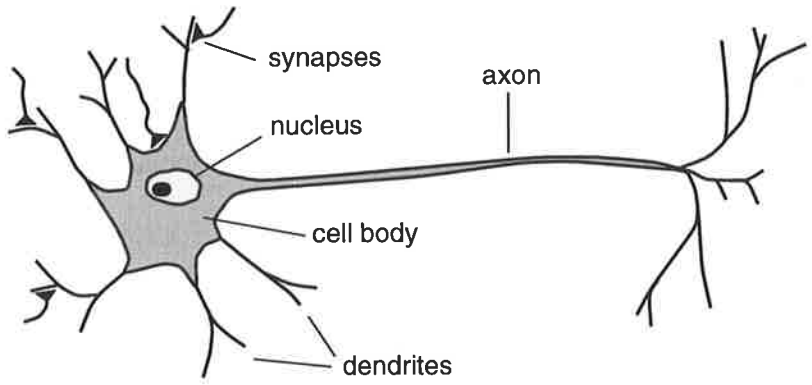


Figure 1.4 Schematic drawing of a typical neuron (from Hertz et al., 1991).

An equivalent electrical circuit for a cell is shown in Figure 1.5 (Bouzerdoum & Pinter, 1993). Note that we shall call a path from the synapses to the soma a *channel*. Such a circuit is an approximation to a uniform lump of membrane, where each cell may correspond to a subunit, i.e., a piece of dendritic field that is isolated from other such fields, and on the whole has a uniform potential within. When this is the case the resistance between synapses is negligibly small, and the entire excitatory and inhibitory

synapses can be lumped together into a common circuit element. We denote by V_m and C_m the *membrane voltage* and *capacitance*, respectively.

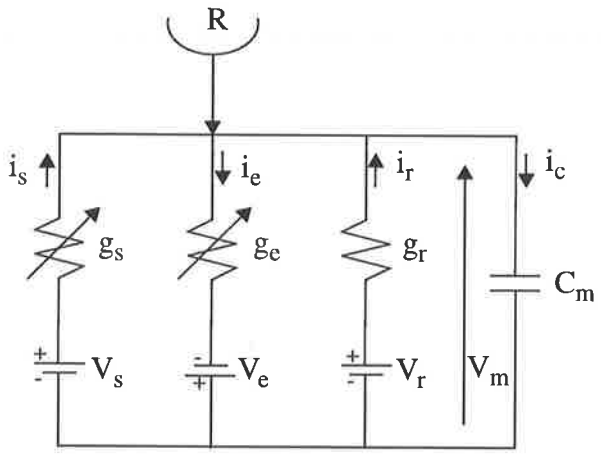


Figure 1.5 An equivalent electrical representation of a biological cell, or neuron.

When the neuron is not excited, or at rest, the corresponding resting conductances and batteries of all possible channels can be lumped together into the resting conductance g_r and potential V_r. Thus the conductances of all excitatory and inhibitory channels are zero. Now exciting the excitatory synapses increases the conductance of the corresponding ionic channels, or the membrane conductance, causing sodium Na^+ ions to enter the soma. This flow of ions is represented by a modulation of the conductance g_e, with the polarity of the potential V_e reflecting the direction of ionic flow. In actual fact V_e is a measure of the flow of ionic charge across the channel.

The inhibitory synapses are assumed to be of the *shunting type* since this branch of the circuit consisting of g_s and V_s shunts the rest of the circuit. When the inhibitory synapses are excited, more chloride Cl^- ions are able to enter the soma through their ionic channel. This also increases the outer membrane potential with respect to that on the inner membrane. The voltage V_s represents the flow of Chloride ions across the channel. The changing Cl^- ionic channels can be represented electrically as a modulation of the conductance g_s. Another ionic channel also exists for potassium K^+ ions to exert inhibition by leaving the soma and increasing the outer membrane potential, but the Cl^- ions are the dominant one in causing shunting inhibition.

Note that there is always a constant diffusion of charge across the membrane due to the inherent potential difference across it. At rest (or steady-state) the amount of charge diffusing into the soma is equal to the charge flowing out of the cell body, represented by the inhibitory and excitatory potentials.

Thus when the cell is excited, the excitatory conductance g_e becomes larger, indicating an excitatory effect, and g_s increases to divert current from i_r to i_s. This diversion causes the circuit to *clamp* the cell to its resting potential, in other words, it attempts to maintain a constant equilibrium voltage $V_s = V_r$. This is what we shall refer to as *inhibition*.

1.1.3 Role of Inhibition in Early Visual Processing

The peripheral visual system of mammals has a hierarchical organisation - from the retina to the lateral geniculate nucleus (LGN) and to the visual cortex. Information from the retina is relayed via the LGN to the striate cortex; there also exists massive feedback throughout the whole system, such as from the visual striate cortex to the cortex.

There exist two distinct parallel pathways from the retina to the striate cortex: the magnocellular pathway for motion information and the parvocellular pathway for contrast and shape information. Each pathway consists of ON and OFF-centre receptive fields. The ON-centre cell increases its activity, or firing, when the light intensity over the centre of its receptive field increases, while the OFF-centre cell increases its activity when the light intensity decreases over the centre of its receptive field.

Retinal ganglion cells and LGN cells both have receptive field profiles with Centre-Surround (CS) organisation, i.e., a small excitatory band surrounded by an annular inhibitory band. Cortical cells (or simple cells) have receptive fields that can be subdivided into alternating excitatory and inhibitory regions resembling the ON and OFF subregions of a CS field. They are selective to orientation stimulus - thus termed Orientation Selective (OS); see Spillman & Werner (1990). A hierarchical model was presented by Bouzerdoum (1994) that can synthesise both the CS receptive field of retinal and ganglion cells, and the OS receptive field of cortical cells, for both ON and OFF-channels in the parvocellular system using shunting inhibition.

1.2 Shunting Inhibitory Cellular Neural Networks

Lateral inhibition describes the complex mechanism by which sensory cells interact with each other, and was first proposed by Ernst Mach (1886a, 1886b) to describe the edge effects observed at the discontinuity between two different intensity bands. This phenomenon is now referred to as Mach bands. Since the pioneering work of Mach, inhibition has been shown to have an important part in the early visual processing system.

Multiplicative or shunting inhibition describes the case where the interaction between neighbouring cells is of a multiplicative nature; thus, it is inherently nonlinear. Pinter (1983a, 1983b) used lateral inhibition to explain the selectivity of visual units in the ventral nerve of insects for small objects, and also to explain the adaptation of the receptive spatial organisation and the spatial modulation transfer function (Pinter 1984, 1985). Shunting inhibition has also found applications in image enhancement (Jernigan & McLean, 1992; Paradis & Jernigan, 1984), as well as in motion detection (Bouzerdoum, 1991).

Cellular neural networks (CNNs) were presented by Chua & Yang (1988) as a framework for analogue, nonlinear processing arrays. A CNN consists of a nonlinear processing node in a grid layout, with each cell being locally connected to its neighbouring cells. Many possible CNNs have been described, and they have found many applications in image processing, as CNNs have an excellent ability to process information locally, in both time and space.

The architecture and nonlinear processing ability of the CNN makes it ideal for modelling nonlinear inhibition in the mammalian system which typically consists of neurons in a grid-like structure, with many local interconnections. Bouzerdoum & Pinter (1993) & Bouzerdoum (1994) were able to adapt and design a CNN to model shunting inhibition. Using a hierarchical model of such CNNs, Bouzerdoum (1994) was able to synthesize both the centre-surround receptive field of retinal and ganglion cells, and the orientation selective receptive of cortical cells, in the ON- and OFF- channels of the parvocellular system. Iannella & Bouzerdoum (1996) used a hierarchical network of shunting inhibitory cellular neural network (SICNN) to synthesize the spatiotemporal receptive fields of the early mammalian visual system.

1.3 SICNNs for Edge Detection and Enhancement

Edge detection is the detection of significant intensity changes corresponding to physically underlying edges in a visual scene. It is a low-level process as it operates directly on the incoming data and makes no assumptions about it.

Any vision system, either mammalian or otherwise, must deal with enormous amounts of incoming visual data. For example, a 512×512 greyscale image quantised using 8 bits consumes over 250 kBytes. Thus, it is beneficial if the system can avoid processing such huge amounts of data. One way of doing this is to identify salient features in the input, and then process only these features, rather than the entire input image. There is some evidence that the primitives which the mammalian visual system identifies are edges (Marr & Hildreth, 1980). Edge detection is an important part of many image processing algorithms, such as image segmentation, as well as many computer vision applications as described in the following Chapter on edge detection.

Traditionally, most edge detectors described in the literature have been linear operators, with no ability to adapt to the characteristics of the input signal. Indeed, there is no real need to adapt to the input if we assume that the input image is corrupted with additive white noise. Although this is a reasonable assumption, in many cases it does not hold. For example, it is well known that images produced from coherent imaging systems, such as radar, have multiplicative noise present, which is not additive in nature. That is, the noise intensity depends upon the signal's intensity. Clearly, a linear edge detector is not optimal for inputs with multiplicative noise. Furthermore, most edge detectors only have a few parameters to adjust for achieving the best possible performance.

In this thesis, we will take the shunting inhibitory CNN (SICNN) developed by Bouzerdoum & Pinter (1993) and adapt and redesign it for the problem of edge detection. We will show that the SICNN edge detector has the ability to adapt its impulse response to the input signal's intensity, giving it an inherent advantage over linear edge detectors for signals corrupted with multiplicative noise. Furthermore, the SICNN developed here has many parameters which can be tuned and adjusted to obtain the best performance for any given input image.

1.4 Thesis Overview

In Chapter 2, we begin this thesis with an overview of edge detection theory and applications. We present a general review of the more popular edge detectors as found in the literature, a number of performance measures, and the applications of edge detection. We also describe how edges are formed from various types of surfaces.

Chapter 3 presents a general overview of cellular neural network (CNN) theory, and discusses a number of different implementations as found in the literature. Importantly, we present the shunting inhibitory cellular neural network.

Chapter 4 presents a general investigation into shunting inhibitory CNN (SICNN) systems, both recurrent and feedforward. We describe how the recurrent SICNN can be solved for the steady-state, and what this steady-state is, for both uniform and step edge inputs. We derive the SICNN impulse and frequency responses, and then present a detailed investigation into the SICNN response to random inputs.

In Chapter 5 we discuss how the SICNN is designed and adapted for edge detection. We begin by defining the edge detection performance measures which are used throughout this thesis. We then outline a number of issues related to the design and implementation of the SICNN. The factors affecting the performance are investigated, as well as the effect of each SICNN parameter on the performance.

Chapter 6 presents a study into how various SICNN parameters are optimised, according to certain criteria, to maximise the SICNN edge detection performance. In Chapter 7, we present a number of post-processing methods for the SICNN that can be used to improve its performance. Particular emphasis is placed on how various outputs of the SICNN can be constructively combined to achieve an improvement in performance.

In Chapter 8, we present the culmination of the previous Chapters work, by comparing the SICNN edge detection performance with that of a number of other standard edge detectors. The results are presented for synthetic 1-D and 2-D edges, as well as for 2-D real images.

In Chapter 9, we apply the SICNN to edge enhancement. We begin by developing a new edge enhancement measure, and then review a number of standard edge enhancement schemes as found in the literature. We describe the various ways in which the SICNN can

be used for edge enhancement, and finally we compare its enhancement abilities to those of standard edge enhancement operators.

Finally, we present the conclusions of this thesis and a summary of its major contributions, and also describe a number of recommendations for future work.

Chapter 2 Edge Detection: a Review

2.1 Introduction

Before reviewing a number of edge detectors proposed in the literature, we explain the generation of image intensity profiles, and show how image intensity understanding can provide us with the fundamental knowledge of the creation of edge intensity profiles. It is important to know which profiles are most frequently generated by physical edges, rather than attempting to detect all edge profiles - some of which may be of no real use to higher level computer vision tasks. Thus, we begin with the generation of typical edge profiles; that is, those that we should concentrate on detecting. It is also more satisfying if we can appreciate which physical edges produce what profiles.

Edge detection is a critical component of many computer vision systems. It reduces the vast amounts of visual data presented to a vision system by locating and highlighting only the salient edges. These edges can be used by the higher-level processes of the system to make decisions about navigation, obstacle avoidance, etc. In this Chapter, we discuss a number of the more important edge detectors found in the literature, and how they operate. Common performance measures which can be used to rank or compare different edge detection schemes are also reviewed.

2.2 The Generation of Edges

In this section we begin with the fundamental definitions of light, luminance, and brightness, along with some background of how intensity profiles, or edges, are generated,

i.e., what physical underlying scenery generates the most common edge profiles which we should detect.

2.2.1 Light, Luminance & Brightness

Light is an electromagnetic radiation that stimulates the visual system. Its wavelength, λ, is of the order of 350nm to 780nm. The primate visual system does not respond equally to light of all frequencies. It is tuned to the range of wavelengths (frequencies) stated above. This characteristic of the visual system can be described by the function $V(\lambda)$, the *relative luminous efficiency function*, which usually has a bell-shaped curve (Gonzalez & Woods, 1992). The eye appears to be most sensitive to light with wavelengths centred around 600nm.

The *luminance* or *intensity* of an object is measured in *lumens* (lm) and is a measure of the light energy perceived by an observer. For example, the luminance of an infra-red source would be almost zero to a human observer, even though the source is emitting energy.

The *brightness* (or *apparent brightness*) of an object is defined as the *perceived* luminance, and depends upon the luminance of the *surroundings* of that object. Thus, two objects can have identical luminance but different brightness depending upon their backgrounds. Phenomena such as *simultaneous contrast* and *Mach bands* illustrate this effect.

2.2.2 Edge Generation

In most papers dealing with edge detection, the actual physical source of an edge profile is often overlooked or ignored. In general, this has not restricted the progress of edge detection theory since the most commonly studied edge profiles (i.e., the simplest ones) are indeed those readily found in natural scenes.

We shall now qualitatively discuss a number of luminance profiles, or edges, and the underlying physical scene events, or physical edges, which give rise to them. Although the theory is not crucial to our work, it gives us a greater understanding of edges and hence edge detectors.

Horn (1977) showed, under some simplifying assumptions, that an edge can be described in terms of the underlying physical edges. These assumptions are:

- the reflectance of objects is *Lambertian*, i.e., the brightness from any surface patch is proportional to the incident light on that patch, and is equally bright from all directions.
- The illumination is mainly coherent, i.e., a single light source subtending a small solid angle.

Such assumptions rarely hold in practice but they do yield useful models.

Figure 2.1 shows three of the most common edge profiles according to Horn. They are the **step**, **peak** and **roof** profiles. A study by Herskovitz & Binford (1970) also showed such edges to be very common in real scenes. In general, most edges can be decomposed into a combination of these fundamental edges, so we do not need to consider others.

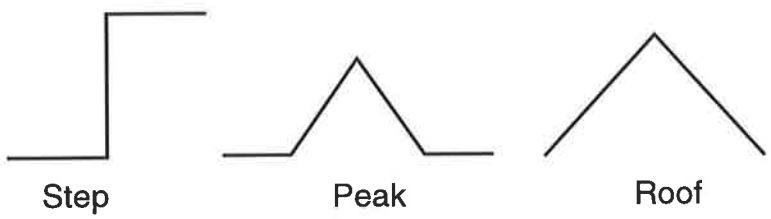


Figure 2.1 Commonly found intensity profiles.

A step profile is generated when two faces of an object occlude each other, as shown in Figure 2.2(a). The border between the faces is very sharp (from the illuminating light's frame of reference), hence the luminance profile of the edge will change abruptly, or like a step function (Horn, 1977).

With convex edges, such as Figure 2.2(b), the luminance profile is constant along surfaces A and B, and then peaks somewhere along the curved edge in between A and B. This usually results in a peak-like edge profile (Horn, 1977).

With an elongated concave edge such as in Figure 2.2(c), the luminance profile decreases gradually along both surfaces, generally resulting in a ramp-like edge profile (Horn, 1977).

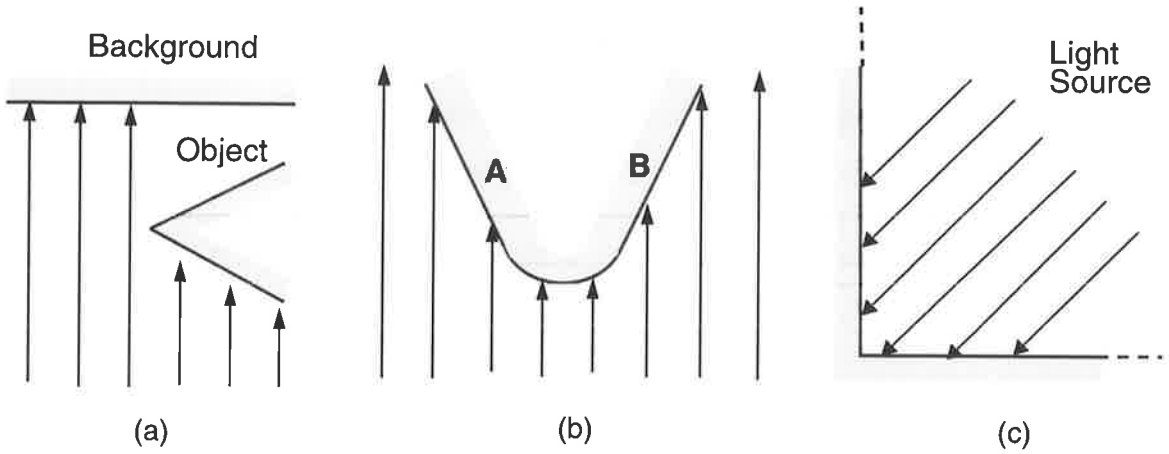


Figure 2.2 Illustration of (a) an obscuring edge, (b) a convex edge, and (c) a concave edge.

2.3 Applications of Edge Detectors

In this section we begin by discussing the need for edge detectors in vision systems. The vast amounts of incoming data need to be represented in terms of a few primitives, which can then be acted upon or interpreted by the higher levels parts of the vision system. Following this we describe a number of industrial applications of edge detection, such as electronic manufacturing and parts inspection. We also describe the use of edge detection for consumer electronics and other computer vision applications.

2.3.1 The Need for Edge Detectors

Computer vision systems are designed to recover useful information of a scene from one or more 2-D images of that scene. Such systems are often considered as consisting of three processing levels: low, intermediate, and high level vision. A common belief is that low-level processes (or early processes), such as edge detection, are data-driven whereas the high-level processes use explicit knowledge. A simple analogy is that low-level vision performs the same tasks as the human eye, whereas high-level processes have a role similar to that of the brain.

It is clear that the success of many computer vision systems is critically dependent upon the accuracy of the low-level processes (which supply the high-level processes with the necessary scene information). Errors at such low-level may propagate through the higher

levels of the system. As a consequence, a great deal of research effort has been applied to developing these early processes, and in particular edge detection.

In simple terms, edge detection is the identification of intensity changes that correspond to physically underlying edges, some of which we have discussed above. Detecting edges is useful for image segmentation and registration, robotics, character recognition, and medical and industrial applications to name a few. Notwithstanding such a simple description, research has yet to produce an edge detector of any real sophistication that can yield satisfactory results, particularly when applied to real images and whose results are viewed objectively by humans. The lack of good results has been a major source of frustration for researchers developing higher level algorithms for computer vision systems (Nalwa & Binford, 1986).

Although not an important aspect of this work, edge detection also reduces the amount of information that must be communicated within any visual or computer vision system. The quantity of incoming data is enormous for any system, and edge detection reduces the amount of data to be transmitted by identifying only the salient edge points. Thus only the location of edges need to be transmitted.

2.3.2 Industrial Applications

The advent of modern mass production techniques and automated production line technology has initiated substitutes for the traditional human inspectors. It is impossible for these inspectors to inspect parts or components for both cosmetic and functional defects, whilst still maintaining 100% quality assurance. For example, in the manufacturing of PCB, a human inspector must monitor the results of over 50 process steps in the fabrication. Not only is this work labour intensive, but it is also subjective since the operator must compare the fabricated part to a "standard" part. Thus, automatic visual inspection systems are used to:

- remove the subjectivity of human operators,
- relieve human inspectors of tedious jobs,
- achieve daily inspection consistency and high quality assurance levels,
- reduce rework cost,

- reduce excessive (and costly) scrap-rates of defective parts,
- achieve production rates beyond the human inspection capability,
- inspect tolerances which are too tight for humans,
- can categorize defect types to control and improve the manufacturing process, and
- in critical applications, it can identify a faulty product before it reaches the customer.

See Moganti et al. (1996), Ejiri (1989), and Novini (1995) for more details. Thus, the net result of replacing a human inspector with an automated visual inspection system is to produce a cheaper product of better quality.

Electronic Manufacturing

Computer vision has become popular for automatic visual inspection of electronic devices and components, as the physical dimensions are always decreasing, it provides a useful auditing facility, with such systems now being cost-effective.

As the physical dimensions of the devices and components decrease, it is becoming increasingly difficult for a human inspector to inspect and measure components quickly and accurately. As we mentioned above, automated inspection can be used to determine how and why faults occur in the manufacturing process. Not only does this improve the process, but it also eventually reduces the number of discarded components, which represents a direct saving of money. Visual inspection is also a solution of PCB inspection due to its ability to process information of a high data rate and resolution, and its ability to inspect PCBs with no contact or damage to the board (West, 1984).

Edge detection has an important role in the metrology of electronic devices and substrates (Pau, 1990). Edges are used to measure linewidths of tracks to determine if they are within a prescribed tolerance limit. The peak local slope of an area of interest, along with other measurements, is also used to determine if the flatness is within a prescribed tolerance. Flatness measurement is essential for semiconductor wafers, multi-layered PCBs, and for surface mounted devices (SMD). For IC inspections, edges are located and used to measure features which are then compared to the reference CAD layout.

By identifying components and their placement, edge detection is useful for component insertion, wafer or board repairs, desoldering, probers of automatic test equipment, PCB drilling, connector mountings, wafer saw alignment, etc. (Pau, 1990).

West (1984) proposed a system for the automatic visual inspection of bare PCBs using a hierarchy of tests for different types of faults. To detect "large faults", such as missing tracks or pads, the inspected board is compared to a "standard" board held in memory. The board is segmented into grids, and edges are then used to measure the area of tracks in each grid. By comparing the areas in each grid to the areas found using the standard board, errors can be located. Edges are also used to measure the track width over a long length of track, and detect where the track width is too narrow.

Kaufmann et al. (1984) describe a method of inspection using a feature based description of the image and model. Edge detection is used to generate line segments which are used to describe all the parts being inspected. These line segments are then used to symbolically describe the part, and then comparing this description to the one of the stored model. Parts which can be detected include PCBs, IC chips, capacitor and battery contacts, while defects include broken PCB, missing IC, missing capacitor, and misplaced or missing battery contacts.

Parts Inspection

Figure 2.3 below shows a typical machine vision system for parts inspections or robot control. The system consists of an imaging system or sensor (camera and optics), a preprocessor for feature enhancement and extraction, and an intelligent processor to either orientate the inspected part or to guide robot's gripper.

Kelley (1992) describes the bin-picking problem as acquiring a part from a bin and then presenting the part, in its correct orientation, to a robot or the following stage of the assembly process. Not only does this remove the limitations of human performance, but it also relieves humans of monotonous jobs, and it can remove them from an environment which may be unhealthy or hazardous. For acquisition, he describes how features, such as edges, are used to provide reliable indicators for good holdsites for a robot's grippers. An example is presented where edges are used for the robot gripping of automobile engine connecting rod castings which are then placed onto a conveyor belt.

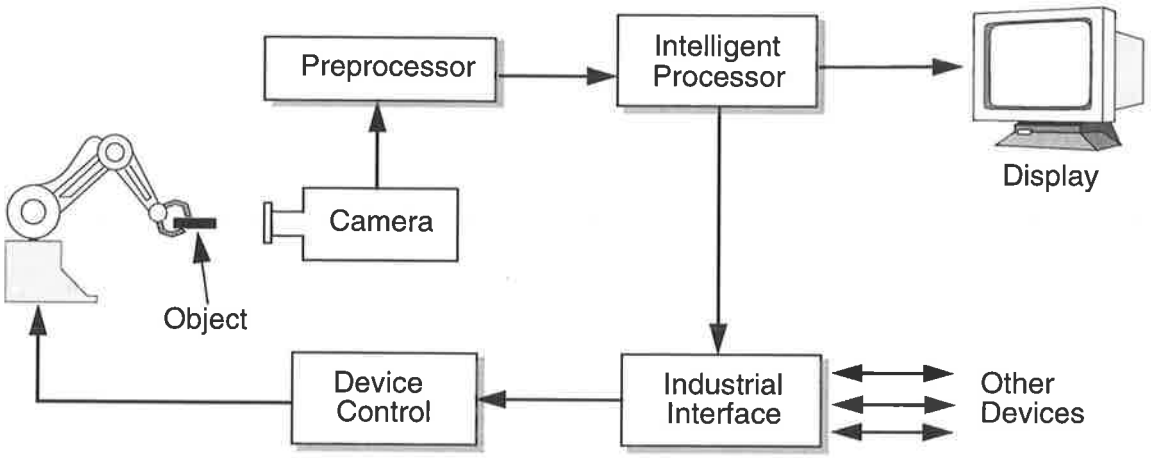


Figure 2.3 A machine vision system for automated inspection or control (Batchelor & Braggin, 1992).

Perkins (1983) describes a very simple system which can be trained to inspect parts, and then inspect new parts automatically. During training, the system forms a model of the part by using edge detection to determine identifying points which characterise the part. The system can then distinguish between good and bad parts by comparing the intensity or edge distribution of the inspected part compared to the training set of parts.

Other Automatic Inspection Systems

Suresh et al. (1983) describe an automatic visual inspection system for the detection of imperfections on hot steel slabs being produced in steel mills. By inspecting the steel slabs while they are still hot, they not only remove the need for time-consuming human inspection, but there can be significant savings in operating costs. Edges are used in their algorithm to identify imperfections on the incoming piece of steel slab, and to determine if these imperfections are localised, or are part of a larger, more serious, imperfection.

Ala et al. (1992) provide a systematic study of the use of line and edge detection for real-time inspection of masonry units, such as blocks or bricks. They fit a model to the edge data, which can be used to determine the quality of the masonry unit.

Nowak et al. (1992) describes an implementation of a vision system for the classification of halved pig carcasses. The meat quality is estimated by measuring the fat thickness in the nape area. This is done by first imaging the nape area, then performing edge detection, edge thinning and line following to determine the fat's boundaries.

2.3.3 Consumer Electronics

Paik et al. (1992a) propose a new digital image stabilization system, based on an edge pattern matching technique, and a new adaptive motion decision method. Such a system is particularly useful for digital cameras, where small motions of the camera can seriously degrade the picture quality. In the system, once a motion estimation block (MEB) is located, local motion vectors (LMVs) can be estimated. These LMVs are estimated by correlating the edges in the MEB in a given frame of the input sequence to the edges in the same region, of previous frames. A number of LMVs are computed, each for a different MEB. All these LMVs are then averaged in the appropriate manner to produce the field motion vector (FMV).

To stabilize more than two image sequences, the FMVs of consecutive fields are accumulated, and the resulting motion vector is called the accumulated motion vector (AMV). These AMVs can then be used to compute the field memory read address, and the corresponding data from the field memory can be used to stabilise the image sequence.

Uchiyama et al. (1992) use edge detection in their decoding scheme for a digital still camera. The camera applies adaptive differential pulse code modulation (ADPCM) for each of the red, green and blue fields. This allows for a significant reduction in the hardware, i.e., a smaller camera, whilst still maintaining picture quality. Edge detection is used in the decoding scheme to obtain the Y (luminance) component of the picture.

Inamori et al. (1993) describe a new control method which uses edge detection and motion detection, to eliminate noise, blur and picture trails in moving pictures without degrading picture quality. The method has uses in all video applications such as TV, VCR and cameras.

Typical noise reduction systems for these applications average the pixels of the present and previous frame. Thus, when there is a moving picture, i.e., a temporal signal fluctuation, this technique generates a perceptually noticeable picture trail. The new noise suppression techniques makes use of the fact that these picture trails generally occur at the edges of the picture. So, by locating the edges, the noise reduction can be inhibited or disabled at the neighbouring pixels of these edges. The new method improves the noise reduction for moving pictures, and has been implemented for use on MUSE (Japanese HDTV).

Appelhans & Schroder (1995) use edge detection for equalisation or ghost cancellation for stationary and mobile television. The ghost or echo signal can be removed by estimating the channel using an LMS filter, and then performing equalisation. The authors present a blind equalisation algorithm where edge positions are candidates for echo parameter estimation. At these points, the algorithm estimates the delay of any echo signal as well as the echo amplitude by minimising the mean square error between the scaled original signal and the corresponding echo distortion. Edge detection is also used in the iterative algorithm to improve the estimation of the channel's impulse response.

Lee et al. (1992) describe a new multi-purpose two-layered video compression system which uses edges in its encoding scheme. Video signal compression is essential for applications such as DSM (Digital Storage Media), ISDN (Integrated Services Digital Network) and HDTV (High-Definition TV). The proposed encoding scheme achieves greater compression, without loss of picture quality, by only focusing on those areas of the video which have the greatest detail or complexity. Most of the information is lost in the areas of the video where the complexity is negligible. The complexity of the video sequence is determined by the number of edges in any given area. Thus, a block of the sequence is only fully encoded without loss of information if the number of edges exceeds a threshold.

Kadono & Yamamitsu (1993) also use edge detection for the efficient coding of HDTV. HDTV usually requires a transfer rate of 1.2Gbps, hence compression must be used to remove redundant information. The authors propose new techniques to encode HDTV at 50 Mbps. The encoder essentially consists of a DCT, quantizer and variable length encoder, but first, the chrominance signal of the input is decimated. Typical decimation filters introduce distortion, thus an adaptive decimation filter is designed. For each input pixel, the filter searches for an edge in the following pixel. The filter then selects from one of four filters depending on whether there is an edge in the previous pixel, an edge in the following pixel, edges on either side, or no edge at all. Results on images show that this adaptive decimation scheme significantly reduces the amount of distortion in the encoding process.

2.3.4 Other Computer Vision Applications

Regazzoni et al. (1996) present a system able to provide accurate quantitative estimates of the number of people present in a scene monitored by a set of cameras. This information is useful in any transport station for crowd control and traffic planning. The task is performed using a distributed multisensor data-fusion approach, with a probabilistic Bayesian network implementing the crowd estimation system. The system consists of two networks: one with low-level nodes, and another with high-level nodes. The low-level nodes use edge detection to provide estimates for the number of people in a camera's scene. The high-level nodes combine the information from all the low-level nodes to provide an estimate for the total number of people in a scene. Due to the computational overhead, edge detection is also used to focus the system's attention on to the dynamic foreground of the scene, and not the static background.

Govindaraju et al. (1990) look at the problem of face recognition in the newspaper domain. They model the face with features such as edges, lines, arcs, etc. as well as the structural relationship between these features. These features are then matched against the features of a face model, and successful matches form hypotheses. Each hypothesis is checked to determine if it is a face using a cost-functional approach.

Schneiderman & Nashman (1992) presented a successful algorithm for autonomous driving on freeways, large local roads, and two-lane rural roads. The algorithm extracts edges from the road scene. These edges are then matched to a road model in order to associate edges with valid lane markings, and remove any edges not associated with lane markings. Essentially, the autonomous vehicle is steered by tracking lane markings on the road.

Barnes & Liu (1995) also implemented a vision system to guide an autonomous robot around objects of arbitrary pose. From a camera, the robot's preprocessor extracts relative features such as the length of edges, their orientation, and the relative location of edges. These features are then matched against the object model to determine the pose and distance of the object relative to the model, and then to plan the robot's next position, to which it should move.

2.4 Performance Measures

The issue of measuring the performance of an edge detector is one that has not been resolved adequately in the literature. One of the main stumbling blocks is the very wide range of possible edge detectors, as we shall see in the next section. How do we develop a measure which can be applied to maxima-based edge detectors, zero-crossing based edge detectors, or even heuristic edge detectors, amongst others? This has not been achieved to date; usually each author defines an optimality criteria which is most relevant to their particular detector, hence the abundance of "optimal" edge detectors throughout the literature. These optimality conditions are all quantitative, and incorporate nothing of the psychovisual aspects of human vision and perception. Significant progress has been made, however, in explaining qualitatively what a good edge detector should achieve, though it is often impossible to express such criteria mathematically.

Eventually, however, the merits of any edge detector should be judged on its performance with real images rather than synthetic ones. Fundamental questions then arise, such as: does optimality of an edge detector on a simple, isolated edge profile tell us how it will perform on real images consisting of very complex structures? Indeed, what do we mean when we talk about optimality? To complicate matters even further, the set of real images used to evaluate edge detectors often varies from author to author; see e.g. Price (1986) for a good discussion on this point. There appears to be no standard of any sort when evaluating performance of edge detectors.

Thus, the ultimate test for any edge detector is how well its output correlates to those edges that are perceived as being significant by a human observer. The results from such tests are very objective, so in order to compare the edge detectors with some sort of solid mathematical background, a number of performance measures can be used, even if they tend to be overly simplistic or heuristic in nature. The following measures that we shall discuss are applicable only to synthetic images since the true edge position needs to be known *a priori*. They include *distance* and *statistical* measures (van Vliet et al., 1989), *Canny's criteria* (Canny, 1986), and a related measure proposed by Tagare & deFigueiredo (1990).

2.4.1 Distance Measures

Distance measures are based on *edge deviations*, or *error distances* (the minimum distance between the detected edge and the "ground truth"). This requires the edge positions to be known *a priori*, which, in general, can only be done for synthetic images. Some distance measures are (van Vliet et al., 1989; Abdou & Pratt, 1979; Peli & Malah, 1982)

- the mean square deviation: $MSD = \frac{1}{N_D}\sum_{i=1}^{N_D} d_i^2$,
- the mean absolute deviation: $MAD = \frac{1}{N_D}\sum_{i=1}^{N_D} |d_i|$, and
- Pratt's *Figure of Merit*: $FOM = \frac{1}{max\{N_D, N_I\}}\sum_{i=1}^{N_D}\frac{1}{1+\alpha d_i^2}$

where N_D is the number of detected edge points, N_I is the number of ideal edge points, $\alpha > 0$ is a scaling constant, and d_i is the edge deviation for edge point i. All three measures are intuitive, though there is no mathematical derivation to Pratt's FOM. This measure was used quite frequently in early papers on edge detection, but has now come into disuse, replaced by more sophisticated measures (such as Canny's criteria (Canny, 1986)).

2.4.2 Statistical Measures

Some simple quantitative *statistical measures* that can be found, generally by numerical simulations on synthetic images, include:

- the percentage of edge points detected on the ideal (desired) edge,
- the number of edge points which do not coincide with true edges,
- the mean width of a detected edge, i.e., the ratio of the total number of detected edge points to the number of ideal edge points, and
- the noise-to-signal ratio (NSR), i.e.,

$$NSR = \frac{FPE + FNE}{TPE} \qquad \text{EQ (2.1)}$$

where FPE is the number of false positive edges, FNE is the number of false negative edges, and TPE is the number of true positive edges. A FPE is a non-edge pixel which is declared to be an edge pixel, a FNE is a true edge pixel which is not declared to be an

edge pixel, while a TPE is a true edge pixel which is indeed declared to be an edge pixel.

The problem with these measures is that it is often impossible to find analytical expressions for them, and hence they must be obtained by Monte Carlo simulations.

2.4.3 Canny's Criteria

Canny (1986) was the first to summarise the three criteria that a "good" edge detector should satisfy. Although they may appear straightforward and "obvious", Canny was the first to concisely state them and incorporate them into his edge detection formulation. The criteria are:

- *Good Detection*: i.e., maximising the probability of detecting a true edge point (or pixel), and minimising the probability of identifying a false edge pixel. Both of these, in fact decrease with output signal to noise ratio (SNR), hence the objective is really to maximise the output SNR,
- *Good Localization*: all edge pixels should be as close as possible to the true edge position, and
- *Single Response Criteria*: there should only be one response for every true edge pixel (this is clearly linked to the first criteria).

By applying a linear filter $h(x)$ of support $[-W, W]$ to the ideal step $u(x)$ with additive Gaussian noise of variance σ_n^2, Canny obtained the following expressions for his criteria:

$$SNR = \frac{\left| \int_{-W}^{W} u(-x)\, h(x)\, dx \right|}{\sigma_n \sqrt{\int_{-W}^{W} h^2(x)\, dx}}, \text{ and}$$

$$Localisation = \frac{\left| \int_{-W}^{W} u'(x)\, h'(x)\ (dx) \right|}{\sigma_n \sqrt{h'^2(x)\, dx}} \qquad \text{EQ (2.2)}$$

with an average inter-maxima spacing of noise peaks of $x_{ave} = \pi \left(\frac{\int_{-W}^{W} h'(x)\, dx}{\int_{-W}^{W} h''(x)\, dx} \right)^{1/2}$.

Generally the first two criteria are simultaneously maximised, while keeping the third expression as large as possible. Note the assumptions made in deriving these expressions: that $h(x)$ is linear and has a maximum response at the edge position, and that the additive

noise is Gaussian. These assumptions are not always valid, or applicable, hence these measures (of good detection, localisation, and a single response) are limited in their usefulness.

2.4.4 Tagare & deFigueiredo's Measure

Although Canny's criteria made a significant impact on the edge detection community, Tagare & deFigueiredo (1990) correctly pointed out an error in the derivation of one of Canny's equations for his criteria measures, namely the localisation measure given in EQ (2.2). For $h(x)$ an odd-filter with only one zero (at $x = 0$), they suggested instead maximising the measure

$$Q = \int_{-\infty}^{\infty} x^2 (1 - \mu_{rel}(x)) \, dx \qquad \text{EQ (2.3)}$$

where $\mu_{rel}(x)$ is the *relative density of maxima* of the filters output, which is to be minimised in order to obtain a single response to an edge. For a filter $h(x)$, $\mu_{rel} = \exp\left(-h^2(x) / \left(2\sigma_{\zeta'}^2 \right) \right)$ where $\sigma_{\zeta'}^2 = \sigma_n^2 \int_{-\infty}^{\infty} h'^2(x) \, dx$.

As with Canny's measures, the difficulties with this measure is that it pertains only to the authors' particular filter, and there is no rigorous justification for their proposed measure (EQ (2.3)). Indeed their proof is not entirely satisfactory (see also Tagare & deFigueiredo (1994) and Boyer & Sarkar (1994) for an interesting discussion on this particular performance measure).

2.5 Edge Detectors

In this section we give a summary of a number of edge detectors proposed in the literature. We shall discuss in some detail the most common edge detectors found in the literature. The simplest edge detectors, such as the differential edge operators which are discrete approximations to either the first or second derivative, are discussed first. We then describe the biologically inspired Marr-Hildreth Laplacian-of-Gaussian (LoG) filter, which is based on finding the optimal filter that minimises the time-frequency bandwidth product, and Canny's operator developed to optimise a set of mathematical criteria (output

SNR, localisation, etc.). We shall also look at Deriche's operator, which is a recursive 1st derivative, and finally we review the neural network approaches to edge detection.

2.5.1 Differential Operators

We have already discussed that the most common type of intensity profile is the step edge. Intuitively, the easiest way to detect such a profile is to differentiate it, and then to detect the maximum (or zero-crossing in the case of the second derivative). Most of the early edge detectors were formulated along these lines, i.e., masks (or filters) that approximate either the 1^{st} or 2^{nd} derivatives. The downfall of such derivative based edge detectors, as we shall discuss, is their poor performance with noisy data.

2.5.1.1 Formulation

Consider the general 3×3 edge templates A and B that correspond to discrete derivatives in the x and y directions respectively.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

and an image window

$$Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix}$$

Then by definition (White & Schowengerdt, 1992) we can approximate the directional derivatives in the x- and y-directions as

$$\frac{\partial Q}{\partial x} = Tr(AQ) \qquad \frac{\partial Q}{\partial y} = Tr(BQ)$$

where "*Tr*" represents the *Trace* of a matrix. The estimate of the edge orientation is given by

$$\hat{\theta} = \text{atan}\left(\frac{\partial Q/\partial y}{\partial Q/\partial x}\right) \qquad \text{EQ (2.4)}$$

Clearly $\hat{\theta}$ is unchanged by any *scaling* of the input, but to make it also invariant to *bias* (i.e. input + constant) the sum of the template coefficients must equal zero. Thus

$$\begin{bmatrix}1 & 1 & 1\end{bmatrix} A \begin{bmatrix}1 & 1 & 1\end{bmatrix}^T = 0 \text{ and } \begin{bmatrix}1 & 1 & 1\end{bmatrix} B \begin{bmatrix}1 & 1 & 1\end{bmatrix}^T = 0$$

It is also reasonable to assume that the derivatives are *symmetric,* i.e., $A = B^T$. Finally, further restrictions (Lyvers & Mitchell, 1988) on the templates reduces them to the form

$$A = \begin{bmatrix}-1 & 0 & 1\\ -w & 0 & w\\ -1 & 0 & 1\end{bmatrix} \qquad \text{EQ (2.5)}$$

where w is the only free parameter. When $w = 2$ we have the *Sobel operator* (Jain, 1989), the *Prewitt* operator when $w = 1$, and the so-called *Frei-Chen isotropic operator* when $w = \sqrt{2}$. For instance, the Sobel operator is

$$A = \begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} \text{ and } B = \begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}$$

and the Prewitt operator is

$$A = \begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix} \text{ and } B = \begin{bmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{bmatrix}.$$

The approximations to the first (and second derivatives) are not only limited to these classes of templates; other templates include *Robert's Cross operator* (Gonzalez & Woods, 1992):

$$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \qquad \begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}$$

and also the discrete *Laplacian* $\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$: $\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}$.

This operator is not uniquely defined since there may be many discrete approximations to the Laplacian. Davies (1992) showed that if each pixel in the image window is subjected to noise of the same variance, then the Laplacian mask that gives minimal output noise variance is

$$\begin{bmatrix}2 & -1 & 2\\ -1 & -4 & -1\\ 2 & -1 & 2\end{bmatrix}.$$

2.5.1.2 Implementation

These templates can be thought of as matched filters, giving a large output whenever the input is similar to the template. For all the edge detectors discussed above (Sobel, Prewitt's, Robert's, etc.), we convolve each template with the input image and compute the overall gradient (for two templates) using either the L_1, L_2 or L_∞ norm, i.e., if I_x and I_y are the partial derivatives with respect to the x and y coordinates respectively, then the *overall gradient* can be approximated by one of the following:

$$M_1 = |I_x| + |I_y|$$

$$M_2 = \sqrt{I_x^2 + I_y^2}$$

$$M_\infty = max\{|I_x|, |I_y|\}$$

The edge orientation ($\hat{\theta}$) may be estimated using the ratio of the magnitudes of the partial differential equations (EQ (2.4)). In the simplest case, a pixel is defined as belonging to an edge if its edge magnitude is greater than some preset threshold. More sophisticated techniques are available that also take into account the local edge structure within a small neighbourhood when considering whether a pixel belongs to an edge, but such techniques are usually reserved for multi-template edge detection as we shall see.

2.5.1.3 Precision of 1st Derivative Operators

Since we approximate the derivative with a finite set of discrete templates, it is easy to see that errors will be introduced in the estimation of the gradient magnitude and orientation.

White and Schowengerdt (1992) investigated the effects of blurring on the accuracy of various operators for various realistic point spread functions (PSFs). Considering only the errors in the edge orientation ($\hat{\theta}$), they found that the Sobel operator performed the best for small PSF sizes, but the Prewitt operator outperformed others for larger PSF sizes.

Kitchen & Malik (1989), extending the study by Abdou & Pratt (1979), investigated the reported edge orientation and magnitude detected by the edge detector (using three different norms). They showed that often the reported edge magnitudes depend *solely* upon the edge orientation, and that the results were critically dependent upon the type of norm used. Lyvers & Mitchell (1988) showed that for the case where only single-pixel width edges were sought, the edge orientation error was a minimum with the weighting

factor of $w = 2.03$, where w is defined in EQ (2.5). Note how close this is to the Sobel operator.

White & Schowengerdt (1992) also found that the Sobel operator had a smaller edge orientation error for blurred inputs, than Robert's Cross operator. They postulated that its superior performance was due to the higher weighting of pixels close to the centre element ($w = 2$ rather than 1). Lyvers & Mitchell (1988) showed that the minimum angular error occurred for a weighting factor of $w = 2.62$ (for their single pixel wide edges).

2.5.1.4 Noise Analysis

Most of the discussion to this point has not included the effects of noise. Abdou & Pratt (1979) performed a limited investigation using Pratt's Figure of Merit (FOM) they observed that the Prewitt and Sobel operators gave substantially higher FOMs than Robert's operator. Although the Sobel operator is the best finite difference edge operator, an analysis by Lyvers & Mitchell (1988) clearly showed that it was overall the worst edge detector amongst those considered (of the non-derivative type filters) for a large range of signal-to-noise ratios (SNRs). Their conclusions are not surprising since it is well known that differentiating an *unfiltered* noisy signal only accentuates the noise. Thus, we can expect second derivative operators, such as the Laplacian, to have even worse performance in the presence of noise since taking the second derivative increases the noise level even further compared to applying the first derivative alone. Of course the effect of noise can always be reduced by increasing the support size of the operator, but this invariably leads to poorer resolution or accuracy.

2.5.1.5 Multi-template Edge Detection

Multi-template edge detection is an extension to the gradient based edge operators, in that rather than employing one or two masks, a large number are used, each being sensitive to gradient changes in a particular direction.

Examples of multitemplate edge detectors can be found in Robinson (1977) who uses among others, the *Kirsch* masks shown below (where the approximate gradient direction is shown beneath each mask):

$$\begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} \quad \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix}$$

N N/W W S/W

$$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \quad \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix} \quad \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} \quad \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix}$$

S S/E E N/E

The advantage in using more templates is the increase in accuracy of edge orientation estimation, along with greater flexibility in the post-processing of the outputs. Disadvantages include the increase in the amount of computation and complexity in handling the many outputs. The approach used by Robinson is fairly typical of multitemplate edge detection schemes, and is shown in Figure 2.4.

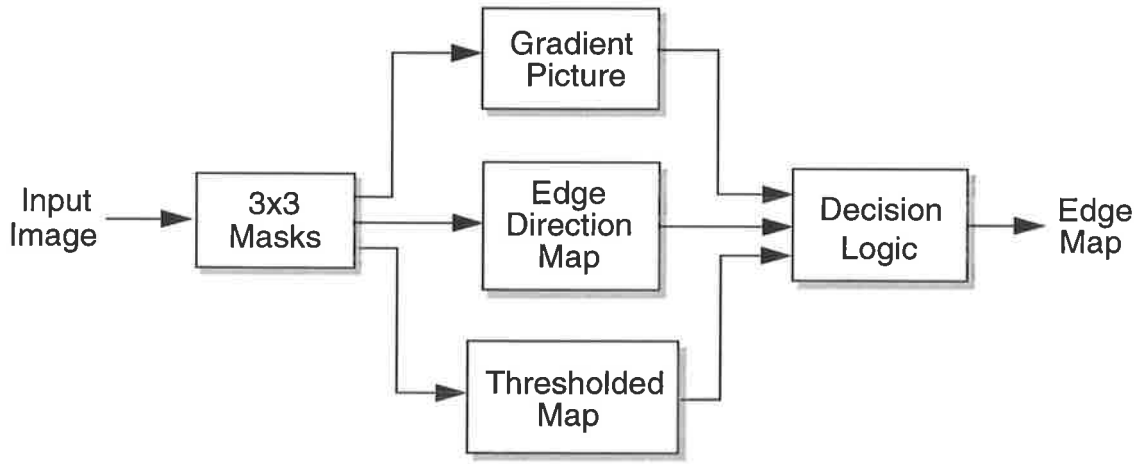


Figure 2.4 An example of a multitemplate edge detection scheme as used by Robinson (1977).

The input image is convolved with each of the masks in turn, with the mask yielding the greatest magnitude response defining the edge direction. Thus, for each pixel the magnitude of the gradient is stored along with its corresponding "direction". The magnitude is then thresholded to obtain a binary image, which in turn passes to some sort of decision logic. This decision logic may be of the form of an edge-linking process i.e., a pixel is labelled as an edge if it conforms to a predefined edge pattern specified by the surrounding edge pixels. For instance isolated edge pixels may be discarded, or the

directions of edge pixels must not vary greatly from those of the surrounding edge pixels (see Gonzalez & Woods (1992) for more information on edge- or line-linking).

Other multi-template approaches have been performed by McLean & Jernigan (1988), who use a coarse-to-fine template approach, Nevatia & Babu (1980) who extracted linear features, Brzackovic et al. (1991) who use templates of varying size, and Meer at al. (1989) who define masks that respond to only 1 stimulus.

2.5.2 The Laplacian-of-Gaussian (LoG) Operator

Neurophysiological studies of the primate visual system (Spillman & Werner, 1990), together with pyschophysical studies of phenomena like Mach Bands, have indicated that there exist two types of retinal ganglion cells in cats that have receptive fields with the shape of the Difference-of-Gaussians (DoG), the Laplacian-of-Gaussian (LoG), or the Mexican 'sombrero', as shown in Figure 2.5.

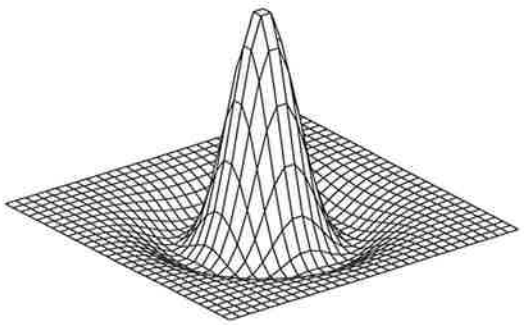


Figure 2.5 Laplacian-of-Gaussian (LoG).

Marr & Hildreth (1980) proposed that the output of one of the cells in the primate visual system, the X cell, is equivalent to the convolution of the DoG with the intensity image falling upon the receptors. At the visual cortex there are supposedly a variety of cells able to detect the zero-crossings in the output of the X cells.

2.5.2.1 Formulation

In accordance with these findings, Marr & Hildreth (1980) proposed applying the second derivative of the Gaussian to the input image, and then detecting the resulting zero crossings. They used the Gaussian to principally smooth the image and remove any high-

frequency noise. In general, the smoothing filter needs to compromise between attenuating high frequency noise, while not blurring the edges sought within the image (edges are inherently of a high frequency nature). Thus, the filter's spectrum needs to be *bandpass* with as small a bandwidth as possible, i.e., its frequency variance $\Delta\omega$ should be as small as possible. The zero-crossings in the spatial domain also need to be localised to the true edge positions; thus, the filter in the spatial domain should be *smooth* and *localised*, and its variance Δx in this domain should be small too. These two requirements are inherently conflicting, and are shown to be related by the uncertainty principle (Marr & Hildreth, 1980): $\Delta x \Delta\omega \geq \frac{1}{2}$.

The unique real filter that minimises this uncertainty product is the Gaussian (Torre & Poggio, 1986)

$$g(x, y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

whose Fourier transform is $G(u, v) = \exp\left(-\frac{1}{2}\sigma^2(\omega_x^2+\omega_y^2)\right)$, where σ^2 is the variance of the filter and determines its spatial spread, while ω_x and ω_y are the spatial frequency variables. Thus, the filter $g(x, y)$ provides the optimal trade-off between the two conflicting requirements stated above. The zero-crossings of the second derivative D^2 are then sought in the smoothed image

$$f(x, y) = D^2(g(x, y) * I(x, y)) .$$

By using the derivative rule for convolution, this is equivalent to convolving the image with the second derivative of the Gaussian

$$g''(x) = D^2(g(x, y)) * I(x, y)$$

and then locating the zero-crossings of the output.

Thus, the only parameter left to choose for the filter is the Gaussian space constant σ. Hildreth (1983) suggested using an operator of central width $w = 2\sqrt{2}\sigma$ as shown in Figure 2.6. The variance itself is intimately related to the resolution of the edge detection process. When w is small, fine or "steep" edges can be detected, whereas when w is large, broad or ramp edges can be detected. The variation of this width is exploited in the scale-space approaches to edge detection (Bergholm, 1987; Goshtasby, 1994).

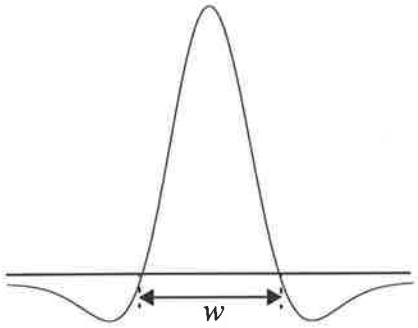


Figure 2.6 LoG with central width $w = 2\sqrt{2}\sigma$.

There is, however, no clear or easy method to determine the optimal filter size. Lunscher & Beddoes (1986) give a selection criteria in the ideal case of an infinite staircase model and square-wave edges.

2.5.2.2 Fast Implementation

The original LoG proposed by Marr & Hildreth has theoretically a very large support, and hence is quite time consuming to convolve with an image. Fast implementations of the LoG (Huertas & Medioni, 1986; Chen et al., 1987; Sotak & Boyer, 1989) make use of the fact that the 2-D Gaussian can be written as the sum of two separable 1-D Gaussians

$$\nabla^2 g(x, y) = h_{12}(x, y) + h_{21}(x, y)$$

where $h_{12}(x, y) = h_1(x)\, h_2(y)$, $h_{21}(x, y) = h_2(x)\, h_1(y)$, and

$$h_1(\zeta) = \sqrt{K}\left(1 - \frac{\zeta^2}{\sigma^2}\right)\exp\left(\frac{-\zeta^2}{2\sigma^2}\right),$$

$$h_2(\zeta) = \sqrt{K}\exp\left(\frac{-\zeta^2}{2\sigma^2}\right).$$

When σ is large, LoG filters require very large supports, but they can only extract low-frequency information since they are bandpass, hence further computational savings can be achieved by reducing the sampling frequency. The procedure is to reduce the resolution of the original image by a *decimation factor* of k (either through subsampling, or by averaging), then convolving this reduced image with a LoG of *reduced space constant* σ/k, and finally the output is *expanded* by the factor k.

Sotak & Boyer (1989) use a decimation factor depending upon the amount of aliasing allowable. They use *bilinear interpolation* to obtain the original scale. Chen et al. (1987)

claim no aliasing effect by their decimation method, and report speed-ups of two orders of magnitude over the original decomposed signal, while Huertas & Medioni (1986) also decimate the image and filter as usual, and then use the facet model (Haralick & Watson, 1981) to achieve subpixel accuracy.

2.5.2.3 Accuracy of the LoG

In Marr & Hildreth's original formulation (Marr & Hildreth, 1980) they showed that the orientation formed locally by the zero-crossings corresponds to the zero-crossings with maximum slope, providing the *condition of linear variation* holds, i.e., the intensity variation near and parallel to the edge should be linear. This assumption can break down at corners, curves, for nonlinear illumination, finite camera resolution and obviously noise. An analysis by Berzins (1984) describes the possible errors when using the LoG.

2.5.3 Canny's Operator

The motivation in Canny's (1986) formulation was to obtain mathematical expressions that correspond to criteria that a good edge detector should possess. These have already been discussed, but in summary they are (1) good *detection*, (2) good *localization*, and (3) a *single response* to an edge. Canny was one of the first to express these concisely, and to attempt a mathematical derivation based upon a numerical optimization approach. Note that while these criteria are applicable in general, it is often difficult to derive mathematical expressions similar to those of Canny for an arbitrary filter.

2.5.3.1 Formulation

For a linear, anti-symmetric filter of support $[-W, W]$ operating on an edge $g(x)$, with the edge located at $x = 0$, Canny (1986) shows that to achieve the first two criteria, both the signal-to-noise ratio (SNR) and the localization need to be maximised simultaneously, i.e.,

$$SNR = \frac{\left| \int_{-W}^{W} g(-x)\, h(x)\, dx \right|}{\sigma_n \sqrt{\int_{-W}^{W} h^2(x)}}.$$

Localization is the inverse of the standard deviation of the spatial spread of detected edges about their true positions, so clearly the greater the localisation the better the edge detectors performance. Thus,

$$Localization = \frac{\left|\int_{-W}^{W} g'(-x)\, h'(x)\, dx\right|}{\sigma_n \sqrt{\int_{-W}^{W} h'^{2}(x)}}$$

where σ_n is the standard deviation of the additive Gaussian noise. Note that increasing the SNR decreases the localization, and vice versa, hence there is an inherent trade-off between these two measures or criteria. A simple (but not unique) strategy is to optimise the *product* of these two, i.e., $SNR \times Localisation$. The third criterion, while not included in this product, is usually made as large as possible, i.e., the distance between noise generated peaks should be as large as possible. Expressed mathematically,

$$x_{max} = 2\pi \left(\frac{\int_{-\infty}^{\infty} h'(x)\, dx}{\int_{-\infty}^{\infty} h''(x)\, dx} \right)^{1/2}.$$

If this is some fraction k of the edge detector's width W, then $x_{max}(h) = kW$, and the average number of noise maxima in this region is $N_N = 2/k$. So fixing k also fixes the number of noise maxima that could lead to false responses.

2.5.3.2 Finding the Optimal Detector

In practice it is impossible to find a closed-form operator that maximises both the SNR and localisation, subject to the multiple response criteria. A solution can be found, however, using numerical optimization (or constrained optimization). For the case of a simple step edge profile, expressions for h can be derived, however the resulting expressions are very long and complicated. The general solution in the range $[-W, 0]$ is (Canny, 1986)

$$h(x) = a_1 e^{\alpha x} \sin \omega x + a_2 e^{\alpha x} \cos \omega x + a_3 e^{-\alpha x} \sin \omega x + a_4 e^{-\alpha x} \cos \omega x.$$

Note that $h(x)$ is anti-symmetric, so that $h(-x) = -h(x)$. The boundary conditions can be used to determine the unknown quantities

$$h(0) = 0 \qquad h(-W) = 0 \qquad h'(0) = s \qquad h'(-W) = 0.$$

Thus, an optimization over an infinite range of functions can be reduced to a nonlinear optimization over a few variables. Canny noted that his "optimal" step edge detector has a very similar shape to the first derivative of the Gaussian. The latter is much simpler to

implement, but is "suboptimal" in its performance, as compared to the numerical solution derived by his measures.

2.5.3.3 In Two Dimensions

Canny also chooses to approximate his filter with the 1st derivative of the Gaussian, since an n-D Gaussian can be separated into n 1-D Gaussians, thus providing a much faster implementation. In 2-D the Gaussian is

$$g(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

with the first derivative taken in some direction $\underset{\sim}{n}$, i.e., $g_n = \frac{\partial g}{\partial \underset{\sim}{n}} = \underset{\sim}{n} \cdot \nabla g$. For edge detection, $\underset{\sim}{n}$ should be orientated normal to the edge surface. As the edge orientation is not known *a priori*, a good estimate is

$$\underset{\sim}{n} = \frac{\nabla (g * I)}{|\nabla (g * I)|}. \qquad \text{EQ (2.6)}$$

This is accurate, even for smoothed step profiles, since they usually have a strong normal component. The convolution of g_n with the image I has a local maximum at the position of an edge point, thus

$$\frac{\partial}{\partial \underset{\sim}{n}} (g_n * I) = 0$$

or

$$\frac{\partial^2}{\partial \underset{\sim}{n}^2} (g * I) = 0 \qquad \text{EQ (2.7)}$$

with an edge strength of $|g_n * I| = |\nabla (g * I)|$. Thus, EQ (2.7) can be used to find the edge position, with EQ (2.6) providing an estimate of the edge strength. An edge detector can then be implemented in an arbitrary number of dimensions by first convolving the image with an n-D separable Gaussian filter.

2.5.3.4 Canny's Localisation Measures

Canny's derivation of localisation makes use of the formula

$$E\left[H_n'(x_o)^2\right] = \sigma_n^2 \int_{-W}^{W} h'^2(x)\, dx \qquad \text{EQ (2.8)}$$

where E is the expectation operator, $H_n(x)$ is the root-mean-square response to the noise only, i.e.,

$$H_n = \sigma_n \sqrt{\int_{-W}^{W} h^2(x)\,dx}$$

and x_0 is the detected edge position, whose spatial variance is inversely related to localization.

Tagare & deFiguerado (1990) correctly pointed out that EQ (2.8) only holds when evaluated at the *same position* for each realisation, i.e., x_0 needs to be constant, but the variation in x_0 is exactly what is sought by Canny. Clearly, Canny's localisation measure is in error, and a suggested correction is given by Tagare & deFiguerado (1990).

Kakarala & Hero (1992) show that the *Cramer Rao lower bound* on achievable edge localization, for the 1st derivative of the Gaussian, is about 2.2 times smaller than that achievable with Canny's "optimal" operator. The Cramer Rao bound is a lower bound on the variance of an unbiased estimator.

2.5.4 The Deriche Operator

Deriche (1990) in his paper introduced a recursive filtering structure that reduces the computational effort for smoothing an image and performing the first derivative of an image. The key to his approach is first the use of recursive filtering and using an exponentially based filter family. The aim was to improve the computational efficiency as compared to the direct or frequency domain methods.

2.5.4.1 Recursive Filtering

Consider the convolutional operation relating the input sequence $x(i)$ to the output sequence $y(i)$ of a causal, non-recursive system is

$$y(i) = \sum_{k=0}^{N-1} h(k)\,x(i-k)\ .$$

The transfer function of this system is:

$$H(z^{-1}) = \sum_{k=0}^{N-1} h(k)\,z^{-k}.$$

To implement this requires a large number of operations particularly if the length of the sequence is large. Recursive filtering deals with the problem of finding a recursive system whose transfer function matches as closely as possible the transfer function of the non-recursive system. Thus, the problem is of finding the coefficients of the recursive sequence $y_a(i)$ of order n, where

$$y_a(i) = \sum_{k=0}^{m-1} b_k x(i-k) - \sum_{k=1}^{n} a_k y_a(i-k)$$

whose transfer function $H_a(z^{-1})$,

$$H_a(z^{-1}) = \sum_{k=0}^{m-1} b_k z^{-k-1} / \left(1 + \sum_{k=1}^{n} a_k z^{-k} \right)$$

best approximates the transfer function $H(z^{-1})$. This is done by finding the coefficients of the sequence $h_a(k)$ which minimises the least-squares error

$$E = \sum_{k=0}^{\infty} (h(k) - h_a(k))^2 .$$

where the sequence $h_a(k)$ is the impulse response of the recursive system as defined by $H_a(z^{-1})$,

$$H_a(z^{-1}) = \sum_{k=0}^{\infty} h_a(k) z^{-k} .$$

By using a recursive system instead of a non-recursive one reduces the number of operations per output element from N to $n+m$. The coefficients can be chosen so that the recursive filters matches the Gaussian filter, but Deriche found it better to use a new family of exponential filters as it has good edge detection properties and more importantly, they can be implemented exactly using recursive filters. Furthermore, the coefficients of this filter can be easily determined analytically.

2.5.4.2 Smoothing with a Second Order Recursive Filter

The smoothing filter proposed by Deriche is given by

$$S(n) = k(\alpha|n| + 1) \exp(-\alpha|n|) \qquad \text{EQ (2.9)}$$

The normalisation requirement is $\sum_{n=-\infty}^{\infty} S(n) = 1$, which leads to the constant

$$k = \frac{(1 - \exp(-\alpha))^2}{1 + 2\alpha\exp(-\alpha) - \exp(-2\alpha)}.$$

The second order recursive realisation of the smoothing filter $S(n)$ is given by $y(n) = y_1(n) + y_2(n)$ for $n = 1, \ldots, M$, where

$$\begin{aligned} y_1(n) &= k[x(n) + \exp(-\alpha)(\alpha - 1)x(n-1)] \\ &\quad + 2\exp(-\alpha)y_1(n-1) - \exp(-2\alpha)y_1(n-2), \\ y_2(n) &= k[\exp(-\alpha)(\alpha + 1)x(n+1) - \exp(-2\alpha)x(n+2)] \\ &\quad + 2\exp(-\alpha)y_2(n+1) - \exp(-2\alpha)y_2(n+2), \end{aligned}$$

with initial conditions $x(0) = 0$, $y_1(0) = 0$, $y_1(-1) = 0$, $x(M+1) = 0$, $x(M+2) = 0$, $y_2(M+1) = 0$, and $y_2(M+2) = 0$.

The above equations represent an efficient way of smoothing a signal with a recursive filter. The parameter α can also be adjusted to vary the amount of smoothing or noise suppression, without increasing the number of operations per output.

2.5.4.3 First Derivative with the Second Order Recursive Filter

The first derivative of the smoothing operator as given by EQ (2.9) is

$$D(n) = kn\exp(-\alpha|n|) \qquad n = -r, \ldots, -1, 0, 1, \ldots, r \qquad \text{EQ (2.10)}$$

where k is a constant chosen to ensure no response to a constant input signal. Its value is given by $k = \frac{-(1 - \exp(-\alpha))^2}{\exp(-\alpha)}$, where α is a parameter which determines the shape of the filter, i.e., the filter's energy. A small value of α yields a large-sized operator which can deal well with noise, whereas a large value of α yields a small-sized filter which has good resolution, but poor noise properties.

If we consider a Gaussian filter of width σ, then the Deriche filter has the same energy as the Gaussian if α is chosen according to the following:

$$\alpha \approx \frac{2.5}{\sigma\sqrt{3.2}} \qquad \text{EQ (2.11)}$$

With this value of α, the Deriche first derivative appears very similar to Canny's operator (although it is not intended to emulate it). We can see the comparison between the two filter in Figure 2.7. Typical values of α lie in between 0.5 and 1.5, depending upon the amount of smoothing desired.

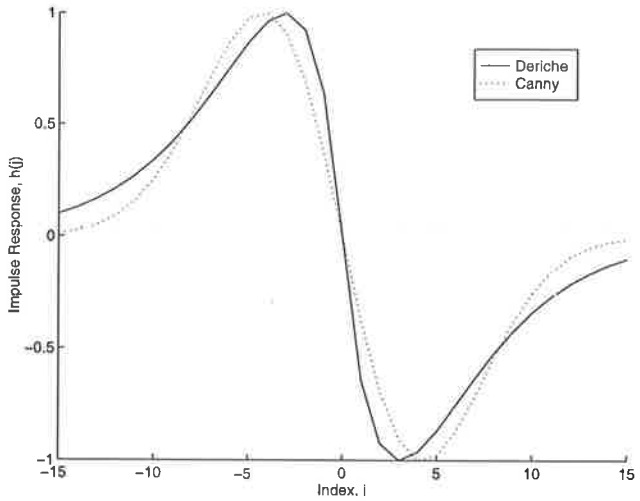


Figure 2.7 Comparison between Deriche first derivative and Canny's operator, when the width of the Gaussian, σ, and α of Deriche are chosen according the relationship of EQ (2.11).

2.5.5 Neural Network Edge Detectors

A neural network is a massively parallel distributed processor that can acquire knowledge through a learning process and store it for future use (Haykin, 1994). Such neural networks are, in general, not easily adapted to image processing tasks such as edge detection due to the dynamic and random nature of the input. We will now discuss two neural network edge detectors found in the literature that have achieved some success. They are the Adaptive Linear neural network (ADALINE) and the Hopfield-Tank neural network.

2.5.5.1 ADALINE

The Adaptive Linear Neuron, or ADALINE (Widrow et al., 1988), is an adaptive threshold logic unit. The ADALINE can operate either in the *retrieval phase* (as shown in Figure 2.8) or in *learning phase*. ADALINEs have found applications in adaptive filtering, adaptive channel equalisation, and speech recognition among others.

For an N-element input vector $\underset{\sim}{x}(t) = \left[\begin{array}{cccc} x_0(t) & x_1(t) & \ldots & x_N(t) \end{array} \right]$, with weights given by $\underset{\sim}{w}(t) = \left[\begin{array}{cccc} w_0(t) & w_1(t) & \ldots & w_N(t) \end{array} \right]$. the output is given by

$$\underset{\sim}{y}(t) = \underset{\sim}{x}(t)\,\underset{\sim}{w}(t)^T = \sum_{j=0}^{N} x_j(t)\,w_j(t)$$

The basic weight w_0, and the constant input x_0, control the threshold level used to obtain the binary output via the nonlinear function G, $q(t) = G(y(t))$ where G is typically the sign, step or sigmoidal function. In the retrieval phase, we have the desired output, and the weights are chosen to minimise the difference between this desired output and the actual output using the Least-Mean-Squares (LMS) algorithm.

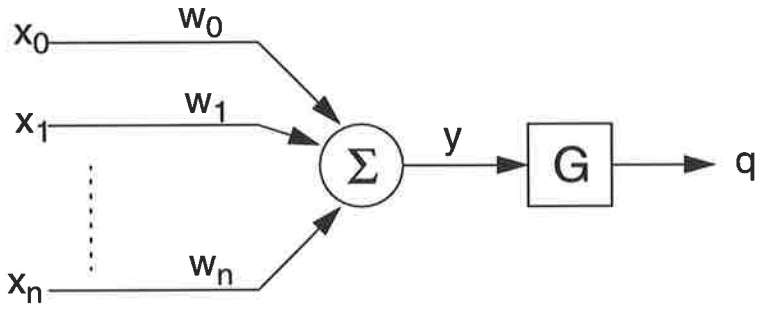


Figure 2.8 Block diagram of the retrieving phase of the ADALINE.

Its application to edge detection is given in Paik et al. (1992b). If I_i, which is a small window of the input, is the lexicographically ordered input with local mean m_i, then the input to the ADALINE is defined as belonging to either the low side of the edge, no edge, or the high side of the edge respectively, i.e.,

$$v_j = \begin{cases} -1 & \text{if } I_j \le m_j - \delta \\ 0 & \text{if } m_j - \delta \le I_j \le m_j + \delta \\ 1 & \text{if } I_j \ge m_j + \delta \end{cases}$$

where $v_j = -1, 0$ or 1 for the low side of an edge, no edge, or high side of an edge, respectively. Also $\delta = \frac{\tau\sigma^2}{E} + \frac{E}{2}$, where τ is the probability of occurrence of an edge, $2E$ is the edge amplitude, and σ^2 is the noise variance. These parameters must all be estimated or guessed heuristically.

For 3×3 input windows there are 12 possible pairs of bidirectional edges corresponding to edges in the direction $0°, 45°, 90°, 135°$, and their complements. For example, the $0°/180°$ pair of edges are

$$\begin{bmatrix} 1 & X & -1 \\ 1 & X & -1 \\ 1 & X & -1 \end{bmatrix} \qquad \begin{bmatrix} -1 & X & 1 \\ -1 & X & 1 \\ -1 & X & 1 \end{bmatrix}$$

where X represents *don't care*, a 1 signifies a high part of an edge, and -1 signifies the low part of an edge.

The 12 templates are then compared to the input v_j - if any of these templates have a minimum number of matching elements with the template, then an edge is said to be present. The problem with this method is in determining τ, the probability of edge occurrence, which must be guessed (for real images). The edge amplitude $2E$, and the noise variance σ^2, also need to be estimated. There are restrictions also on the types of edges that can be detected since the templates must be specified *a priori*. Also, by going to larger neighbourhood sizes, such as 5×5 windows, many more templates are needed, hence the computational load increases considerably.

2.5.5.2 Hopfield Edge Detection

The Hopfield-Tank (H-T) neural network (Hopfield & Tank, 1986) has been used in many areas such as associative learning, or computational networks, to solve optimization problems. For edge detection the principal papers are by Chao & Dhawan (1994a, 1994b)

In a H-T network there is a neuron for every input pixel, and each neuron is connected to all other neurons except itself. The state of each neuron represents the normalised gray scale of the input pixel. The weight of connection between two neurons represents the rate of change of pixel intensity between those neurons. Edge detection is performed as follows:

1. initialise the state of neurons as stated above,
2. randomly pick a neuron, and
3. calculate the dynamic function for that neuron

$$\frac{du_{ik}}{dt} = -\frac{u_{ik}}{\tau} + \sum_j \sum_l T_{ikjl} V_{jl} + I_\alpha$$

where *u* is the state of the neuron, T_{ikjl} is the connection strength between neurons *ik* and *jl* and τ, I_0 are constant. V_{jl} is given by

$$V_{jl} = \frac{1}{2}\left(1 + \tan\left(\frac{u_{ik}}{u_0}\right)\right) \qquad \text{EQ (2.12)}$$

4. Update the state of the neuron using EQ (2.12).
5. Repeat from step (2) until convergence.

The authors show that when the energy of the H-T network converges to a minimum, the neuron state values close to 1 represent edges, while state values close to 0 represent no edges. Thresholding then produces the binary edge map.

The problem with this network is the difficulty in analysing the output with relation to the network parameters. The algorithm is also very computationally intensive, taking many millions of iterations before converging.

2.6 Conclusions

This Chapter has summarised and reviewed many important concepts that are to be used throughout the thesis. We began with a simplified description of some properties of electromagnetic light, followed by a discussion of the most common edge intensity profiles found in real images, and also what underlying physical edges most likely give rise to such profiles. We showed that the step edge is often found in practice, being generated from occluding edges.

We discussed a number of edge detection performance measures, and then we reviewed why edge detection is important for biological and computer vision, followed by some applications of edge detection. In particular we looked at the industrial applications such as parts or component inspection, as well as the applications to consumer electronic, among others. Clearly, edge detection is very widely used in industry and constitutes an important part of many computer vision systems.

Finally we presented a comprehensive review of edge detection theory, followed by a review of the more popular and successful edge detector schemes available in the literature. They include derivative schemes as well as two neural network approaches.

Chapter 3 Cellular Neural Networks: a Review

3.1 Introduction

The Cellular Neural Network (CNN) paradigm is a powerful framework for analogue nonlinear parallel processing arrays defined on a grid. CNNs are suited to problems which are defined in space-time e.g., image processing tasks, and partial differential equations (PDEs). These problems are all characterised by the fact that the necessary information and interactions are generally constrained to small local areas, rather than large global ones.

Thus, the main difference between CNNs and other Neural Network (NN) paradigms is that all information is processed locally, although global processing is still indirectly possible by dynamic diffusion of information. This local processing property of CNNs makes them amenable to either electronic or optical implementations, which is generally impossible, or very difficult, to achieve with the other forms of neural networks.

Traditionally, if operations are logical and requiring only a few bits, then the von Neumann cellular automaton is the ideal tool. If the operations are numerical (typically 8, 16, 32, or 64 bits) then systolic arrays are the best approach. If however, the signal values are continuous and/or analogue real-time operations are necessary, then the CNN is a good solution with respect to both speed and time. CNNs are also ideally suited for VLSI/ CMOS implementations.

In this Chapter we present an overview of the CNN architecture and system operation in the general case, and then look at a number of particular variants of this general model. The stability of some of these variants is discussed, as well as their application. Finally

shunting inhibitory Cellular Neural Networks (SICNN) are introduced and discussed in some detail, in particular their derivation and stability issues.

3.2 Cellular Neural Networks

In this section a brief general overview is given of both CNN system architecture and operation, including some fundamental definitions and the most general equations defining its operation. This section also includes a discussion on the numerous types of cell grids possible, and how local interactions can cause a global flow of information throughout the network. We also briefly mention the stability issue of CNNs.

3.2.1 CNN Architecture

The basic unit of the CNN is referred to as a *cell.* It generally contains linear and non-linear circuit elements, usually in the form of resistors and capacitors in its electrical implementation. Each cell is only connected to the cells in its local neighbourhood, hence only local interaction occurs; though *global processing* is still possible via dynamical information diffusion.

CNNs can be defined over any dimension, though it is much easier to visualise them in 1-D or 2-D. In Figure 3.1 we show a 2-D CNN defined on a square grid, with each cell connected, or interacting, with only its immediate neighbours.

If we consider a 2-D $M \times N$ CNN with a total of MN cells in M rows and N columns, then we denote the cell in the i^{th} row and j^{th} column as $C(i,j)$. Each cell is only connected to other cells within its local neighbourhood, hence the inherent local processing property of CNNs. This neighbourhood of $C(i,j)$ is called its r-neighbourhood or neighbourhood size and is defined by Chua & Yang (1988):

$$N_r(i,j) = \{C(k,l) \,|\, max\{|k-i|, |l-j|\} \leq r, 1 \leq k \leq M, 1 \leq l \leq N\} \qquad \text{EQ (3.1)}$$

where r is a positive integer. It is easy to show that this neighbourhood system exhibits a symmetry property; that is, if $C(i,j)$ is a member of $N_r(k,l)$, then $C(k,l)$ is also a member of $N_r(i,j)$.

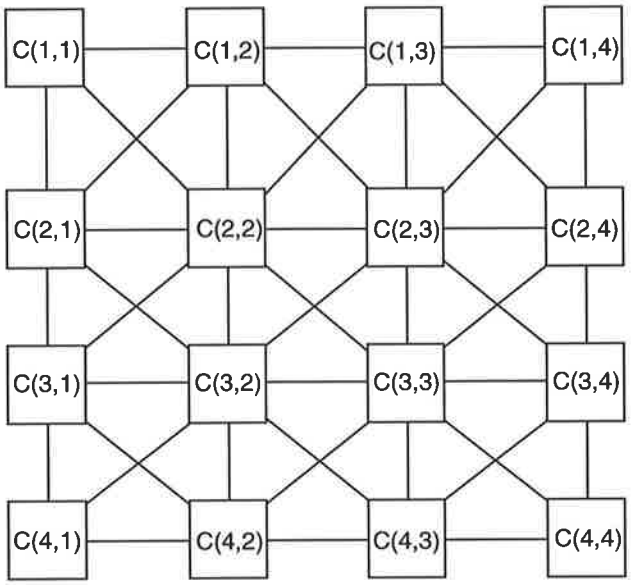


Figure 3.1 2-D CNN defined over a 4×4 square lattice.

The cell grid can be a planar array with rectangular, triangular, or hexagonal geometry, a torus, or a 3-D array. Cells may also be identical or they may belong to one of a few different types. More than one connection network may also be present, each with different neighbourhood sizes - such as short range interaction and subsystem connections. The neighbourhood size may be as large as the network, in which case we have a *fully connected* network. Cellular networks, however, are generally implemented with only small neighbourhoods.

3.2.2 System Operation

The CNN is a dynamical system operating either in continuous or discrete time. A general form of the cell dynamical equations, as given in (Chua & Yang, 1988), is as follows:

For *continuous time*:

$$\frac{dx_j}{dt} = g\left[x_j(t)\right] + \sum_{k \in N_f} A_{\mu_j}\left(x_j\big|_{(t-\tau,\,t]}, y_j\big|_{(t-\tau,\,t]}, p_j^A\right) + \sum_{k \in N_f} B_{\nu_j}\left(x_j\big|_{(t-\tau,\,t]}, I_j\big|_{(t-\tau,\,t]}, p_j^B\right) + u_j(t) \qquad \text{EQ (3.2)}$$

$$y_j(t) = f\left(x_j\big|_{(t-\tau,\,\tau\,]}\right) \qquad \text{EQ (3.3)}$$

For *discrete time*:

$$x_j(n) = g[x_j(n)] + \sum_{k \in N_f} A_{\mu_j}\left(x_j|_{(n-m,n]}, y_j|_{(n-m,n]}, p_j^A\right) + \sum_{k \in N_f} B_{\nu_j}\left(x_j|_{(n-m,n]}, I_j|_{(n-m,m]}, p_j^B\right) + u_j(n) \quad \text{EQ (3.4)}$$

$$y_j(n) = f\left(x_j|_{(n-m,n]}\right) \quad \text{EQ (3.5)}$$

In EQ (3.2) to EQ (3.5) x, y, I and u denote the cell state, output, input, and bias respectively; j and k are the cell indices; g is the local instantaneous feedback function; N is the neighbourhood function; p^A and p^B are arrays of parameters; and the notation $z|_T$ denotes the restriction of the function z to the interval T. In EQ (3.2) and EQ (3.3), t is time, Δ_t is the differential operator, τ is the memory duration, while in EQ (3.5) n is time and m is the units of delay. In both sets of equations A is the neighbourhood feedback functional, and similarly B and f are the input and output functionals, respectively. These equations represent the most general dynamic equations possible for CNNs. A number of particular implementations are discussed in the next section.

3.2.3 Stability

As with all dynamical systems, stability is an important issue with CNNs. For stability, the network must converge to a finite number of states. To date, the most general form of the CNN cannot be proven to be stable, but stability can be shown for some subsets of the general model. The interested reader may find further details in (Chua & Wu, 1992).

3.3 Variants of CNNs

From such a broad and general definition of the CNN many variants are clearly possible. Generally each of these variants is developed to suit a particular application. We shall review the variations of the general model such as the specific forms of the activation function, cell grid structure, template model, and discrete time implementation. Examples of these include polynomial or linear activation functions, uniform and non-uniform grid structures, and space-invariant and time-variant templates.

3.3.1 Polynomial CNN (P-CNN)

A variant of the general CNN is the *Polynomial* CNN (P-CNN) described by Barone et al. (1993), which has as its local feedback function an odd-order polynomial function. A third order polynomial function is shown in Figure 3.2. In the case of pattern recognition such functions act as shape attractors. The use of higher order polynomials also raises the number of possible distinct equilibria of the cell.

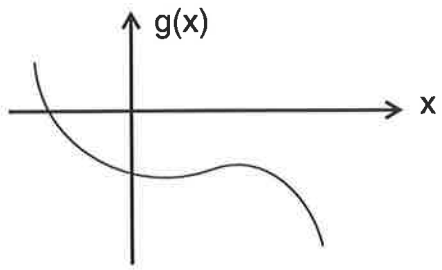


Figure 3.2 A third order polynomial local feedback function.

3.3.2 Non-linear, Delay Type and Non-Uniform Grid CNNs

Rather than having two linear controlled sources

$$A(i,j;k,l)\,v_{ykl} \text{ and } B(i,j;k,l)\,v_{ukl}$$

associated with cell $C(i,j)$ and neighbours $C(k,l)$, non-linear and delayed controlled sources can be employed (Roska & Chua, 1992), such as

$$\hat{A}_{ij,kl}(v_{ykl}, v_{yij}) + A^{\tau}{}_{ij,kl}v_{ykl}(t-\tau) \text{ and } \hat{B}_{ij,kl}(v_{ukl}, v_{uij}) + B^{\tau}{}_{ij,kl}v_{ukl}(t-\tau)\,.$$

We can possibly have $\tau = \tau_{kl}$. The structure of the non-linearity is that it is at most a function of two variables: the output voltages of cell $C(i,j)$ and its neighbour $C(k,l)$.

Motivated partly by neurobiological structures, Roska & Chua (1992) introduced a non-uniform grid CNN, or a *Non-Uniform Processor* CNN (NUP-CNN). An example of a NUP-CNN is shown in Figure 3.3 for two different processors (black and white). All interprocessor connections are space-invariant.

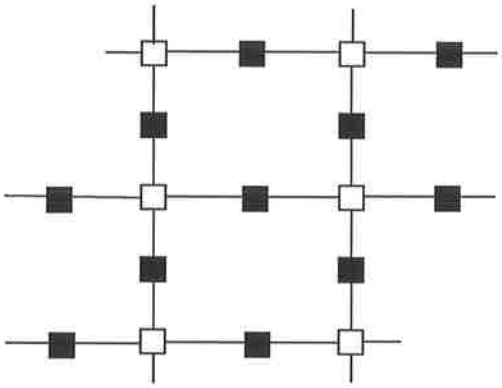


Figure 3.3 Example of Non-Uniform Processor CNN (NUP-CNN). The black and white cells indicate two different types of processing units.

CNNs can have more than one type of cell processor and/or more than one neighbourhood size, thus they are referred to as *Multiple Neighbourhood Size* CNNs (MNS-CNNs). Figure 3.4 is a diagram of a MNS-CNN. Layer A (white cells) is a finely connected CNN with neighbourhood size $r = 1$. While layer B (dark cells) is a coarser grid with a neighbourhood size of $r = 2$ (connected to grid A). Only one cell of layer B is shown for simplicity.

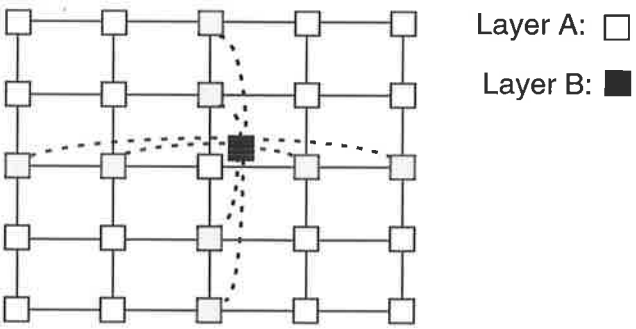


Figure 3.4 Example of a Multiple Neighbourhood Size CNN (MNS-CNN). White nodes indicate a finely connected CNN, and black nodes indicate a coarser grid. The r-neighbourhood for the white cells is 1, and 2 for the black ones.

3.3.3 Discrete-Time CNNs (DT-CNNs)

As opposed to continuous-time CNNs, discrete time CNNs (DT-CNNs) (Harrer & Nossek, 1992) have clocked variables and a comparator for their non-linear function. They are defined by the following 1-D algorithm:

$$x_j(n+1) = a_j^i y_l(n) + b_j^i u_l + I_j$$

$$y_j(n) = f(x_j(n-1)) = \begin{cases} 1 & \text{if } x_j(n-1) \geq 0 \\ -1 & \text{if } x_j(n-1) < 0 \end{cases}$$

Here j denotes the cell of interest, i is the cell in the neighbourhood of cell j, and n is the discrete time variable. The distinction from cellular automata is the continuously valued template coefficients and inputs.

In continuous time CNNs, the propagation speed depends upon the derivative of x_i, which in turn depends upon the template and input/output signals. Advantages of DT-CNNs include *constant* propagation time, *simpler* simulation (no numerical integration required), and *insensitivity* to template coefficients if they are chosen wisely.

3.3.4 Time-Variant Template DT-CNN

A more generalised architecture for DT-CNN is the extension to time-variant templates (Harrer, 1993). A template is normally a matrix with numerical values describing the amount of interaction between neighbouring cells. Time variant DT-CNNS have cyclic templates, that is, templates whose coefficients are changed at every iteration step, and with the entire set of templates applied periodically in a cyclic manner. With such a paradigm, the hardware can be reduced, hence the realisation can be simplified. Some applications of time variant template DT-CNNs include half-toning and skeletonization.

3.3.5 Applications of CNNs

Many CNNs have been proposed, each designed with a particular task in mind. CNNs have found wide applicability in image processing, particularly where local processing of information is either necessary or advantageous. Significant and successful applications of CNNs include:

- *Feature extraction*: Slot (1992) demonstrated the application of CNNs to binary images. The user specifies the feedback and feedforward operators depending upon the desired features to be extracted, and the CNN is then able to reconstruct an output which is a modified version of the input, with the desired outputs emphasised in greater detail.

- *Character recognition:* Sziranya & Csicsvari (1993), and Suzuki et al. (1992) employed CNNs to extract the necessary features of the input, which can then be used by a classification network to identify handwritten characters. The recognition rates for both networks are around the 90% level, with extremely fast recognition speeds due to the inherent parallelism. Sziranya & Csicsvari (1993) were able to identify 100,000 characters/second with a recognition rate of 95% when implemented in hardware.

- *Motion detectors:* Cimagalli et al. (1993) detect, in real-time, the trajectory of moving objects in a noisy environment. Roska et al. (1992) define various templates to detect different types of motion, where the discrete-time inputs are fed into the network, and the resulting steady-state outputs give the necessary information for estimating the direction and magnitude of the velocity vector. See also Shi et al. (1993).

- *Spatial recognition*: Perez-Munuzuri et al. (1993) use a Chua circuit to implement spatial recognition, i.e., recognising open curves from closed ones, and locating the shortest path between the two locations.

- *Evaluation of logical boolean functions*: Galias (1993) employs a time varying template CNN to define an arbitrary boolean function on the r-neighbourhood.

- *Halftoning*: Crounse et al. (1993) were able to reproduce more faithful binary reproductions of the original image with a CNN, than those produced by error diffusion, a standard algorithm for half-toning.

- *Hole-filler* (Matsumoto et al., 1990a); *shadow-detector* (Matsumoto et al., 1990d); *image thinning* (Matsumoto et al., 1990b); *connected component detector* (Matsumoto et al., 1990c; Cruz & Chua, 1991), and *Associative memories* (Liu & Michel, 1993).

3.4 Shunting Inhibitory CNNs (SICNNs)

Another particular variant of the general CNN architecture defined above, is the Shunting Inhibitory Cellular Neural Network (SICNN) (Bouzerdoum, 1994; Bouzerdoum & Pinter, 1993). We discuss it here in greater detail since our edge detection and enhancement operator will be developed from such a network. This nonlinear network can reasonably model the shunting inhibition phenomenon as discussed previously and evident in the

Mach bands. Shunting inhibition is characterised by the local interactions, hence they are clearly more amenable to be modelled by CNNs than the usual NNs.

We begin by reviewing the derivation of the SICNN, eventually arriving at the state equation that will be used throughout this thesis. Following this we shall state the necessary conditions for this system to be BIBO stable and convergent.

3.4.1 Derivation

The equivalent electrical circuit of a biological cell, or neuron, was given previously in Figure 1.5. The nodal equation for this circuit is (Bouzerdoum, 1991):

$$C_m \frac{dV_m}{dt} + g_e (V_e + V_m) - g_r (V_r - V_m) - g_s (V_s - V_m) = 0 \qquad \text{EQ (3.6)}$$

where V_r, and g_r are the lumped resting potential and conductance respectively, g_e and g_s are the excitatory and inhibitory synaptic conductances, respectively, in series with the synaptic batteries V_e and V_s, respectively, while C_m and V_m are the membrane capacitance and voltage, respectively.

Now if we designate ΔV to be the deviation of the membrane voltage from the resting potential, i.e., $V_r - V_m = \Delta V$, and with $V_s = V_r$, then the change with time of V_m relative to V_r is described by the differential equation

$$\frac{d}{dt}\Delta V = \frac{g_e}{C_m}(V_e + V_m) - \frac{g_r}{C_m}(\Delta V) - \frac{g_s}{C_m}(\Delta V) . \qquad \text{EQ (3.7)}$$

Each cell may then be represented by an electrically independent circuit, as in Figure 1.5, with x_{ij} representing the deviation of the membrane voltage from the resting potential of the cell $C(i,j)$ at the $(i,j)^{th}$ position of the lattice, and $f(x_{ij})$ denoting its firing rate. We assume that the inhibitory synapses of a cell are controlled by the activity of neighbouring cells. We also assume that the shunting conductance of a cell is the sum of the conductances of all the individual inhibitory cells. If each one of these is proportional to the firing rate of the cell controlling it, then we can write the shunting conductance g_s of $C(i,j)$ as

$$\frac{g_s}{C_m} = \sum_{C(k,l) \in N_r(i,j)} w_{ij}^{kl} f(x_{kl})$$

where the coefficients w_{ij}^{kl} and C_m are positive constants, $f(x_{kl})$ is the output of cell kl, and w_{ij}^{kl} is the weighting given to its inhibitory effects on cell ij. In other words, the inhibition exerted on a cell is a weighted sum of the outputs of surrounding cells within the appropriate neighbourhood.

The conductance g_e is controlled by the excitatory inputs that work to increase the membrane conductance to sodium Na^+ ions. We assume that the cell's input, I_{ij}, controls the excitatory current i_e, in which case we have

$$\frac{i_e}{C_m} = \frac{g_e}{C_m}(V_e + V_m) = I_{ij}(t) .$$

The remaining term is the decay factor of excitation:

$$\frac{g_r}{C_m} = a_{ij} .$$

Then EQ (3.7) becomes

$$\frac{dx_{ij}}{dt} = I_{ij}(t) - a_{ij}x_{ij} - \sum_{C(k,l) \in N_r} w_{ij}^{kl} f(x_{kl})\, x_{ij} . \qquad \text{EQ (3.8)}$$

It is clear from this equation that the interaction effects come from all cells in the local r-neighbourhood of cell (i, j), hence the local nature of information exchange. Also note that this is a particular implementation of the general CNN, given by EQ (3.2)-EQ (3.5).

In EQ (3.8), x_{ij} represents the input intensity to cell ij, I_{ij} is its input, a_{ij} determines the rate at which any excitation decays away, w_{ij}^{kl} is the weighting given to the output of cell kl to cell ij, and $f(x_{kl})$ is the output of cell kl.

3.4.2 Stability Analysis

A dynamical system is one where the state of that system changes with time and depends upon both the state itself and the input to that system (Sandefur, 1990). A dynamical system is bounded-input bounded-output (BIBO) stable if the output, initialised at an arbitrary initial state, is bounded when the input is bounded (Ogata, 1987).

Bouzerdoum & Pinter (1993) were able to prove that if the activation function f is *continuous* and *non-negative* on the entire real axis, that is,

$$f(\zeta) \geq 0 \quad \text{for } \zeta \in (-\infty, \infty) ,$$

then the SICNN is a *BIBO* stable dynamical system.

Given a dynamical system $\dot{x} = f(x, t)$ where x is the state vector, then the equilibrium state is the state x_e where $f(x_e, t) = 0$. Such a dynamical system is said to be *convergent* if every trajectory converges in the steady-state to an equilibrium point (Sandefur, 1990). The *Lyapunov function* $V(x, t)$ (Ogata, 1987) is a positive definite scalar function which is continuous, along with its first partial derivatives, in a region about the origin. It has a time derivative which, when taken along the trajectory, is negative definite (or negative semidefinite). If $V(x, t)$ is positive definite and its derivative $\dot{V}(x, t)$ is negative semidefinite, then the system is stable (though not necessarily asymptotically stable).

Now suppose the following conditions are satisfied:

1. *Symmetry* in the SICNN interaction weights, $w_{ij}^{kl} = w_{kl}^{ij}$.
2. The activation function f is *continuous*, *nonnegative*, and *decrescent*; that is,

$$\xi \dot{f}(\xi) \geq 0 \quad \text{for} \quad \zeta \in (-\infty, \infty) ,$$

then the SICNN is a dynamical system with a global Lyapunov function in $\mathcal{R}^{mxn}$ defined as

$$V(x) = -\sum_{(i,j)} I_{ij} \int_0^{x_{ij}} [\,(\dot{f}(x))/\zeta]\, d\xi + \sum_{(i,j)} a_{ij} f(x_{ij}) + \frac{1}{2} \sum_{(i,j)} \sum_{C(k,l) \in N_r} w_{ij}^{kl} f(x_{ij}) f(x_{kl}) . \quad \text{EQ (3.9)}$$

Every trajectory of this function asymptotically approaches the set of all equilibrium points. Furthermore, if the input pattern has the same polarity, then each of these trajectories converges to an isolated equilibrium point (Bouzerdoum & Pinter, 1993).

3.5 Conclusions

Since the entire work of this thesis is based upon a particular model of the CNN, we began by reviewing CNN theory. We started with the general CNN system architecture, operation and stability, and then reviewed a number of variations and applications of this model.

We discussed the use of CNNs for modelling visual phenomenon such as Mach bands. The cellular structure of CNNs and their local interactions make them particularly suitable

for modelling such visual effects, as each cell of the CNN consists of the equivalent electrical circuit of a biological neuron. From this circuit, the cell's state equation can be derived, showing clearly inhibition of the shunting or multiplicative type. The stability of such a model was also discussed.

Chapter 4 Response Properties of SICNN Systems

4.1 Introduction

This Chapter presents a detailed analysis of SICNN systems - both recurrent and feedforward - which is the first step to understanding how such systems behave along with their operating characteristics. Although a large part of the theory and results presented here is of a general nature and not necessarily related to edge detection or enhancement, we shall use it throughout the remainder of this thesis.

We begin by describing the recurrent SICNN, which was defined in the previous Chapter, and give a means of solving for its steady-state. We also investigate its response to various inputs, such as constant and step-edge signals.

We define the feedforward SICNN and describe how it performs shunting inhibition on step edges. We discuss the implementation and thresholding issues, and the advantages of the SICNN over some existing edge detectors. We also derive its impulse response using perturbation analysis and linearisation. From this impulse response, the frequency response is derived and plotted for the symmetrical and asymmetrical weight distributions, and the DC gain is also investigated. With the frequency response, we define and discuss a number of quantities related to its passband ripple and cutoff frequency.

We then look in detail at the SICNN response to both step edges and constant signals with some perturbation, such as a noisy DC signal. Using the impulse response, an expression is derived for the output-input noise variance ratio, which is compared to the experimental ratio for various SICNN parameters, such as decay factor, neighbourhood size, sum of weights, as well as signals of varying SNR and mean input intensity.

Finally we compute the noise figure (NF) of the SICNN, which provides a measure of degradation of any input signal. We compare the theoretical NF with the experimental one for various activation functions, and illustrate how this function can be chosen to obtain a small noise figure in the SICNN output.

4.2 The Recurrent and Feedforward SICNNs

In this section we define the recurrent and feedforward SICNNs, of which the latter will form the basis of our edge detector, and whose properties shall be analysed in the subsequent sections of this Chapter. We begin by showing that the steady-state of the recurrent SICNN can be solved using an iterative-type solution, where if the initial "solution" is chosen appropriately, then an architecture similar to that of the feedforward SICNN is implemented. We then show the recurrent SICNN response to constant and step edge inputs. Finally, we define the feedforward SICNN and show how it performs shunting inhibition on step edges.

4.2.1 The Recurrent SICNN

We previously discussed the derivation and stability of the Shunting Inhibitory Cellular Neural Network in Section 3.4. We showed that the differential equation in 1-D for the state of each cell is:

$$\frac{dx_i}{dt} = I_i - a_i x_i - \sum_{j \in N_r(i)} w_{ij} f(x_j) x_i \qquad i = 1, 2, \ldots, M \qquad \text{EQ (4.1)}$$

where x_i is the state of cell i, I_i is its input, a_i is its passive decay factor of excitation, f is the activation function, w_{ik} is the interaction weight between cells j and i, N_r is the neighbourhood function, and M is the total number of nodes which corresponds to the total length of the input. Such a network is *recurrent,* as each x_i depends upon the value of the neighbouring x_j, which in turn depend upon that cell's state itself. The network can be represented pictorially as in Figure 4.1, where for simplicity only three nodes are shown with each node interacting directly with its two nearest neighbours.

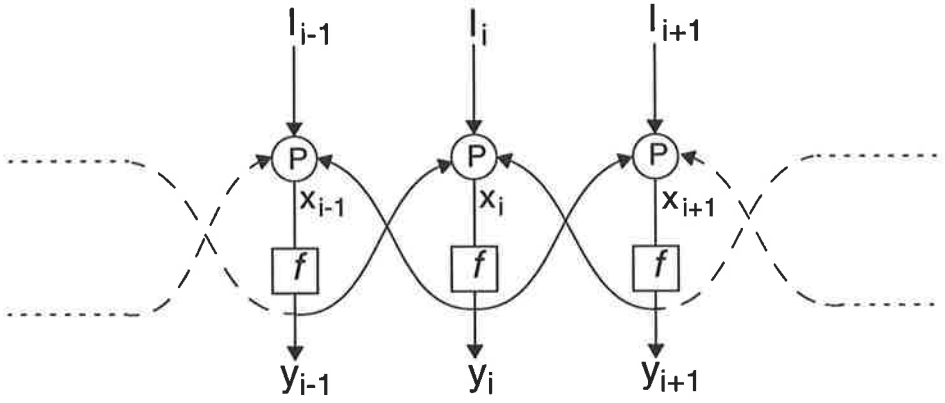


Figure 4.1 The recurrent SICNN architecture. Only three nodes and nearest neighbour interactions are shown for simplicity. *I* is the input, *x* is the cell's state, and *y* is the output. P is a processing node implementing EQ (4.1).

The steady-state solution of EQ (4.1) to a time invariant input I_i satisfies

$$x_i = \frac{I_i}{a_i + \sum_{j \in N_r(i)} w_{ij} f(x_j)}.$$

Converting the neighbourhood function into the distance of separation between cells, or input units, we can write the previous equation as

$$x_i = \frac{I_i}{a_i + \sum_{j=-r}^{r} w_j f(x_{i+j})} \qquad i = 1, 2, \ldots, M \qquad \text{EQ (4.2)}$$

where r is the neighbourhood size which represents the range of cells that can have a direct inhibitory effect on cell i. Thus, the inhibitory effects of neighbouring cells on cell i is a weighted linear combination of their respective outputs. We should recall that, in general, there is no feedback from any cell to itself, hence $w_i = 0$ for all i.

4.2.1.1 Steady-State Response

The steady-state value of each cell of the recurrent SICNN can be obtained by numerically solving the system of differential equations given in EQ (4.1), which is normally computationally intensive. Alternatively, we can define an equivalent discrete-time dynamical system that has a steady-state solution equal to the steady-state solution of EQ (4.2). For the recurrent SICNN such a discrete-time dynamical system is described by

$$x_i(k+1) = \frac{I_i}{a_i + \sum_{j=-r}^{r} w_j f(x_{i+j}(k))} \qquad k = 0, 1, 2, \ldots \infty \qquad \text{EQ (4.3)}$$

where k is the *discrete-time step*, or iteration number. The sequence is solved iteratively, i.e., given an initial estimate of the steady-state solution $x_i(0)$, we use EQ (4.3) to derive $x_i(1)$, which in turn is used to obtain $x_i(2)$ by the same process, and so-on. A simple initial value to choose is $x_i(0) = I_i$, for all i. That is, the input to each node also serves as the initial value of the steady-state value for that node, provided the SICNN converges. We now discuss the convergence of such a network defined by the above discrete equation.

4.2.1.2 Convergence

It was showed by Bouzerdoum & Pinter (1993), and noted in Section 3.4.2, that under certain conditions the continuous SICNN converges to an equilibrium point (from a possible set of many). If the discrete SICNN given by EQ (4.3) converges, then it will converge to an equilibrium point of the continuous SICNN. Due to the nonlinear nature of EQ (4.3), the general conditions required for convergence are unknown; it can be shown for special cases, such as when the weight distribution is asymmetric, as we now proceed to do.

Consider an asymmetrical weight distribution such as $w = [1\ 0\ 0]$. If we consider the left-most node of the network, then we note that the output after 1 iteration $x_1(1)$ only depends upon the input intensity and the decay factor of that node, since there is no node to the left of this one, i.e., $x_0(1)$ does not exist. As both a_1 and I_1 are constant, then so is $x_1(1)$. Looking at the next node, we see that $x_2(1)$ depends upon $x_1(0)$, a_2 and I_2, all of which are constant, so $x_2(1)$ is also constant. Thus, by induction, the entire output $x_i(1)$ is constant, i.e., it converges. This same results holds for asymmetrical weights of any size, and even in the reverse order such as $w = [0\ 0\ 1]$ (since we can repeat the above argument for the right-most node rather than the left-most one). Thus, a SICNN with any asymmetrical weight distribution is convergent.

An Example

Consider the 1-D example where all inputs to the SICNN are constant and equal to I_0, and the decay factor of excitation is $a_i = 1$ for all nodes. For the sake of simplicity we choose

an $r = 1$ neighbourhood, with $w_{-1} = w_1 = 1$ and linear activation function $f(x) = x$. Ignoring any boundary effects, let

$$I_i = x_i(0) = 5 \quad \forall i$$

where $x_i(0)$ is the initial estimate of the steady-state value. Better approximations to the steady-state solution are provided by continually iterating the network as we discussed above. For instance the first two estimates are

$$x_i(1) = \frac{I_i}{a_i + 2wx_i(0)} = \frac{5}{11}, \quad = \quad \text{and} \quad = \quad x_i(2) = \frac{I_i}{a_i + 2wx_i(1)} = \frac{55}{21}.$$

By repeating the process the sequence eventually converges to a value given by

$$x_i^0 = \frac{I_0}{a_i + \sum_{j=-1}^{1} w_j f(x_{i+j}^0)}$$

where x_i^0 is the steady-state value of node i. Figure 4.2 shows the value of $x_i(k)$ for the network described above for an increasing number of iterations, with a steady-state value of $x_i^0 = 1.35$ after approximately 20 iterations.

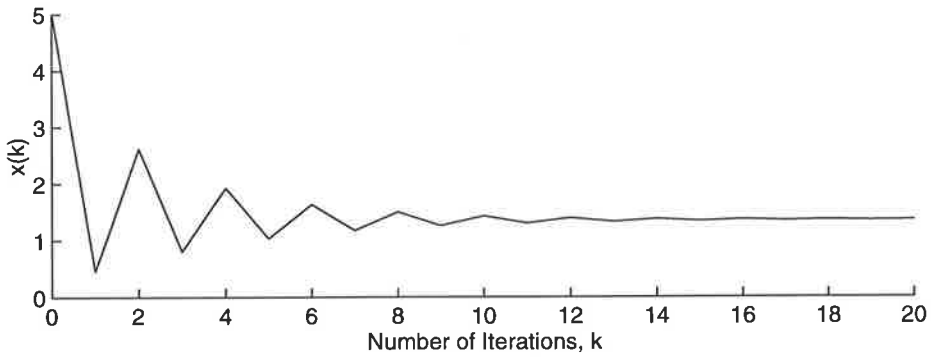


Figure 4.2 Value of $x_i(k)$ of EQ (4.3) until convergence, for an input of intensity 5.

4.2.1.3 Step Edge Response

Now consider the extension of the above discussion to a step edge input as shown in Figure 4.3(a). Some nodes or cells in the network have the lower intensity part of the edge (denoted I_L) as their input, while the remaining nodes have the upper step intensity (I_U) as their input. Consider the case of an input sequence $2M$ elements long with the step

discontinuity exactly half-way along its length, i.e., the first M nodes of the SICNN have I_L as their inputs, and the remaining M nodes have I_U as their inputs. Thus,

$$I_i = \begin{cases} I_L & \text{if } 1 \le i < M \\ I_U & \text{if } M+1 \le i \le 2M \end{cases}$$

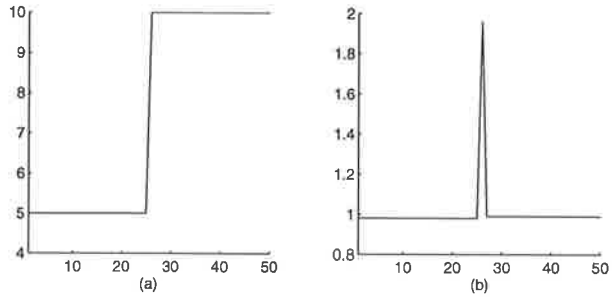


Figure 4.3 The (a) step edge input and (b) the output of the recurrent SICNN with asymmetrical weights after 1 iteration of EQ (4.3). The network parameters are given in the text.

Neglecting the effects of the sequence's boundaries and the step discontinuity itself, the output of all nodes can be computed in an analogous manner to that given above for a constant input, since the step edge is piecewise constant. That is, the output of the nodes in each of the constant parts of the step edge input can be found using the iterative method outlined above, with each node's state approaching its steady-state value as the number of iterations increases. Obviously, the interesting effects occur at the edge discontinuity. For an $r = 1$ SICNN recall from EQ (4.3) that the output at iteration k for node i is

$$x_i(k) = \frac{I_i}{a_i + w_{-1} f(x_{i-1}(k-1)) + w_1 f(x_{i+1}(k-1))} \qquad \text{for } i = 1, 2, \ldots, 2M$$

where all boundary effects have been ignored. As a numerical example, if we choose $I_L = 5$, $I_U = 10$, $w = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$, $a_i = 0.1 \ \forall i$, f a linear function, and $M = 25$, then we have for the edge point,

$$x_{26}(1) = \frac{I_{26}}{a_{26} + (1 \times I_{24} + 0 \times I_{27})} = \frac{10}{0.1 + 5} = 1.96.$$

Similarly, the outputs of neighbouring nodes are $x_{25}(1) = 0.980$ and $x_{27}(1) = 0.990$. Figure 4.3 shows the initial step input and the first iteration output for the SICNN with an asymmetrical weight distribution. Figure 4.4 shows the first and second iteration outputs for the SICNN with symmetrical weight distribution e.g., $w = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$. SICNNs with both

symmetrical and asymmetrical weight distributions shall be discussed in greater detail later on.

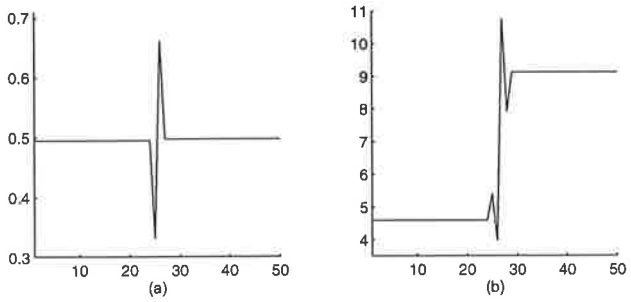


Figure 4.4 The recurrent SICNN output after (a) one iteration and (b) two iterations of EQ (4.3). The SICNN has asymmetrical weights, and the other network parameter are given in the text; the input step edge is shown in Figure 4.3(a).

Clearly the output after one iteration of the SICNN with asymmetrical weights, as shown in Figure 4.3(b), can be used for edge detection if the maximum output is located. For the output of the SICNN with symmetrical weights as shown in Figure 4.4, the first iteration output can be used for edge detection by finding where the 'zero-crossings' in the output are (once demeaning has been performed), while the second iteration output can be used for image enhancement, in particular edge enhancement.

Observing the output for different iterations, we see that for the SICNN with symmetrical weights that the first iteration, and all subsequent odd-numbered iterations, give a large response at the position of the edge, whereas the second and all subsequent even-numbered iterations give an output that is similar to the original edge but with the edge enhanced. This is characteristically similar to the *Mach band* effect, and is not totally unexpected as, after all, the SICNN is designed to perform inhibition.

4.2.2 The Feedforward SICNN

By analogy to the recurrent system given in EQ (4.2), we can also define a *feedforward system* as

$$\frac{dx_i}{dt} = I_i - a_i x_i - \sum_{j=-r}^{r} w_j f(I_{i-j})\, x_i \qquad i = 1, 2, \ldots, M \qquad \text{EQ (4.4)}$$

with the steady-state of each cell given by

$$x_i = \frac{I_i}{a_i + \sum_{j=-r}^{r} w_j f(I_{i-j})}. \qquad \text{EQ (4.5)}$$

This is also equivalent to the output of the discrete-time solution of the recurrent SICNN after 1 iteration of its steady-state solution, or EQ (4.2) assuming the initial state is the input intensity. This *feedforward* architecture is shown in Figure 4.5. The input to each node depends upon the *inputs* of adjacent nodes rather than their *outputs*, as is the case with the recurrent SICNN. So, the inputs to each node of the feedforward SICNN is the input signal, which is known; the state of each cell can be easily derived in terms of the input signal. Compare this to the recurrent SICNN, whose state equations is nonlinearly dependent upon the outputs of neighbouring cells, so the equations are much more difficult to analyse. As we shall see, the computational step in calculating the feedforward SICNN output is essentially one convolution and a small number of matrix operations.

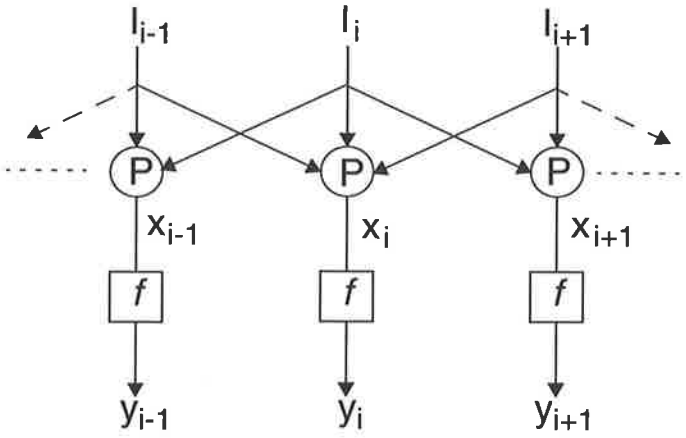


Figure 4.5 The feedforward SICNN architecture corresponding to EQ (4.5). The node P represents a processing unit implementing EQ (4.5).

The outputs of this feedforward SICNN to edge inputs are identical to the corresponding recurrent SICNN output after 1 iteration, as given in Figure 4.3(b) and Figure 4.4(a) for asymmetrical and symmetrical weights, respectively.

4.2.2.1 Implementation

If we choose the weighting or weight distribution to be *space-invariant*, that is the weighting on the output of neighbouring inhibitory nodes is dependent only upon their relative position and not their absolute position with respect to a node of interest, then the

weights can be represented by a vector w, often referred to as a *template* or *mask*. The interaction between all cells is then achieved by the convolution of the weight template with the vector $f(I)$ where f is the activation function and I is the input intensity vector.

If Y is the output vector, X a vector of state values, I a vector of inputs, w the weight template, and A the vector of decay factors, then X_i, the i^{th} element of X, is given by

$$X_i = \frac{I_i}{A_i + [w * f(I)]_i} \qquad \text{EQ (4.6)}$$

$$Y_i = f(X_i) \qquad \text{EQ (4.7)}$$

where $*$ denotes 1-D convolution, and $f(I)$ is a vector with $[f(I)]_i = f(I_i)$. Thus, the output of the feedforward SICNN is obtained by using a few simple vector operations. These operations generally are not very computationally intensive, and can be implemented rapidly in either hardware or software.

4.2.2.2 Step Edge Response

The feedforward SICNN output is identical to Figure 4.3(a), and shunting inhibition on this edge can be easily explained by considering a step edge input to an $r = 1$ neighbourhood SICNN, for example. The state of each cell or node is inhibited by the inputs of both its nearest neighbours. The feedforward SICNN output is given by:

$$x_i = \frac{I_i}{a_i + [w_{-1} f(I_{i-1}) + w_1 f(I_{i+1})]} \qquad \text{EQ (4.8)}$$

where the term $[w_{-1} f(I_{i-1}) + w_1 f(I_{i+1})]$ represents the inhibitory effect from neighbouring inputs. The output of each node also depends upon its input intensity and its decay factor.

Asymmetrical Weight Distribution

Consider first a feedforward SICNN with asymmetrical weights of $[0\ 0\ 1]$. Inhibition is effected by only those nodes to the immediate left of any given node. Figure 4.6 illustrates the effects of inhibition in different regions of a step edge. Figure 4.6(a) shows the inhibitory effects away from the discontinuity. For the node to the left of the discontinuity, both the input intensity and the inhibition signal are weak, hence the total output tends to be small. For nodes to the right of the discontinuity, both the input intensity and the inhibition signals are large, but from EQ (4.8) the total output again tends to be small.

Figure 4.6(b) shows what happens at the edge point. Although the inhibition is weak, the intensity of the input is large, hence the output is large. This results in the peak in (c), the overall output of the SICNN.

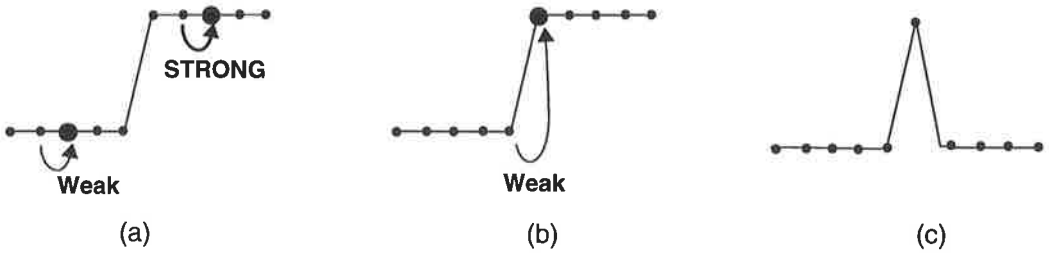


Figure 4.6 The inhibitory effects on a step edge input for a SICNN with an asymmetric weight distribution of [0 0 1]. The inhibitory effects of some nodes are indicated and (c) is the SICNN output.

Symmetrical Weight Distribution

Figure 4.7 shows the output for a SICNN with symmetrical weights of $\begin{bmatrix}1 & 0 & 1\end{bmatrix}$. The effect on an input step edge, away from the actual edge is shown in (a). Consider first the input to the left of the discontinuity. On this part of the edge the inhibition on each node is small, as is the intensity, hence x_i in general will be small. Figure 4.7(a) also shows the effect on the input to the right of the edge and away from the discontinuity. Both the inhibitory effects and intensity are now larger, so the response here also tends to be small, but may be somewhat larger than the response to the left of the discontinuity.

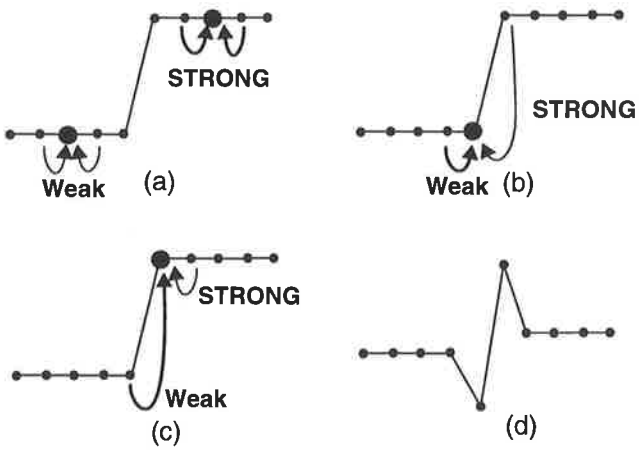


Figure 4.7 The inhibitory effects on a step edge for a SICNN with symmetrical weightsm while (d) is the overall output.

Now consider the case shown in (b), where the node has a small amount of inhibition from the node to its left (to the left of the edge), but a large inhibitory effect from the adjacent node to the right on the edge. As the intensity of the pixel itself is still small, the net effect of the increased inhibition is to reduce the output compared to those nodes further to the left of the edge. In (c) the node again has both a large and a small inhibitory influence from its neighbouring cells, but the value of input intensity is now large, hence the output suddenly increases greatly compared to that of (b). The overall output is shown in (d).

Like the case of the SICNN with asymmetrical weights, this is a gross simplification since the amount of inhibition also depends on other network parameters (see EQ (4.8)). The edge is located at the zero-crossing of this output once the local mean has been removed.

Clearly from the three previous figures the SICNN can perform edge detection. More precisely the SICNN is an edge enhancer, but we shall call it an edge detector hereafter whenever used for this purpose. Thus, for a step-edge the feedforward SICNN can be used for both edge detection and enhancement. The output, in the edge detection case, can be thresholded, or the zero-crossings located, to detect the position of edges. Also note from EQ (4.5), Figure 4.6 and Figure 4.7 that the number of output pixels affected by the step edge is r pixels to the left of the discontinuity, and r pixels to the right of it (or $r-1$ if we exclude the edge pixel itself).

4.2.2.3 Advantages

The SICNN has many tunable parameters, such as the weight distribution, decay factor of excitation, activation function, and neighbourhood size. This may give us greater flexibility in adapting the network to a particular type of input edge and/or noise, hence achieving better results than most linear filters that cannot adapt or have very few tuneable parameters, such as a gradient operator. Another advantage over some edge detectors, such as the neural network edge detectors discussed in Section 2.5.5, is the speed at which the output of the feedforward SICNN can be computed. It essentially requires one vector convolution, and one vector addition and division, all of which can be implemented rapidly.

A drawback of the SICNN is that its output intensity varies nonlinearly for different step edge inputs and SICNN parameters. Thus, its edge detection performance will also vary for different SICNN parameters and different inputs. We cannot test the effect of every

possible combination of these parameters and inputs on the performance since there are infinitely many. We can, however, thoroughly test and observe the effects of each individual parameter in isolation, though there will always remain the small possibility of overlooking a potentially good combination of parameters. Fortunately, this will become less of an issue as we understand the role of each parameter on the edge detection performance.

4.3 Impulse & Frequency Response

Having described the feedforward SICNN and discussed its application to edge detection, we are now at a position to derive its impulse and frequency responses by linearising the feedforward SCINN using perturbation analysis.

4.3.1 Impulse Response of the Linearised, Feedforward SICNN

Consider a 1-D feedforward SICNN. Let the average intensity of the input be I_0, with a deviation at the ith node or cell of the SICNN equal to υ_i. Thus, the total input to this cell is $I_0 + \upsilon_i$. Clearly any input signal can be decomposed into these two components: a constant signal and a signal of the derivation from this. From EQ (4.5) the output at node i due to the constant input component I_0 is

$$X_{i,0} = \frac{I_0}{a_i + f(I_0)\sum_{j=-r}^{r} w_j}. \qquad \text{EQ (4.9)}$$

Let the perturbation from the constant component of the ith output of the SICNN be x_i. Inserting these values into the dynamical equation of EQ (4.4), and with all derivatives equal to zero at steady-state, gives

$$0 = I_0 + \upsilon_i - a_i(X_{i,0} + x_i) - (X_{i,0} + x_i)\sum_{j=-r}^{r} w_j f(I_0 + \upsilon_{i+j}). \qquad \text{EQ (4.10)}$$

Using the fact that, from EQ (4.9) it follows that I_0 and $X_{i,0}$ satisfy

$$0 = \dot{X}_{i,0} = I_0 - a_i X_{i,0} - X_{i,0}\sum_{j=-r}^{r} w_j f(I_0) \qquad \text{EQ (4.11)}$$

and the Taylor expansion is

$$f(I_0 + \upsilon_{i+j}) = f(I_0) + f'(I_0)\,\upsilon_{i+j} + \text{h.o.t.}, \qquad \text{EQ (4.12)}$$

where "h.o.t" denotes *higher order terms*, then applying these to EQ (4.10) gives

$$0 = \upsilon_i - a_i x_i - X_{i,0} \sum_{j=-r}^{r} w_j f'(I_0)\,\upsilon_{i+j} - x_i \sum_{j=-r}^{r} w_j [f(I_0) + f'(I_0)\,\upsilon_{i+j}] + \text{h.o.t.}.$$

Ignoring the h.o.t., we obtain

$$0 = \dot{x}_i = \upsilon_i - a_i x_i - X_{i,0} \sum_{j=-r}^{r} w_j f'(I_0)\,\upsilon_{i+j} - x_i \sum_{j=-r}^{r} w_j [f(I_0) + f'(I_0)\,\upsilon_{i+j}]$$

$$\Rightarrow x_i = \frac{\upsilon_i - X_{i,0} \sum_{j=-r}^{r} w_j f'(I_0)\,\upsilon_{i+j}}{a_i + \sum_{j=-r}^{r} w_j f(I_0)}.$$

The impulse response of the linearised, feedforward SICNN at the i^{th} node is given by

$$h_i(n) = \frac{\delta(n) - X_{i,0} \sum_{j=-r}^{r} w_j f'(I_0)\,\delta(n+j)}{a_i + \sum_{j=-r}^{r} w_j f(I_0)}, \; n = 1, 2, \ldots, M \qquad \text{EQ (4.13)}$$

where M is the total number of nodes in the SICNN, and $\delta(n)$ is the (impulse) unit-sample sequence. Note that the response is clearly dependent upon both the SICNN parameters and the input mean intensity. Figure 4.8 shows the impulse responses for both the symmetrical and asymmetrical rectangular weight distributions. The overall length of the impulse response is $2r + 1$, since this is the size of the weight distribution. Beyond this, the weight distribution is zero, hence the impulse response is also zero.

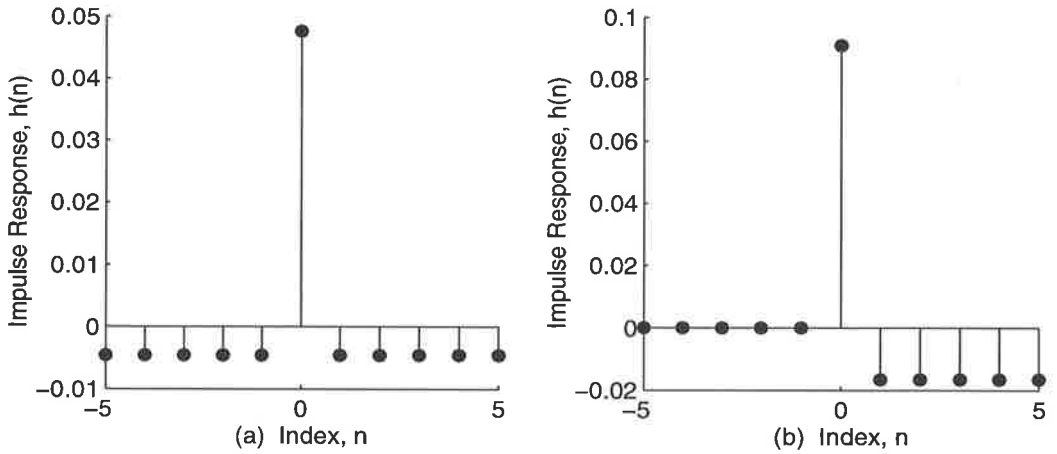


Figure 4.8 Two examples of the linearised, feedforward SICNN impulse responses $h_i(n)$ given by EQ (4.13) for the (a) symmetrical and (b) asymmetrical, rectangular weights, with $r = 5$, $a = 1$, and $I_o = 10$.

4.3.2 Frequency Response

Taking the discrete-time Fourier transform of EQ (4.13) gives the frequency response of the linearised SICNN as

$$H_i(\omega) = \sum_{n=-r}^{r} h_i(n) \exp(-jn\omega) \quad 0 \le \omega \le \pi.$$

This can be written as

$$H_i(\omega) = h_i(0) + \sum_{n=1}^{r} [h_i(n) \exp(-jn\omega) + h_i(-n) \exp(jn\omega)]$$

$$= \frac{1}{a_i + f(I_0) \sum_{j=-r}^{r} w_j} - \frac{X_{i,0} f'(I_0)}{a_i + f(I_0) \sum_{j=-r}^{r} w_j} \sum_{n=1}^{r} [w_{-n} \exp(jn\omega) + w_n \exp(-jn\omega)]$$

assuming $w_0 = 0$. This frequency response can be written as

$$H_i(\omega) = \alpha_i \left(1 - X_{i,0} f'(I_0) \sum_{n=1}^{r} [w_{-n} \exp(jn\omega) + w_n \exp(-jn\omega)] \right) \quad \text{EQ (4.14)}$$

where $\alpha_i = \dfrac{1}{a_i + f(I_0) \sum_{n=-r}^{r} w_n}$

4.3.2.1 Symmetric and Asymmetrical Weight Distribution Cases

Asymmetrical Weight Distribution Case

If we consider the linearised, feedforward SICNN with asymmetrical weights, that is $w_n = 0$ for $n \leq 0$, then the frequency response of EQ (4.14) becomes

$$H_i(\omega) = \alpha_i \left(1 - X_{i,0} \sum_{n=1}^{r} w_n \exp(-jn\omega) \right). \qquad \text{EQ (4.15)}$$

Figure 4.9 shows the magnitude response $|H_i(\omega)|$ for mean intensities $I_0 = 1, 5,$ and 10, with asymmetrical, rectangular weights, and a decay factor of $a_i = 1$. These responses are all similar in shape, though their magnitude values are different. Note the highpass nature of this frequency response, which is what we expect for an edge detector and enhancer.

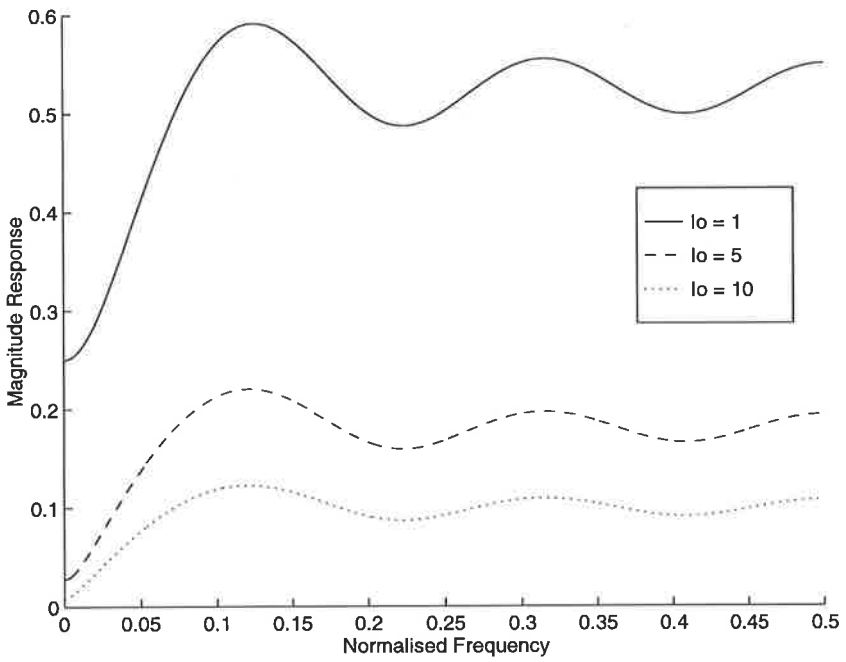


Figure 4.9 The magnitude response of $H_i(\omega)$ given by EQ (4.15) for asymmetrical, rectangular weights, decay factor 1, and three different mean input intensities.

Symmetrical Weight Distribution Case

If we now consider the linearised, feedforward SICNN with symmetrical weights, that is $w_n = w_{-n}$ for all valid n, then from EQ (4.14), we obtain

$$H_i(\omega) = \alpha_i\left(1 - X_{i,0}\sum_{n=1}^{r} w_n\left[\exp(jn\omega) + \exp(-jn\omega)\right]\right)$$
$$= \alpha_i\left(1 - X_{i,0}\sum_{n=1}^{r} w_n \cos n\omega\right) = \sum_{n=0}^{r} c_n \cos n\omega$$

The magnitude of this frequency response is, interestingly, very similar to those shown in Figure 4.9; once again it is highpass in nature.

4.3.2.2 DC Gain

It is clear from Figure 4.9 that the DC gain, defined as $|H_i(0)|$, varies for different input mean intensities. We will now explore this relationship in a little more detail. From EQ (4.14), the DC gain is

$$H_i(0) = \alpha_i\left(1 - X_{i,0}\sum_{n=-r}^{r} w_n\right) = \frac{1}{a_i + f(I_0)\sum_{n=-r}^{r} w_n} \times \left(1 - \frac{I_0\sum_{n=-r}^{r} w_n}{a_i + f(I_0)\sum_{n=-r}^{r} w_n}\right).$$

Figure 4.10 shows the DC gain as the mean intensity is varied. From the above equation, assuming a linear activation function, and for large I_0 the DC gain is roughly inversely proportional to the mean intensity I_0. This is confirmed in the plot below where we can see the gain decreasing with I_0. Interestingly, identical results are obtained for both symmetrical and asymmetrical weight distributions, provided the total sum of the weights is the same.

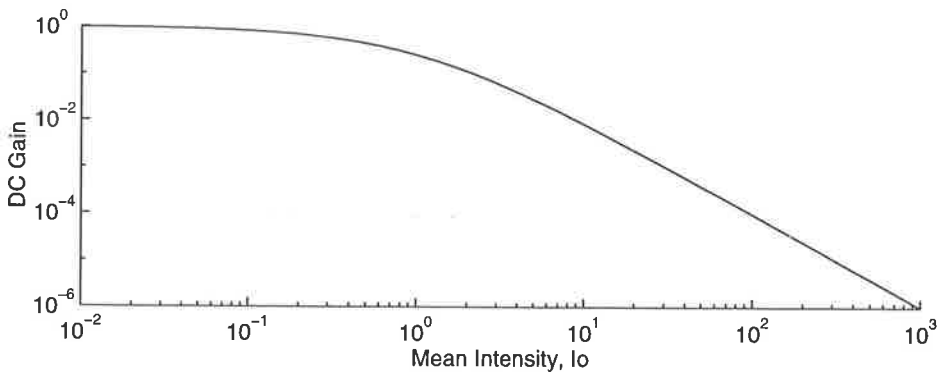


Figure 4.10 DC gain of the feedforward SICNN magnitude response as the mean intensity varies. The SICNN symmetrical, rectangular weights, $a = 1$, and $r = 5$. The result is also identical for asymmetrical, rectangular weights.

4.3.2.3 Passband Ripple and Cutoff Frequency

We have already derived the feedforward, linearised impulse response for the i^{th} cell of the network to be

$$h_i(n) = \frac{\delta(n) - X_{i,0}\sum_j w_j f'(I_0)\,\delta(n+j)}{a_i + \sum_j w_j f(I_0)}.$$

A typical magnitude of its Fourier transform is shown in Figure 4.11. Observe from this plot that the SICNN is essentially a highpass filter, which is required for edge detection.

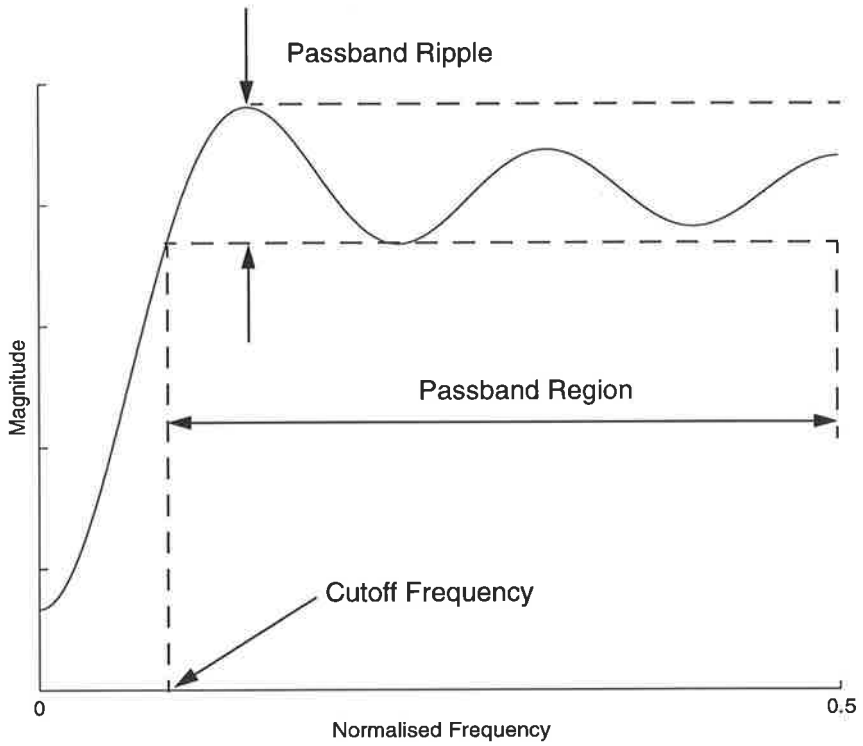


Figure 4.11 Linearised, feedforward SICNN impulse magnitude response with the passband region ripple, and the cutoff frequency indicated.

From this plot we can define two quantities:

- the *Passband Ripple*, which is the difference in magnitude between the largest maximum and smallest minimum in the passband region of the magnitude response, and
- the *Cutoff Frequency*, which defines where the passband region commences. It corresponds to the smallest frequency where the magnitude response equals the magnitude of the smallest minimum in the passband region.

The cutoff frequency determines the bandwidth of the SICNN. We shall show later how this affects the edge detection performance. The passband ripple obviously measures the amount of ripple in the magnitude response.

4.4 SICNN Response to Random Inputs

Having developed the linearised feedforward SICNN impulse response, we can now investigate its response to random signals (noise). We begin with deriving the variance of the SICNN output in response to a random input, using the impulse response derived above, and then experimentally verify the validity of this derivation for a range of network parameters using noisy input signals.

4.4.1 Variance of the Random Output

Previously, we derived the linearised, feedforward SICNN impulse response to be

$$h_i(n) = \frac{\delta_n - X_o f'(I_0) \sum_{j=-r}^{r} w_j \delta_{n+j}}{a_i + f(I_0) \sum_{j=-r}^{r} w_j}$$

where δ_n is the unit sample sequence. Assume now that the constant input signal is corrupted with noise denoted by υ_i. We can compute the noise variance (or energy) of the i^{th} output of the SICNN using the relation

$$\sigma^2_{i,\,out} = E\left[(\upsilon_i * h_i(n))^2 \right] = E\left[\upsilon_i^2\right] \sum_{j=-r}^{r} h_i^2(j)$$

where E is the expectation operator. The above formula can be easily proved by starting with the left-hand side of the equation:

$$E\left[(\upsilon_n * h_i(n))^2 \right] = E\left[\left(\sum_{k=-\infty}^{\infty} h_i(n)\, \upsilon_{n-k} \right)^2 \right].$$

Assuming the noise υ_n is i.i.d. (*independent, identically distributed*) then the noise is statistically independent in the spatial sense, i.e.,

$$E[\upsilon_n \upsilon_m] = 0 \qquad \text{for } n \neq m .$$

Using this we obtain

$$E\left[\left(\sum_{k=-\infty}^{\infty} h_i(k)\,\upsilon_{n-k}\right)^2\right] = \sum_{k=-\infty}^{\infty} h_i^2(k) \cdot E\left[\upsilon_{n-k}^2\right]$$
$$= E[\upsilon_n^2] \sum_{k=-r}^{r} h_i^2(k)$$

since $E\left[\upsilon_{n-k}^2\right] = E\left[\upsilon_n^2\right]$ when υ_n is identically distributed. Thus the *output-input variance ratio* is

$$\frac{\sigma_{i,out}^2}{\sigma_{i,in}^2} = \sum_{j=-r}^{r} h_i^2(n) = \frac{1 + (X_{i,0} f'(I_0))^2 \sum_{j=-r}^{r} w_j^2}{\left(a_i + f(I_o) \sum_{j=-r}^{r} w_j\right)^2}. \qquad \text{EQ (4.16)}$$

Recalling that $X_{i,0} = I_o \Big/ \left(a_i + f(I_o) \sum_{j=-r}^{r} w_j\right)$ then

$$\frac{\sigma_{i,out}^2}{\sigma_{i,in}^2} = \frac{\left(a_i + f(I_o) \sum_{j=-r}^{r} w_j\right)^2 + (I_o f'(I_0))^2 \sum_{j=-r}^{r} w_j^2}{\left(a_i + f(I_o) \sum_{j=-r}^{r} w_j\right)^4}, \qquad \text{EQ (4.17)}$$

An interesting point to note about this equation is that the variance ratio depends in a nonlinear way on a_i, the mean intensity I_0, and the interaction weights $\sum w_j$. Note that these equations still apply for a noisy signal of non-zero mean, since the noise can be de-meaned and its mean added to the mean intensity I_0. The noise variance can then be computed using the above equations.

4.4.1.1 Experimental Validation

In all the following experiments, the theoretical values of the output noise variance were computed using EQ (4.17), while the experimental ones were obtained by computing the input signal's noise statistics and then computing the corresponding noise statistics of the SICNN output.

Signal to Noise Ratio (SNR)

Before we discuss any results, the signal-to-noise ratio (SNR) needs to be defined. Consider a constant (DC) signal with superimposed additive noise. The SNR is defined as

$$SNR = 20\log\left(\frac{I_0}{\sigma_{in}}\right)$$

where I_0 is the mean input intensity, and σ_{in}^2 is the input noise variance averaged over all nodes.

Results for varying SNR

We now investigate the validity of EQ (4.17) for the SICNN output-input noise variance ratio as the input SNR is varied for two different neighbourhood sizes of $r = 5$ and *10*. In all cases a linear activation function is used, with symmetrical, rectangular weights, and a decay factor of 1. The mean input intensity is 10, and the noise is additive white Gaussian.

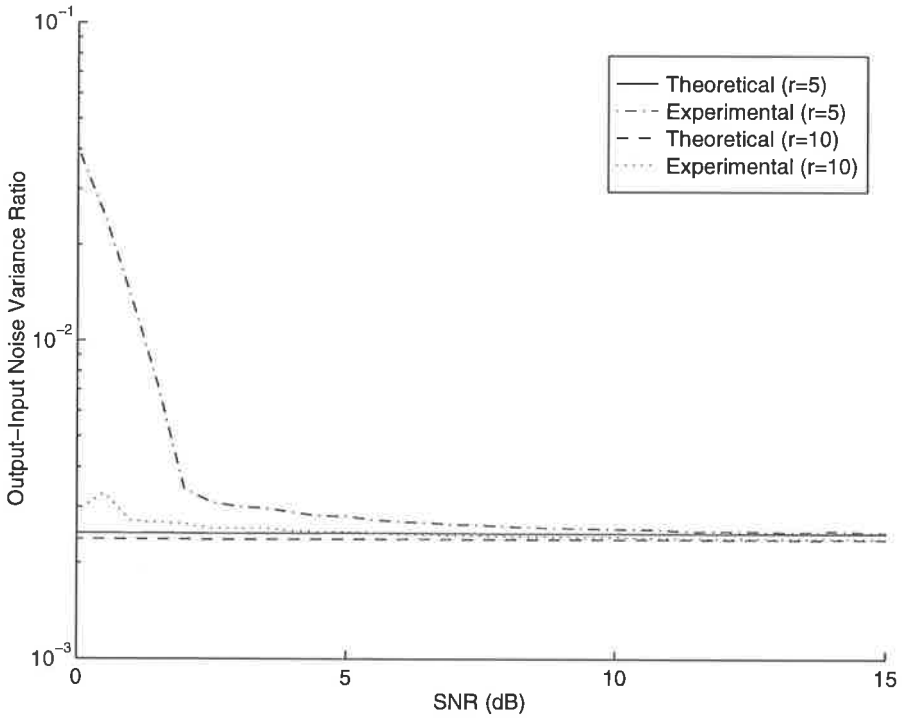


Figure 4.12 Comparison of the theoretical and experimental output-input noise variance ratio for the SICNN as a function of the SNR. The SICNN parameters are symmetrical, rectangular weights with $a = 1$, $I_o = 10$ and additive white Gaussian noise.

As expected, it is clear from Figure 4.12 that the theoretically derived noise variance ratio values are very close to the experimental ones, particularly at large SNR values since the errors from any approximations made, obviously become more insignificant. Overall, the differences between the experimental and theoretical ratios are also smaller for SICNNs

with larger neighbourhood sizes, especially for small SNR values. The likely reason for this will be explained in the next subsection.

Results for Varying Neighbourhood Size

We have just observed that the differences between the theoretical and experimental noise variance ratios appear to diminish as the neighbourhood size increases. To test this, the output-input noise variance ratio is measured as the neighbourhood size, r, is varied between 1 and 15, which can be considered as both "small" and "large", respectively. Again consider a SICNN with symmetrical, rectangular weights, and a decay factor of 1, operating on an input signal of mean intensity 10, and SNR of 5 dB with additive white Gaussian noise.

The results are shown below in Figure 4.13(a). Clearly the difference between the theoretical and experimental noise variance ratios decreases as the neighbourhood size increases; in fact it appears that the difference asymptotically approaches zero as r increases. The reason for this is the following: recall that the smallest order term of the neglected terms of the Taylor series EQ (4.12), in our derivation of the linearised impulse response was

$$x_i \sum_{j=-r}^{r} w_j \upsilon_{i+j} \qquad \text{EQ (4.18)}$$

where υ is the input noise sequence. Thus, we have a weighted sum, or average, of the input noise. Clearly, the greater the number of terms that we add together, i.e., the greater the value of r, then the greater the likelihood that the noise will tend to average itself out. Hence its effect on the output noise statistics diminishes, and our approximations become better. The weight vector w_j can be thought of as a smoothing filter: the greater its support, the smaller the output noise variance.

We also note that the noise variance ratio decreases slightly as the neighbourhood size increases, which can be explained from EQ (4.17). If the SICNN has constant decay factor and sum of weights, then the output-input noise variance ratio is only dependent on the sum of squares of the weight distribution. We have used a rectangular weight distribution here, so by increasing the neighbourhood size, the sum of squares of the weights decreases, which results in a decrease in the noise variance ratio.

Result with Varying Sum of Weights

EQ (4.18) above, which is the most significant term discarded from our Taylor series expansion, also appears to decrease as we make the value of the weights w_j smaller. Thus, we would expect the theoretical noise variance ratio to be much closer to the experimental one the smaller we make the absolute value of the weights.

Consider an $r = 5$ SICNN with a symmetrical, rectangular weight distribution of varying sum, and a decay factor of 1. The input signal has a mean intensity of 10, and a SNR of 5 dB. The comparison of the theoretical and experimental noise variance ratios is shown in Figure 4.13(b). Although it is the sum of the weights that is being varied in this plot, as we use a rectangular weight distribution then each individual weight w_j will also vary proportionally. From this Figure we can see that a large sum of weights, i.e. large w_j, causes there to be a difference between the theoretical and the experimental noise variance ratios, although this is still very small. This simply confirms what we stated above: that smaller weights give more accurate results, or negate the effect of noise to some extent.

We also note that the output-input noise variance ratio decreases as the sum of weights increases. From EQ (4.17), and ignoring the term involving the sum of squared weights, then for very large sum of weights the output-input noise variance ratio is inversely proportional to the sum of the weights. Thus, if the decay factor is kept constant, then increasing the sum of weights clearly decreases the output-input noise variance ratio.

Results with Varying Decay Factor of Excitation

Now we consider the comparison of the theoretical and experimental noise variance ratios with varying decay factor of excitation a_i. From EQ (4.10), the SICNN dynamical equation, at steady-state, can be rewritten as

$$0 = I_0 + \upsilon_i - (X_{i,0} + x_i)\left(a_i + \sum_{j=-r}^{r} w_j f(I_0 + \upsilon_{i+j})\right).$$

To obtain the impulse response of the SICNN, and hence its noise variance ratio, we discarded the higher order terms of the Taylor expansion of $f(I_0 + \upsilon_{i+j})$. It is these higher order terms which cause the difference between the theoretical noise variance ratio and the experimental one. Thus, the larger the decay factor is, the smaller the effect of the discarded terms of $\sum_{j=-r}^{r} w_j f(I_0 + \upsilon_{i+j})$, hence the closer the match between the theoretical and experimental noise variance ratios. This is illustrated in Figure 4.13(c) where we

consider a SICNN with a neighbourhood size of 5, symmetrical, rectangular weights, and an input signal with mean intensity of 10 and a SNR of 5 dB. Clearly, the error between the theoretical and experimental values is large for smaller decay factors, since the discarded terms become more insignificant.

As with increasing the sum of weights, increasing the decay factor decreases the output-input noise variance ratio. From EQ (4.17), if the weight distribution is kept constant, then for large decay factor, the output-input noise variance ratio is inversely proportional to the square of the decay factor, hence as the decay factor increases, so does the output-input noise variance ratio.

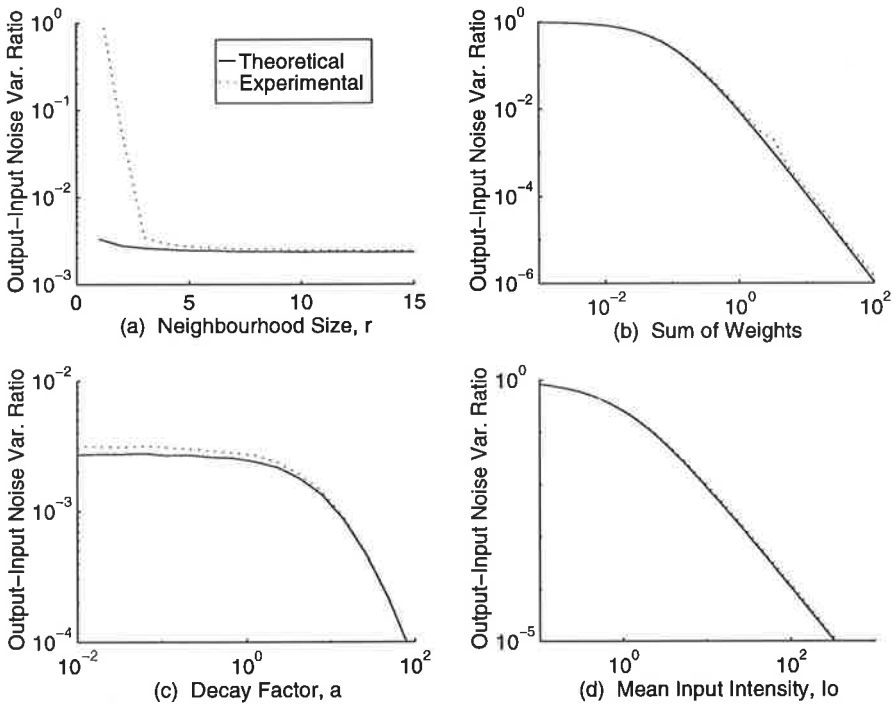


Figure 4.13 SICNN theoretical and experimental output-input noise variance ratio for varying (a) neighbourhood size, (b) sum of weights, (c) decay factor, and (d) mean input intensity. The input has additive Gaussian noise with SNR of 5 dB.

Results with Varying Mean Input Intensity

Finally, we consider the comparison between the theoretical and experimental output-input noise variance ratio as the mean input intensity I_0 is varied. Figure 4.13(d) shows the results for a SICNN with a neighbourhood size of $r = 5$, linear activation function, symmetrical rectangular weight distribution summing to 1, a decay factor of 1, and an input SNR of 5 dB. Overall, the match between the theoretical and experimental noise variance ratio is very close for a large range of I_0. From EQ (4.17) we can see that for

very large input intensities the noise variance ratio is approximately inversely proportional to the square of the mean input intensity, and this is clearly evident in (d). Compare this to linear filters, who for a given input SNR have an output noise variance ratio independent of the mean input intensity. This illustrates the capability of the SICNN to nonlinearly adapt to its input signal, which we shall find useful later in this thesis for both edge detection and enhancement.

4.4.2 Noise Figure

Definition

Another useful measure in the investigation of the SICNN performance with random signals, is to measure its *noise figure* (NF) characteristics. The NF is normally applied to amplifiers in communication systems, and measures the amount of degradation of the signal caused by the receiving system. It is defined as (Adamson, 1992)

$$NF = \frac{S_{in}/N_{in}}{S_{out}/N_{out}} \qquad \text{EQ (4.19)}$$

or in decibels, $NF\,(dB) = 10\log(NF)$, where S_{in}, S_{out}, N_{in}, and N_{out} are the input and output signal strength, and input and output noise strengths, respectively. An ideal amplifier would have a NF of 1, or 0 decibels.

The SICNN with a constant input can be thought of as an amplifier - it has a noisy input signal and it amplifies both the signal component and the noise component of this input. We define the NF as

$$NF = \frac{I_0^2/\sigma_{in}^2}{I_{out}^2/\sigma_{out}^2} \qquad \text{EQ (4.20)}$$

where I_0 is the mean input component or intensity, σ_{in}^2 and σ_{out}^2 are the variances as given by EQ (4.17), and I_{out} is equal to the output mean X_0 in EQ (4.9). Using these equations, the theoretical NF of the SICNN is

$$NF = I_0^2 \times \frac{1}{I_{out}^2} \times \frac{\sigma_{out}^2}{\sigma_{in}^2} = I_0^2 \times \frac{\left(a + f(I_0)\sum_{j=-r}^{r} w_j\right)^2}{I_0^2} \times \frac{\left(a + f(I_0)\sum_{j=-r}^{r} w_j\right)^2 + (I_0 f(I_0))^2 \sum_{j=-r}^{r} w_j^2}{\left(a + f(I_0)\sum_{j=-r}^{r} w_j\right)^4}$$

where $a_i = a$ for all nodes. Simplifying this expression we obtain,

$$NF = 1 + \frac{(I_0 f'(I_0))^2 \sum_{j=-r}^{r} w_j^2}{\left(a + f(I_0) \sum_{j=-r}^{r} w_j\right)^2}. \qquad \text{EQ (4.21)}$$

If we can make the second term of EQ (4.21) small, then the NF will also be small for a constant sum of weights, though not necessarily equal to the smallest possible NF. If we choose $f(x) = x/(1+x)$ [1], then the above equation becomes

$$NF = 1 + \frac{\left(I_0 \times \frac{1}{(1+I_0)^2}\right)^2 \sum_{j=-r}^{r} w_j^2}{\left(a + \frac{I_0}{(1+I_0)} \sum_{j=-r}^{r} w_j\right)^2}. \qquad \text{EQ (4.22)}$$

We can see that for large I_0, the numerator approaches zero. We also choose to use $x/(1+x)$ to avoid the possibility of dividing by zero intensity.

Experimental Results

To determine over what range of noisy input signals EQ (4.21) is valid, the NF was determined experimentally for two different activation functions, $f(x) = x$ and $f(x) = x/(1+x)$, and then compared to the expression above. In Figure 4.14, the theoretical NF of the two activation functions are compared with the experimentally obtained NF for various SNR values as the mean input intensity I_0 is varied. To aid the visual comparison we in fact plot (NF-1), which is the second term of EQ (4.22).

Figure 4.14(a) is for the linear activation function, $f(x) = x$, and shows that the experimental NF values are the same, or very close to, the theoretical values for input signals with SNRs of 10 and 20 dB. For the input signal with 0 dB SNR, the comparison is poor, for a large range of I_0

Figure 4.14(b) shows the results for the nonlinear activation function, $f(x) = x/(1+x)$. Once again the input signal with 20 dB SNR has a NF essentially identical to the theoretical one, while the input for 10 dB SNR has a NF which is similar for small I_0, but

1. The derivative of this function has I_0 only in its denominator and it is also squared. Thus, the numerator of the 2nd term on EQ (4.21) in inversely related to I_0, hence NF is small for large I_0.

much larger for large I_0. The 0 dB SNR input signal again gives an experimental NF which is very different to the theoretical one, being more inaccurate as I_0 increases.

These results show that the theoretically derived NF is only valid, or accurate, for input signals of SNRs of at least 10 dB for the linear activation function and 20 dB for the nonlinear activation function. Nevertheless, the two plots clearly indicate that the NF can have vastly different forms, depending upon the activation function used. Both plots also indicate that the smaller the input SNR, the greater the NF, i.e., the larger the input to output SNR. Also, the NF for the nonlinear activation function is, in general, less than the NF for the linear activation function. The experimental NF, however is more inaccurate for the nonlinear activation function than the linear one. This is particularly true for low SNR inputs and large I_0, because the NF with the nonlinear function is more sensitive to noise than the NF with the linear function, since we have I_0 in the denominator.

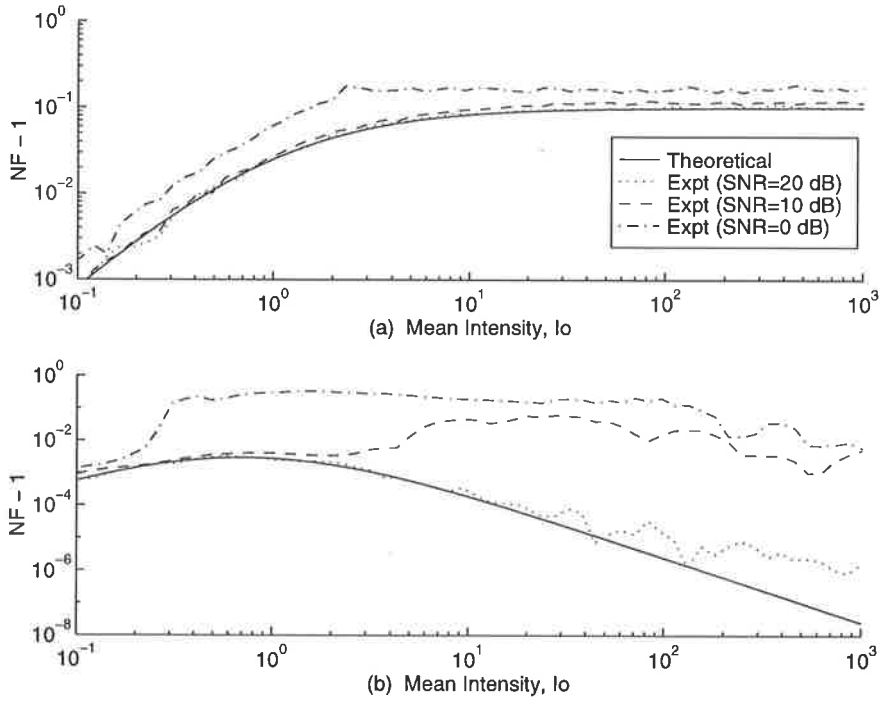


Figure 4.14 Comparison of the theoretical and experimental NF values for (a) $f(x) = x$, and (b) $f(x) = x/(1+x)$. In both cases, $r = 5$, $a = 1$, with symmetrical, rectangular weights, and additive white Gaussian noise.

4.5 Conclusions

This Chapter presented a detailed analysis of SICNN systems as well as their characteristics, providing the foundation for the work used throughout this thesis. We began by defining the recurrent SICNN, and showed how its steady-state solution is nonlinear, that is, the output of any node is dependant on the output of its neighbouring nodes. To solve such a system, an equivalent discrete-time dynamical system was defined with a steady-sate equal to that of the continuous time SICNN. From this sequence we showed that the output after one and two iterations is suitable for edge detection and enhancement, respectively.

We then defined the feedforward SICNN, and showed that its output is identical to that of the recurrent SICNN output after 1 iteration. Using this feedforward SICNN, we described and illustrated the shunting inhibitory nature of the SICNN on step edge inputs. We also discussed the thresholding techniques, the implementation and advantages of the SICNN over existing edge detectors.

Using perturbation analysis and linearisation, the impulse response of the feedforward SICNN was derived, as well as its frequency response. Both the impulse and frequency responses vary with the mean input intensity, hence indicating the ability of the SICNN to adapt to its input. Also from this frequency response we defined the passband ripple and cutoff frequency, and using the impulse response we derived the variance of its output for a noisy DC input. The derived equation shows that the SICNN is a highly nonlinear function of the network parameters, as well as the mean input intensity.

We showed that the experimental output-input noise variance ratio is similar to the theoretical one for large SNR, large neighbourhood sizes, large decay factor, small sum of weights, and for a large range of mean input intensities. When the network parameters are chosen in this way, the approximations used in the linearisation process to derive the impulse response, and thus the output-input noise variance ratio, all become more valid. The discarded terms become insignificant, hence we expect better agreement between the experimental and theoretical noise variance ratios.

Finally, we derived an expression for the SICNN NF, which measures the amount of degradation in the signal after passing through the SICNN. We compared the theoretical value of the NF to the experimental one for both the $f(x) = x$ and the nonlinear

$f(x) = x/(1+x)$ activation functions. The experimental value of the NF compared well with the theoretical ones, especially for large SNRs. We also showed that the SICNN with the nonlinear activation function gave a smaller NF than the SICNN with the linear activation function, but the NF was much more sensitive to noise on the input and the mean input intensity.

Chapter 5 SICNN Parameter Design for Edge Detection

5.1 Introduction

In Chapter 4 we introduced both the recurrent and feedforward SICNNs and presented a detailed investigation into the response properties of the linearised feedforward SICNN. In this Chapter, this analysis is extended to investigate the effect of the SICNN parameters on its edge detection performance.

We begin by defining the mean intensity and contrast of a step edge used throughout this investigation, as well as the edge to noise ratio (ENR). Then three statistical measures that are used throughout this thesis to quantify the performance of any edge detector for a 1-D step edge are defined, along with three weight distributions with well known shapes: the rectangular, triangular and Gaussian distributions.

In Section 5.3 we investigate the factors affecting the SICNN edge detection performance, namely the output noise variance, the shape of its edge response, and the ratio of the peak response to output noise variance (PNR). These three measures are all inter-related with each other. By changing the shape of the edge response, we also change its output noise variance, and both of these affect the PNR.

Using the noise variance, the shape of the edge response and the PNR, the effect of the weight distribution on its edge detection performance is investigated. We indicate how these parameters can be chosen to maximise the SICNN performance.

5.2 Definitions

Before beginning the design of the SICNN for 1-D edge detection, a means of quantifying the edge detection performance is needed so that the effect of different SICNN parameters can be observed. Although a number of measures already exist in the literature, they are either not applicable for the SICNN, or they are not intuitive to understand. In this section three statistical measures are defined which can be used with any edge detector for 1-D step edges, which are both intuitive and easy to measure. We also define variables that characterise an edge and the amount of noise present, and three weight distributions with well-known shapes.

5.2.1 Hit Rate, Edge Standard Deviation, and Edge Bias

In Section 2.4 a number of measures used in developing and comparing the performance of edge detectors were reviewed. Many of these measures are ambiguous, e.g. Pratt's figure of merit (FOM), where many possible scenarios can lead to the same FOM value. Other performance measures appear specifically tuned to a particular type of edge detector, such as Canny's measure (Canny, 1986), which does not apply to zero-crossing based edge detectors such as the LoG. For these reasons, performance measures are sought that can be applied to a wide variety of edge detectors, and without ambiguity. For a 1-D step edge, we use statistical measures based on the deviation of the detected edge position from its true position. These are actually a variation of the statistical measures described in Section 2.4.2. Thus, we define:

- *Hit rate (HR)* - a count, normalised to one, of the number of edges that have been detected in exactly the correct position, i.e., a hit-miss scenario. This is the probability of correct edge detection.

- *Edge standard deviation (ESD)* - or localisation, is the standard deviation of the spatial spread of the ensemble of the detected edges about their mean position. Canny uses this measure as one of his optimality criteria.

- *Edge bias (EB)* - the average distance of the detected edge from the true edge position.

Clearly, the greater the HR, and the smaller the ESD and EB, the better is the edge detector. Ideally, a HR of one, an ESD of zero, and an EB of zero about the true edge position are all desired.

5.2.2 Step-Edge

Consider a 1-D step edge with mean intensity I_0, with the lower and upper background intensities of the edge as I_{min} and I_{max}, respectively. The contrast of the edge is defined as

$$c = \frac{I_{max} - I_{min}}{I_{max} + I_{min}}. \qquad \text{EQ (5.1)}$$

which implies that $I_{max} = I_0(1+c)$ and $I_{min} = I_0(1-c)$. Figure 5.1 shows 1-D noiseless edges of mean intensities $I_0 = 10$ and 100, and contrasts $c = 1/4$ and $1/2$.

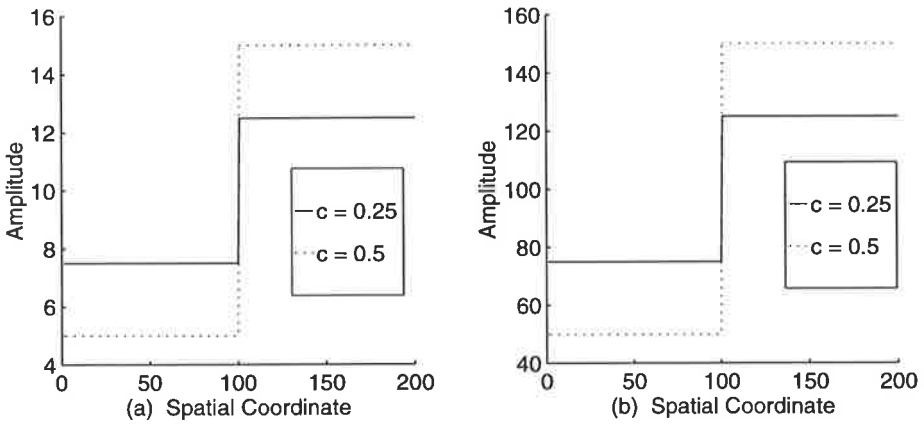


Figure 5.1 Noiseless edges of mean intensity (a) 10, and (b) 100.

For a step edge with additive noise of variance σ^2, the *Edge-to-Noise Ratio* (ENR) is defined as

$$ENR = 20\log\left(\frac{cI_0}{\sigma}\right) \text{ dB}. \qquad \text{EQ (5.2)}$$

This should not be confused with the SNR value defined previously, which applies only to constant signals corrupted with noise, and not noisy step edges.

Figure 5.2 shows a number of noisy step edges with various ENR values. Clearly the edge is virtually indistinguishable in the -10 dB case, and just discernable in the 0 dB case, while for an ENR greater than 0 dB the step edge is certainly distinguishable from the noise. Thus, we note that for an ENR less than 0 dB (particularly for ENRs of -10 dB and less), the noise is so great in magnitude that it becomes difficult to visually determine whether or not an edge is actually present.

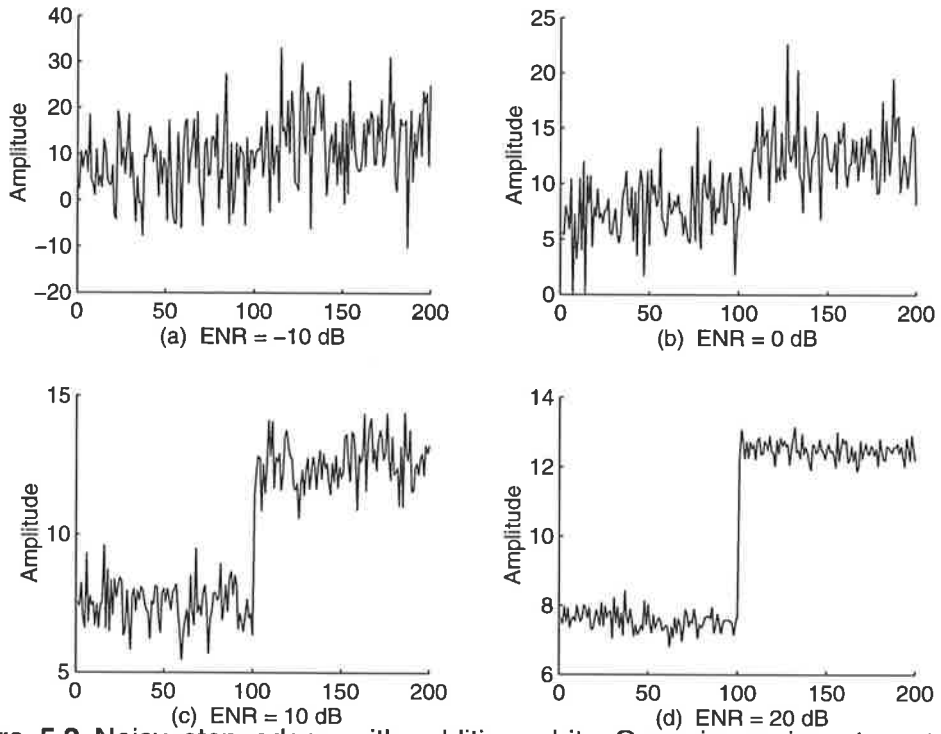


Figure 5.2 Noisy step edges with additive white Gaussian noise, $I_O = 10$ and $c = 0.25$.

5.2.3 Rectangular, Triangular and Gaussian Weight Distributions

In our investigations we use three common weight distributions:

- Rectangular distributions: $w_n = 1 - \delta_n$, where δ_n is the Kronecker delta function, and the index n is in the range $-r \le n \le r$, where r is the neighbourhood size.
- Triangular weight distribution: $w_n = (1 - |n/r|)$, with $w_0 = 0$.
- Gaussian distribution: $w_n = \frac{1}{\sqrt{2\pi}\sigma} \exp(-n^2/2\sigma^2)$ where $w_0 = 0$ and $\sigma = r/3$ is the Gaussian spread. This value of σ ensures that both w_{-r} and w_r are almost zero.

These distributions are all *symmetric,* i.e., $w_{-n} = w_n$ for all valid n, while the *asymmetric* version of the weight distributions is $w_n^* = w_n u_n$, where u_n is the unit step function.

5.3 Factors Affecting the SICNN Performance

In this section we investigate how the output noise variance, shape of the output edge response and the PNR affect the edge detection performance of a SICNN. These factors do not affect the performance independently, rather they are all inter-related.

We begin by investigating the effect of the SICNN output noise variance on its performance and show its relation to the sum of squared weights (SSW). The SSW also determines the shape of the weight distribution, provided the sum of weights (W) is constant, and thus determines the shape of the edge response, which in turn affects the HR. The third indicator to the performance is the peak response to noise ratio (PNR), which is affected by both the SICNN and edge parameters. We experimentally show how the performance is related to the PNR, and investigate the dependence of the PNR on the SICNN parameters, and the mean input intensity and contrast.

5.3.1 Output Noise Variance

It is clear that the output noise variance of any edge detector plays an important part in the detector's performance. If there is no noise in the output, then the edge is always detected in its correct position, but when we increase the output noise variance, the performance begins to deteriorate since the edge is detected away from the true position more often. Thus, as the output noise level increases, the HR should decrease and the ESD should increase.

Recall from EQ (4.17) that the output noise variance of the linearised, feedforward SICNN with decay factor $a_i = a \ \forall i$, and weights w_j, to a constant input of intensity I and input noise variance σ_{in}^2, is

$$\sigma_{out}^2 = \frac{\left[a + f(I)\sum_{j=-r}^{r} w_j\right]^2 + [If'(I)]^2 \sum_{j=-r}^{r} w_j^2}{\left[a + f(I)\sum_{j=-r}^{r} w_j\right]^4} \sigma_{in}^2 . \qquad \text{EQ (5.3)}$$

If all of the SICNN parameters are kept constant, then the output noise only changes in direct response to a change in the input noise. Figure 5.3 shows the edge detection performance as a function of the input ENR, which is inversely proportional to the input noise variance, see EQ (5.2). As the ENR increases, the HR increases and the ESD decreases. This is expected since decreasing the output noise variance, by increasing only the input ENR, must have a positive effect on the edge detection performance.

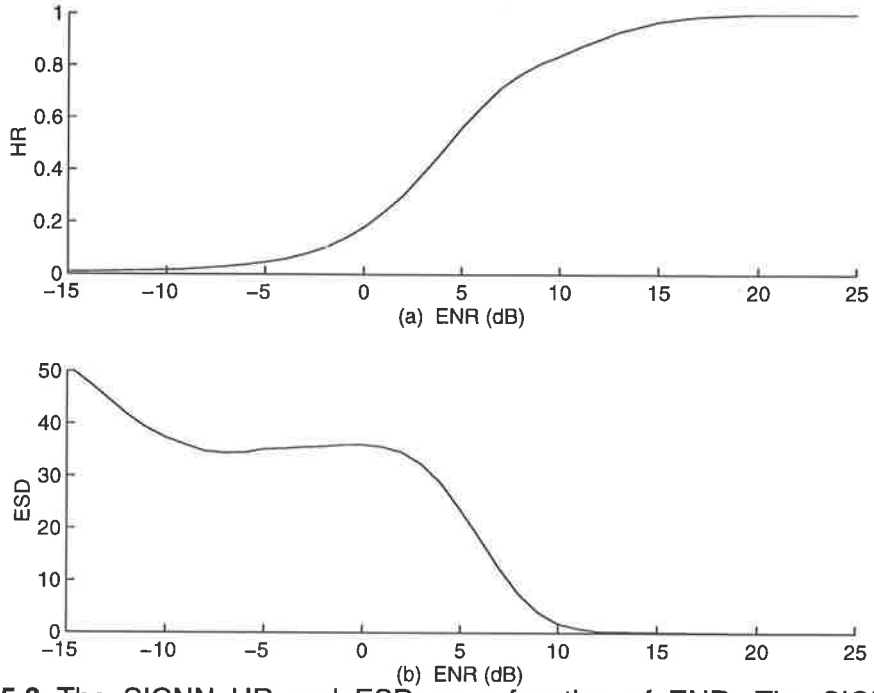


Figure 5.3 The SICNN HR and ESD as a function of ENR. The SICNN has asymmetrical, rectangular weight distribution with $W = 1$, $r = 5$, $a = 1$, $I_o = 10$, $c = 0.25$ and additive white Gaussian noise.

Minimising the Output Noise Variance

It is clear from EQ (5.3) that if the decay factor and W are kept constant, then the only possible way to change the output noise variance is to vary the sum of squared weights (SSW), $\sum w_j^2$. So from EQ (5.3), we minimise the output noise variance by minimising SSW, i.e.,

$$\text{minimise } \sum w_j^2$$

$$\text{subject to } \sum w_j = \gamma$$

where γ is a constant. We use the Lagrange multiplier method which converts a constrained optimisation problem into an unconstrained one. The functional to minimise is

$$L = \frac{1}{2}\sum_{j=-r}^{r} w_j^2 + \lambda\left(\gamma - \sum_{j=-r}^{r} w_j\right)$$

where λ is the Lagrange multiplier. Differentiating w.r.t. λ and w_j gives

$$\frac{\partial L}{\partial \lambda} = \gamma - \sum_{j=-r}^{r} w_j = 0 \quad \Rightarrow \sum_{j=-r}^{r} w_j = \gamma$$

$$\frac{\partial L}{\partial w_j} = w_j - \lambda = 0 \quad \Rightarrow w_j = \lambda \ \forall j, j \neq 0.$$

Thus, $\sum_{j=-r}^{r} w_j = 2rw_j = \lambda \quad \Rightarrow w_j = \frac{\lambda}{2r}, \forall j, w_0 = 0$. So, the weight distribution which minimises the output noise variance is in fact the rectangular distribution.

Figure 5.4 shows the ESD of the SICNN with the rectangular, triangular and Gaussian weight distributions, whose SSW are 0.2, 0.244 and 0.298, respectively. The SICNN with the rectangular weight distribution has the smallest ESD followed by the SICNN with the triangular distribution and then the SICNN with the Gaussian distribution. Although minimising the output noise variance results in a smaller ESD, this does not necessarily improve the HR. The reason is that for a given W, varying the SSW changes the shape of the weight distribution, and hence the shape of the edge response, which in turn affects the HR as we shall see shortly.

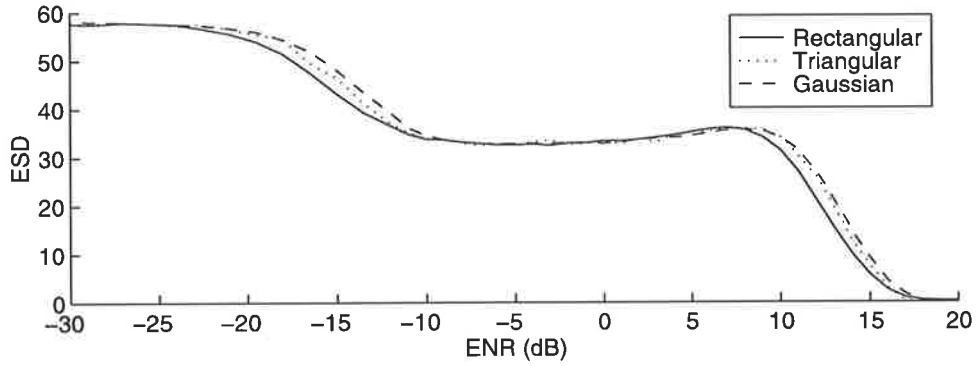


Figure 5.4 SICNN ESD for asymmetrical rectangular, triangular and Gaussian weight distributions with $W = 1$, $r = 5$, $a = 1$, $I_o = 10$, $c = 0.25$ and additive Gaussian noise.

5.3.2 Shape of the Edge Response

We described above how the SSW of the weight distribution affects the output noise variance and hence its performance, in particular the ESD. For the HR, it is insightful to look at the shape of the edge response, which depends upon the shape of the weight distribution. The HR is a "hit-or-miss" measure, so a significant number of misdetections comes from the edge being detected only a few pixels away from its true position. For example, if the edge is always detected just one pixel away from the true position, the ESD would be very small, but the HR would always be zero, indicating very poor performance according to this measure alone. These misdetections are due to the "shape" of the response to a step edge, or how quickly this output falls away from the maximum response as the distance from this maximum increases. Thus, the shape of the weight

distribution affects both the shape of the edge response and the SSW, thereby influencing both the HR and the ESD.

Figure 5.5 shows the response to a step edge of a SICNN with the asymmetrical rectangular, triangular and Gaussian weight distributions. Now consider the edge response magnitude a small distance away from the true edge position, denoted as Δy_r, Δy_t and Δy_g for the rectangular, triangular and Gaussian weight distributions, respectively. We define Δy to be the **peak response**, i.e., the maximum response which occurs at the edge position.

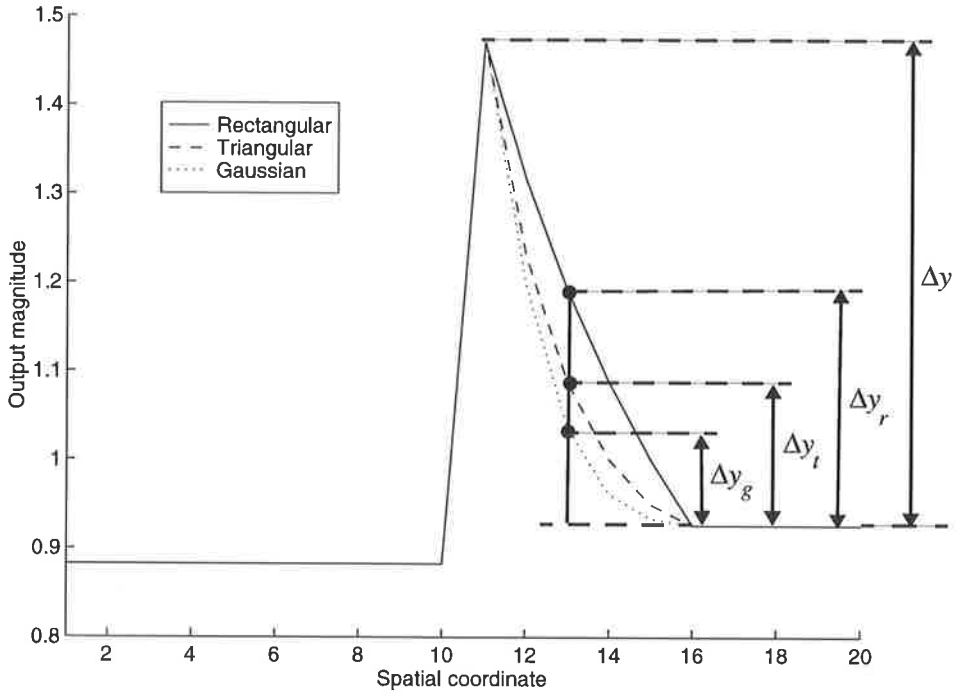


Figure 5.5 SICNN edge response for rectangular, triangular and Gaussian weight distributions. Δy is the peak response, and Δy_r, Δy_t and Δy_g are the edge responses at an arbitrary distance away from the edge point, for the SICNN with asymmetrical rectangular, triangular and Gaussian weights, respectively.

Clearly, the smaller Δy_r, Δy_t and Δy_g are compared to Δy, the greater will be the HR, i.e, the HR increases the faster the edge response decreases away from the true edge position, or the position with the maximum response. Say, for example, that Δy_r is close in magnitude to Δy. If noise is also present on the output, then there is a large probability that the overall magnitude "Δy_r+noise" is greater than "Δy+noise". This misdetection, even if only by a few pixels, causes the HR to decrease significantly. Thus, the shape of the weight distribution is an important factor in determining the HR.

Figure 5.6 shows the HR for three different weight distributions. The SICNN with the Gaussian weight distribution has the largest HR followed by the SICNN with the triangular weight distribution and then the SICNN with the rectangular weight distribution. For low ENR, however, the HR of all three SICNNs is approximately the same since the HR is dominated by the output noise variance, which we showed above to be minimum for the rectangular weight distribution. The difference, though, is insignificant.

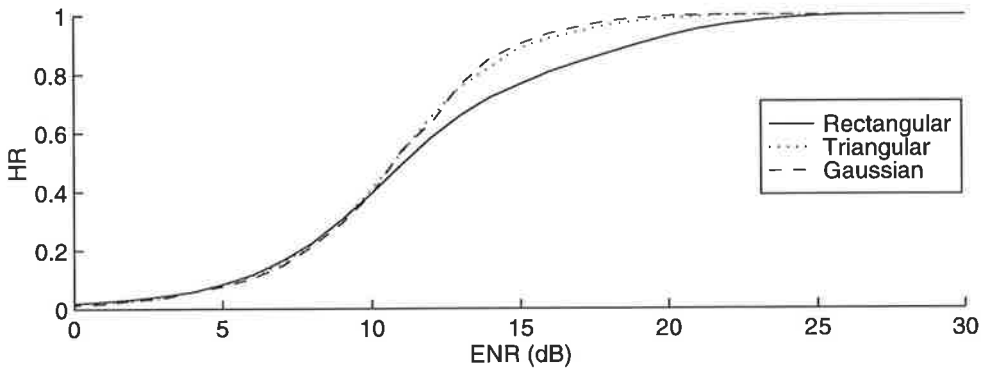
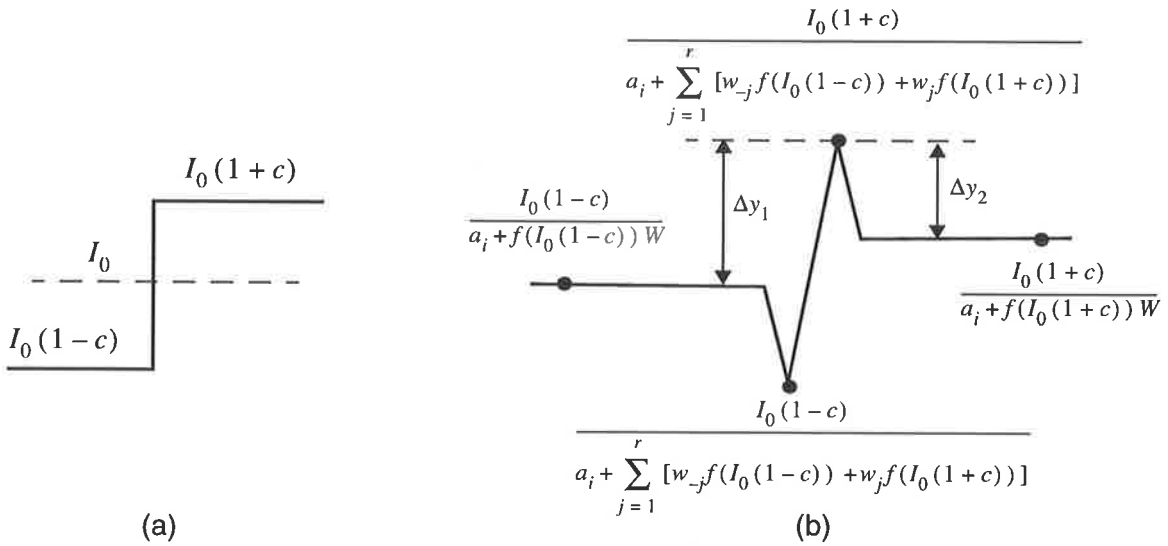


Figure 5.6 SICNN HR for asymmetrical rectangular, triangular and Gaussian weight distributions with $W = 1$, $r = 5$, $a = 1$, $I_o = 10$, $c = 0.25$. and additive Gaussian noise.

5.3.3 Peak Response to Noise Ratio

In the subsections above we discussed how the SICNN output noise variance affects its edge detection performance, particularly the ESD, and how the shape of the weight distribution or edge response affects the HR. By considering only the output noise variance and shape of the weight distribution, however, we are ignoring another important factor which determines the performance: its peak response relative to the output noise variance. Thus, to gain a better understanding of the edge detection process, the ratio of the peak response to the output noise variance, PNR, needs to be investigated.

For the SICNN edge detector, as for most other edge detectors, the magnitude of the peak response compared to the background output noise variance affects the performance. If the edge response is large compared to the output noise level, then the performance will tend to be good. Conversely, if the edge response is small compared to the output noise level, then the performance will be poor.



(a) (b)

Figure 5.7 (a) The input step edge of mean intensity I_0, and contrast c, and (b) the corresponding SICNN output. Δy is the peak response.

If $a_i = a \ \forall i$, then from Figure 5.7(b) the peak response with respect to the lower and upper background intensities are, respectively

$$\Delta y_1 = \frac{I_0(1+c)}{a+\sum_{j=1}^{r}[w_{-j}f(I_0(1-c))+w_jf(I_0(1+c))]} - \frac{I_0(1-c)}{a+f(I_0(1-c))W}, \quad \text{EQ (5.4)}$$

$$\Delta y_2 = \frac{I_0(1+c)}{a+\sum_{j=1}^{r}[w_{-j}f(I_0(1-c))+w_jf(I_0(1+c))]} - \frac{I_0(1+c)}{a+f(I_0(1+c))W}. \quad \text{EQ (5.5)}$$

For symmetrical weights ($w_j = w_{-j}, j = 1 \ldots r$) and a linear activation function, we have

$$\Delta y_1 = \frac{I_0(1+c)}{a+I_0W} - \frac{I_0(1-c)}{a+I_0(1-c)W} = \frac{2acI_0+c(1-c)I_0^2W}{a^2+aI_0(2-c)W+I_0^2(1-c)W^2}, \quad \text{EQ (5.6)}$$

$$\Delta y_2 = \frac{I_0(1+c)}{a+I_0W} - \frac{I_0(1+c)}{a+I_0(1+c)W} = \frac{c(1+c)I_0^2W}{a^2+aI_0(2+c)W+I_0^2(1+c)W^2}. \quad \text{EQ (5.7)}$$

For asymmetrical weights ($w_j = 0, j = 1 \ldots r$) and a linear activation function, the edge responses become

$$\Delta y_1 = \frac{I_0(1+c)}{a+I_0(1-c)W} - \frac{I_0(1-c)}{a+I_0(1-c)W} = \frac{2cI_0}{a+I_0(1-c)W}, \quad \text{EQ (5.8)}$$

$$\Delta y_2 = \frac{I_0(1+c)}{a+I_0(1-c)W} - \frac{I_0(1+c)}{a+I_0(1+c)W} = \frac{2c(1+c)I_0^2 W}{a^2+2aI_0W+I_0^2(1-c^2)W^2}. \qquad \text{EQ (5.9)}$$

In Appendix A, we derived the pdf of the random variable (RV) $y = max\{y_L, y_U\}$, where y_L and y_U are the RVs for the maximum output to the left ("lower") and to the right ("upper") of the edge discontinuity, respectively. If $y_e = y_U + \Delta y_2$ is the RV for the output edge pixel, then the HR is defined as

$$HR = P(y_e > y) = P(y_U + \Delta y_2 > y) = P(y - y_U < \Delta y_2).$$

We show in Appendix A that this evaluates to

$$HR = \int_{-\infty}^{\Delta y_2} R_{f_y, f_{y_U}}(s)\, ds$$

where $R_{f_y, f_{y_U}}(s)$ is the correlation of the pdfs $f_y(y)$ and $f_{y_U}(y)$. It is intuitively clear that the HR increases as Δy_2 increases. If Δy_2 is made large compared to the background noise, then clearly the HR will approach 1, but it is not explicitly clear how the output noise variance affects the HR. It is intuitive, however, that changing only the output noise variance must affect both the HR and ESD. Thus, we cannot investigate the edge response alone. As stated above, we really need to investigate the **peak response to noise ratio** (PNR), defined relative to the lower and upper background intensities as

$$PNR_1 = \frac{(\Delta y_1)^2}{\sigma^2_{out,1}}, \quad \text{and} \quad PNR_2 = \frac{(\Delta y_2)^2}{\sigma^2_{out,2}}$$

where $\sigma^2_{out,1}$ and $\sigma^2_{out,2}$ are the output noise variance in the lower and upper part of the edge and are found by replacing I in EQ (5.3) with $I_0(1-c)$ and $I_0(1+c)$, respectively. Thus, for a linear activation function, from EQ (5.8) and EQ (5.9), the PNR for an asymmetrical weight distribution is

$$PNR_1 = \frac{4[a+I_0(1-c)W]^2}{[a+I_0(1-c)W]^2 + I_0^2(1-c)^2 SSW} \cdot ENR \qquad \text{EQ (5.10)}$$

$$PNR_2 = \frac{4(1+c)^2 I_0^2 W^2}{[a^2+2aI_0W+I_0^2(1-c^2)W^2]^2} \frac{[a+I_0(1+c)W]^4}{[a+I_0(1+c)W]^2 + I_0^2(1+c)^2 SSW} ENR \qquad \text{EQ (5.11)}$$

where $ENR = c^2 I_0^2/\sigma_{in}^2$ is the edge-to-noise ratio. Clearly, both PNRs are directly proportional to the ENR. In general, the value of PNR_1 is different to PNR_2. When the effect of these PNRs on the performance is considered it is actually the smaller of PNR_1

and PNR_2 which is critical in determining the performance. Thus, the overall PNR is defined as

$$PNR = min\{PNR_1, PNR_2\}. \quad \text{EQ (5.12)}$$

All PNR values used henceforth are this minimum value.

Performance with I_o/a

Note that we can readily eliminate the decay factor from the expression of PNR_1 and PNR_2 by letting $\hat{I}_0 = I_0/a$. EQ (5.10) and EQ (5.11) can then be rewritten as

$$PNR_1 = \frac{4\left[1+\hat{I}_0(1-c)W\right]^2}{\left[1+\hat{I}_0(1-c)W\right]^2+\hat{I}_0^2(1-c)^2 SSW} \cdot ENR \quad \text{EQ (5.13)}$$

$$PNR_2 = \frac{4(1+c)^2\hat{I}_0^2 W^2}{\left[1+2\hat{I}_0 W+\hat{I}_0^2(1-c^2)W^2\right]^2}\frac{\left[1+\hat{I}_0(1+c)W\right]^4}{\left[1+\hat{I}_0(1+c)W\right]^2+\hat{I}_0^2(1+c)^2 SSW}ENR. \quad \text{EQ (5.14)}$$

Figure 5.8 shows the PNR, which is the minimum of PNR_1 and PNR_2, as a function of $\hat{I}_0$, c, W, and SSW. In both EQ (5.13) and EQ (5.14), $\hat{I}_0$ is raised to the same power in the numerator and denominator, as is the case for W. Thus, increasing $\hat{I}_0$ or W eventually causes the PNR to saturate and approach a constant value (Figure 5.8(a)). Both PNR_1 and PNR_2 are also inversely related to SSW, so increasing SSW decreases the PNR, but this is not obvious on the scale shown in (b). Likewise in (c), increasing c also increases the PNR gradually. In both (b) and (c), the PNR eventually saturates as $\hat{I}_0$ increases, for the reasons explained above.

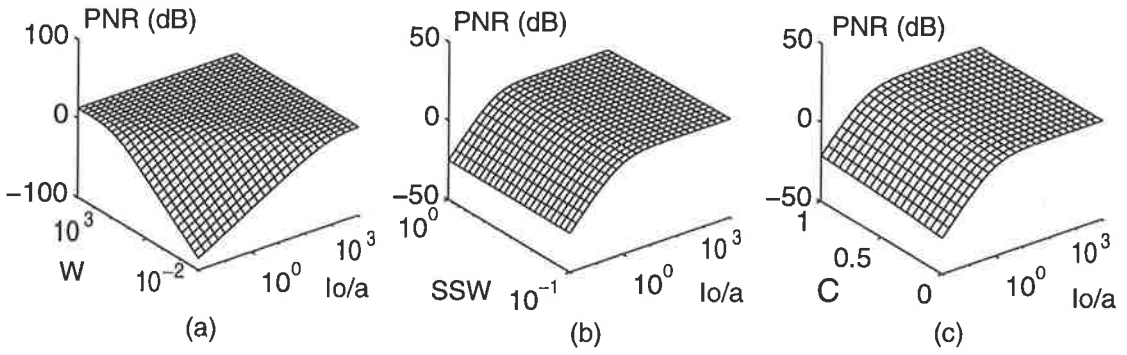


Figure 5.8 PNR (normalised by the ENR) as a function of Io/a and (a) W, (b) SSW, and (c) contrast.

In Figure 5.9 we take slices of Figure 5.8(c) and plot the PNR as a function of $\hat{I}_0$. As $\hat{I}_0$ increases, so too does the PNR, until eventually the PNR either decreases slightly or remains constant, as expected from Figure 5.8. A constant PNR indicates constant edge detection performance. It is interesting to note that the peak PNR values occur where the curves for PNR_1 and PNR_2 intersect. This is illustrated in Figure 5.10 for $c = 0.1$.

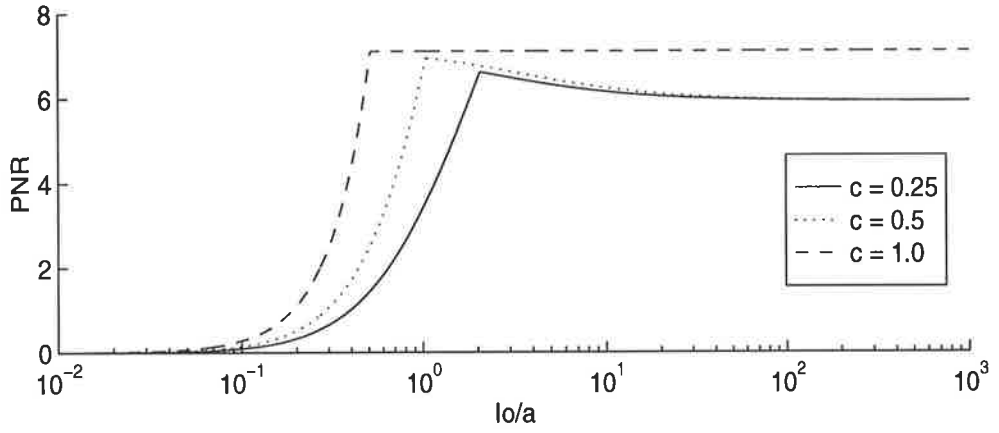


Figure 5.9 PNR as a function of (a) $\hat{I}_0$, for $W = 1$, $SSW = 0.2$ and $ENR = 5$ dB for additive Gaussian noise.

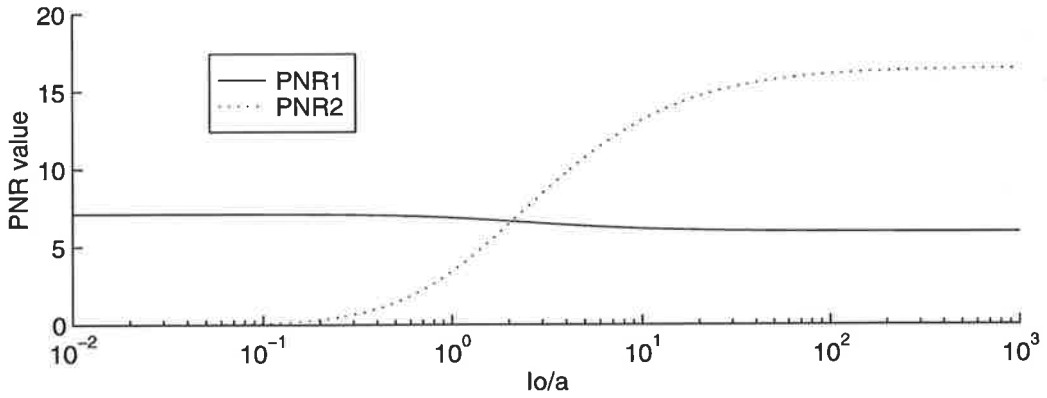


Figure 5.10 PNR_1 and PNR_2 as a function of $\hat{I}_0$ for $W = 1$, $SSW = 0.2$, $c = 0.1$ and $ENR = 5$ dB for additive Gaussian noise.

Figure 5.11 shows the performance as a function of $\hat{I}_0$ for different contrasts. From Figure 5.9, increasing $\hat{I}_0$ increases the PNR, and this is reflected in an improvement in the performance. For very large $\hat{I}_0$ the PNR saturates, hence the performance remains constant with varying $\hat{I}_0$. The most important observation from Figure 5.11, however, is that we obtain the optimum HR, ESD and EB for values of $\hat{I}_0$ which correspond almost

exactly to the peaks of the PNR curves shown in Figure 5.9. These peaks occur at the intersection of the curves for PNR_1 and PNR_2. Thus, for a given I_0, the decay factor that gives equal values of PNR_1 and PNR_2 is close in value to the decay factor that optimises the performance. This will be discussed in more detail shortly.

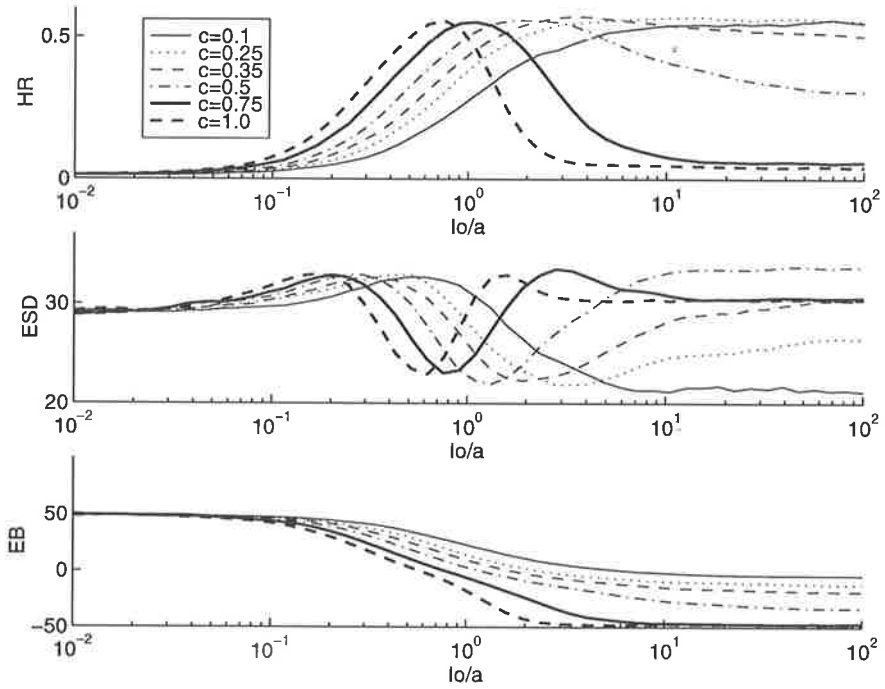


Figure 5.11 SICNN performance as a function of $\hat{I}_o$. The SICNN has asymmetrical, rectangular weights with $W = 1$, $SSW = 0.2$, $r = 5$, $ENR = 5$ dB and additive Gaussian noise.

Performance with $\Sigma w_j/a$

If we assume that SSW is negligible, and with $\hat{W} = W/a = \sum_{j=-r}^{-1} w_j/a$, we can readily eliminate the decay factor from the expression of the PNR_1 and PNR_2, thus EQ (5.10) and EQ (5.11) can be re-written as

$$PNR_1 = \frac{4\,[1 + I_0\,(1-c)\,\hat{W}]^2}{[1 + I_0\,(1-c)\,\hat{W}]^2 + I_0^2\,(1-c)^2\,SSW} \cdot ENR\,, \qquad \text{EQ (5.15)}$$

$$PNR_2 = \frac{4\,(1+c)^2\,I_0^2 W^2}{[1 + 2I_0 W + I_0^2\,(1-c^2)\,W^2]^2} \frac{[1 + I_0\,(1+c)\,W]^4}{[\,(1 + I_0\,(1+c)\,W)^2 + I_0^2\,(1+c)^2 SSW]} ENR\,. \qquad \text{EQ (5.16)}$$

Figure 5.12 illustrates the PNR, which is the minimum of PNR_1 and PNR_2, as a function of $\hat{W}$, SSW, I_0 and c. In EQ (5.15) and EQ (5.16) for large $\hat{W}$, the PNR has $\hat{W}$ raised to

equal powers in both the numerator and denominator, hence when $\hat{W}$ increases, the PNR eventually saturates. The PNR is inversely related to SSW, so by increasing SSW we decrease the PNR but the variation is too small to be observed in Figure 5.12(a). In (b) the PNR also saturates as I_0 increases since we have equal powers of I_0 in both the numerator and denominator of PNR_1 and PNR_2, and in (c) the PNR increases gradually as the contrast increases from 0 to 1. A contrast of 1 gives the largest PNR with respect to contrast. From this figure we conclude that the PNR saturates for large I_0 and $\hat{W}$, hence we expect the performance to remain constant for these values of I_0 and $\hat{W}$.

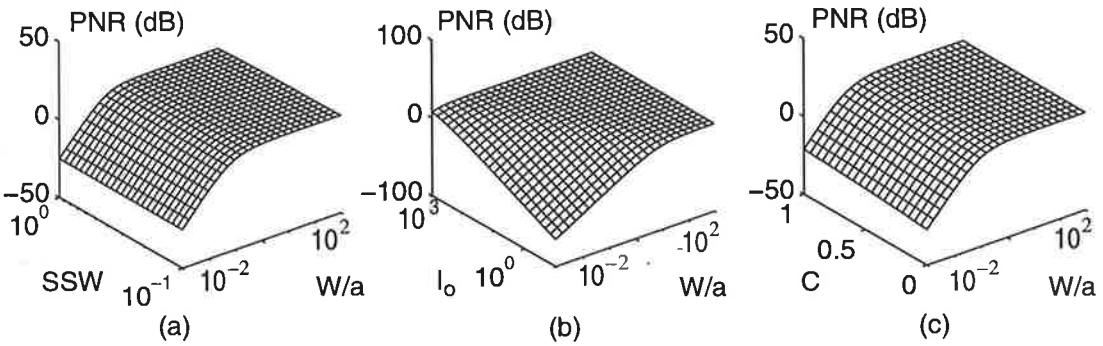


Figure 5.12 PNR as a function of W/a and (a) SSW, (b) I_o and (c) contrast.

Figure 5.13 shows the PNR as a function of $\hat{W}$, for different values of c and I_0. As expected, in (a) the PNR increases as $\hat{W}$ increases, and levels off for large $\hat{W}$. Likewise in (b), the PNR increases as $\hat{W}$ increases, and levels off for large $\hat{W}$ reaching this constant value more rapidly for larger I_0. As in the case of varying $\hat{I}_0$ above, the PNR peaks at the point where the curves of PNR_1 and PNR_2 intersect.

Figure 5.14 shows the performance as a function of $\hat{W}$. From Figure 5.13(a), increasing $\hat{W}$ causes the PNR to increase, resulting in an improvement in the performance as evident in Figure 5.14. For very large $\hat{W}$, the PNR levels off and hence the performance does not vary much as $\hat{W}$ increases. As with the results for varying $\hat{I}_0$ above, the performance is optimal at the value of $\hat{W}$ which also corresponds to the peak PNR as shown in Figure 5.13. This peak value corresponds to the value of $\hat{W}$ where both PNR_1 and PNR_2 are equal. Thus, for a given I_0 and W, the decay factor which gives equal PNR with respect to the lower and upper output edge backgrounds, appears to be close to the experimental decay factor that optimises the performance.

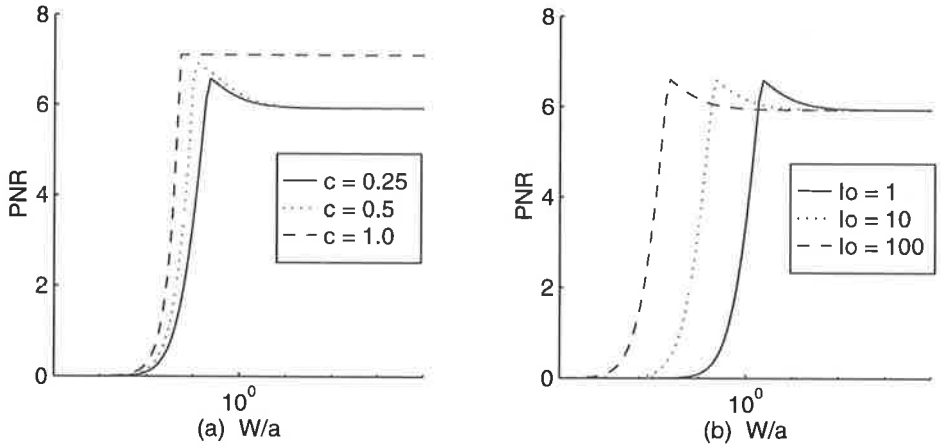


Figure 5.13 PNR as a function of $\hat{W}$ for different (a) contrast and (b) I_o. The SICNN has $r = 5$ and $SSW = 0.1$.

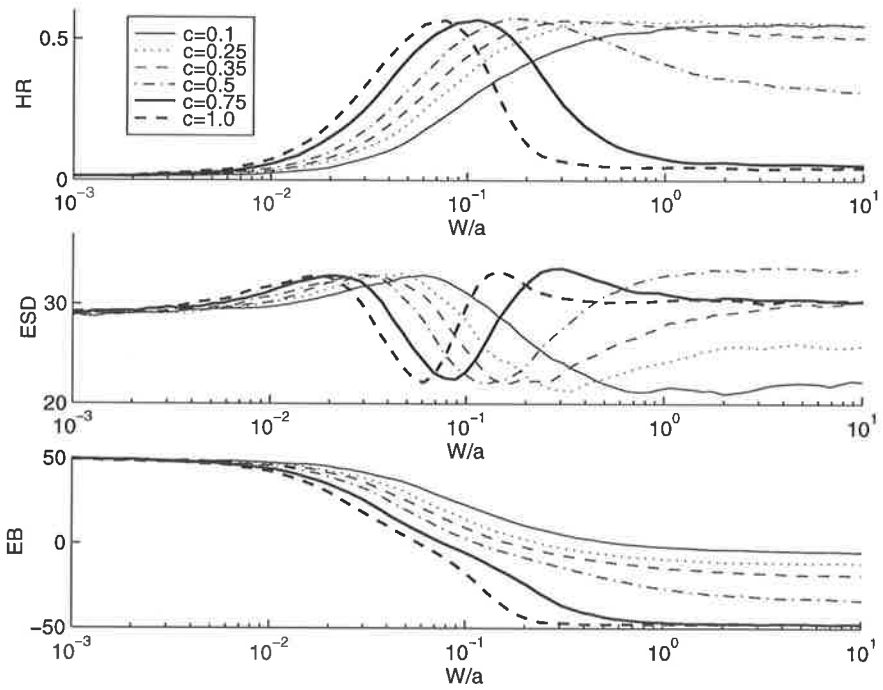


Figure 5.14 SICNN performance as a function of W/a for different input contrast. The SICNN has asymmetrical, rectangular weights with $r = 5$, $I_o = 10$ and additive Gaussian noise.

To investigate this, we now compare the experimental value of $\hat{W}$ that gives optimal performance, with the value of $\hat{W}$ for which $PNR_1 = PNR_2$. Table 5.1 gives the value of $\hat{W}$ which optimises the performance for a number of different I_0 and c. This value of $\hat{W}$ is deemed to be the optimal value. It is immediately obvious from the table that the optimal

$\hat{W}$ is inversely proportional to I_0. Thus, for a given W, the optimal decay factor appears to be directly proportional to I_0.

Table 5.2 shows the value of $\hat{W}$ for which $PNR_1 = PNR_2$ for the same I_0 and c as in Table 5.1. The experimental values of the optimal $\hat{W}$ are very close to the theoretical values, however the experimental values seem to be consistently larger, though by only a small amount. Some possible reasons for the discrepancies are that the performance was not simulated on a fine enough scale of $\hat{W}$, or the noise variance equations used to compute PNR_1 and PNR_2 are not exactly equal to the actual output noise variances. Furthermore, the PNR does not take into account the shape of the weight distribution and thus the shape of the edge response, which we know from Section 5.3.2 affects the HR.

Table 5.1 Experimentally optimal $\hat{W}$ for various I_o and c.

		Mean Input Intensity, I_o					
		1	10	20	50	100	200
Contrast	0.1	5.95	0.617	0.294	0.128	0.0595	0.0316
	0.25	2.26	0.235	0.116	0.0474	0.0244	0.0120
	0.35	1.68	0.175	0.0862	0.0326	0.0175	0.00862
	0.5	1.20	0.116	0.0595	0.0239	0.0116	0.00595
	0.75	0.771	0.0771	0.0381	0.0155	0.0077	0.00381
	1.0	0.573	0.0573	0.0294	0.0113	0.0057	0.002936

Table 5.2 Theoretical $\hat{W}$ for which $PNR_1 = PNR_2$ for various I_o and c.

		Mean Input Intensity, I_o					
		1	10	20	50	100	200
Contrast	0.1	5.11	0.511	0.256	0.102	0.0511	0.0256
	0.25	2.07	0.207	0.104	0.0414	0.0207	0.0104
	0.35	1.48	0.148	0.0741	0.0297	0.0148	0.00741
	0.5	1.04	0.104	0.0519	0.0207	0.0104	0.00519
	0.75	0.688	0.0688	0.0344	0.0138	0.00688	0.00344
	1.0	0.513	0.0513	0.0256	0.0103	0.00513	0.00256

Figure 5.15 shows the performance as $\hat{W}$ is varied for different I_0 and a contrast of 0.25. From Figure 5.13(b) increasing $\hat{W}$ causes the PNR to increase, peak and then eventually levels off, reaching this constant value more quickly for larger I_0. These characteristics

are also observed in the experimental results. Increasing $\hat{W}$ causes the performance to improve, peak, and eventually level off, with the performance reaching this constant value more quickly for larger I_0. As with the results above, the optimum performance occurs approximately at the value of $\hat{W}$ where the PNR peaks, or where $PNR_1 = PNR_2$.

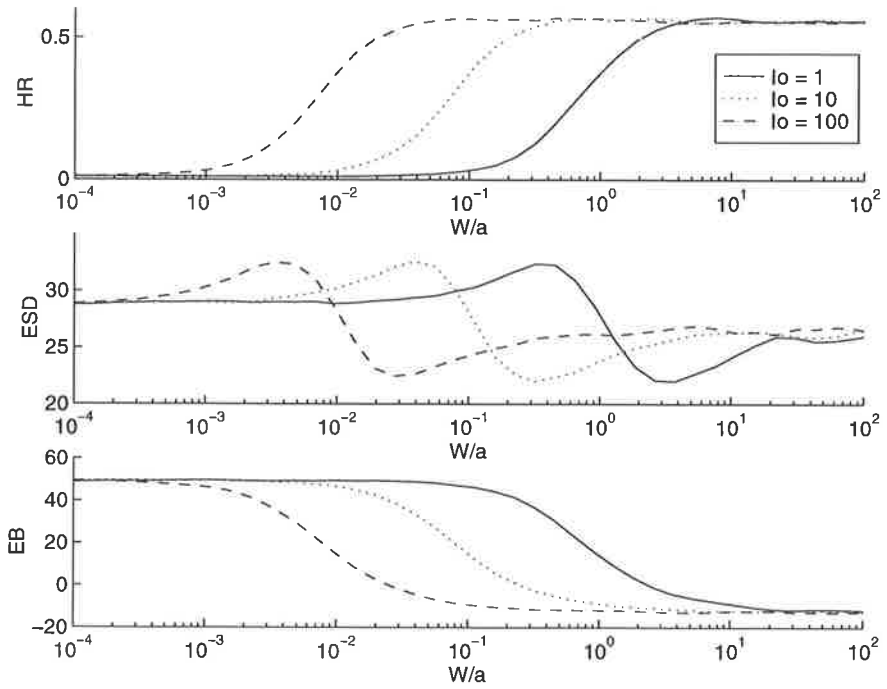


Figure 5.15 SICNN performance as a function of $\hat{W}$ for different I_0. The SICNN has asymmetrical, rectangular weights with $r = 5$, $c = 0.25$ and additive white Gaussian noise.

5.4 Effect of Weight Distribution on the Performance

We now investigate the effect of the weight distribution on the edge detection performance. In particular we look at the effect of the weight's symmetry, SSW and neighbourhood size on the performance.

5.4.1 Symmetric versus Asymmetric Weight Distributions

The symmetric weights are chosen such that $w_{-j} = w_j$, $j = 1, 2, \ldots, r$. For an input step edge the output of such a SICNN was shown in Figure 4.3(b). If this output is first demeaned, then the position of the edge corresponds to the zero-crossing in the output. Using symmetrical weights also gives a peak at the edge position, hence the edge can also

be located at the position of the maximum output. Thus, for a SICNN with symmetrical weights either zero-crossing or maximum thresholding can be used to detect edges.

An asymmetrical weight distribution has $w_j = 0$ for $j \geq 0$. A SICNN with asymmetrical weights has a peak in its output at the position of the edge as shown in Figure 4.4(a) - so maximum-based thresholding is used to locate the edge. Of course, if the weights are chosen randomly, then any edge in the input may not appear as a peak in the output. The step edge responses for both the symmetrical and asymmetrical rectangular weight distributions are compared in Figure 5.16, where the sum of weights (W) for both distributions is equal to 1.

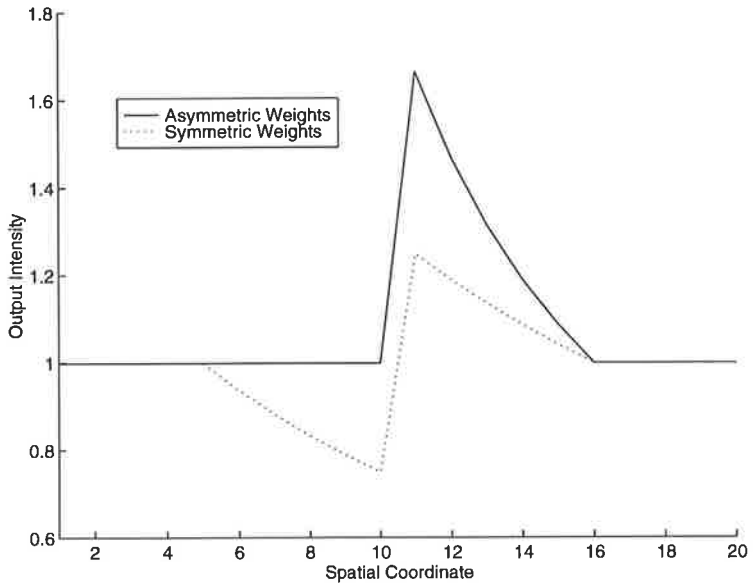


Figure 5.16 SICNN step edge responses for the asymmetrical and symmetrical rectangular weight distributions. In both cases, $W = 1$, $r = 5$, $a = 0.01$, $I_o = 10$, and $c = 0.25$.

Figure 5.17 compares the edge detection results of the SICNN with asymmetrical and symmetrical, rectangular weight distributions. For the SICNN with the symmetrical weights, the edge is detected using both zero-crossing thresholding (ZCT) and maximum thresholding (MT). The SICNN with asymmetrical weights has, in general, better performance than the SICNN with symmetrical weights using either ZCT or MT. For ENR greater than about 10 dB the SICNN with symmetrical weights and ZCT has slightly better HR than the SICNN with asymmetrical weights but worse ESD and EB. For the SICNN with symmetrical weights, using zero-crossing thresholding is always better than using maximum thresholding.

An obvious factor which causes the performance of the SICNN with symmetrical weights and ZCT to be worse than that of the SICNN with asymmetrical weights, is the extra processing step needed for locating the largest zero-crossing. This is essentially a derivative-like operation, and it can cause the performance to deteriorate.

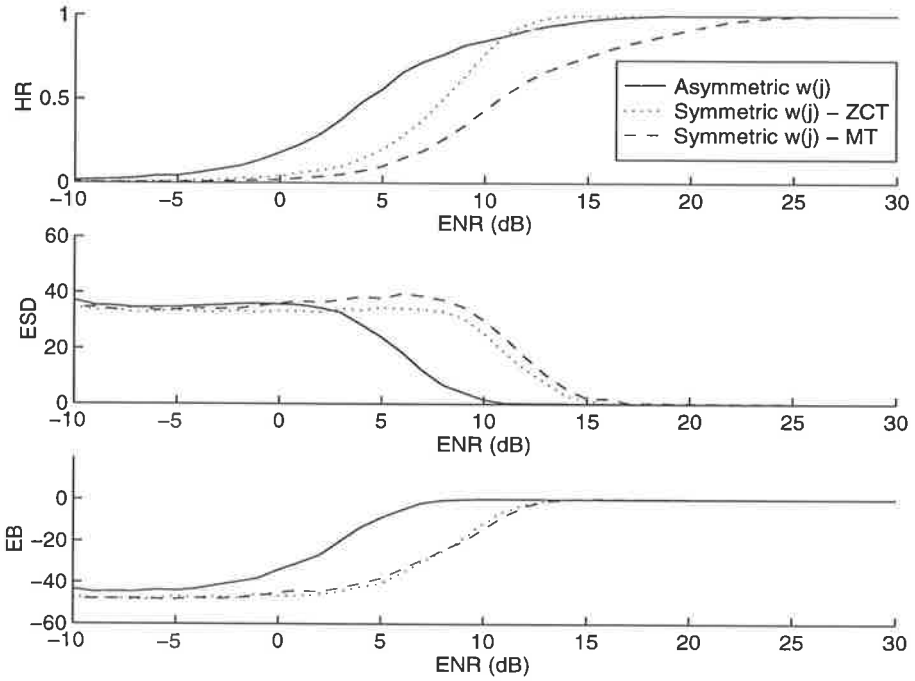


Figure 5.17 SICNN performance with the asymmetrical and symmetrical rectangular weight distributions with zero-crossing (ZCT) and maximum thresholding (MT). Both SICNN have $W = 1$, $r = 5$, $a = 0.01$, $I_0 = 10$, $c = 0.25$ and additive Gaussian noise.

5.4.2 Varying Weight Distribution Shape

We discussed in Section 5.3.3 that the PNR decreases as the sum of squared weights, SSW, increases. By increasing SSW the peak response does not change, only the output noise increases, provided the decay factor and W are constant. Thus, when SSW varies we only need to consider the output noise variance and not its peak response.

We showed in Section 5.3.1 that by increasing SSW, the output noise variance increases, causing the ESD to increase. When we increase SSW for a fixed W, the width of the SICNN edge response decreases and from Section 5.3.2 this increases the HR.

Figure 5.18 shows the performance of the SICNN with the Kaiser weight distribution (Proakis and Manolakis, 1992). The Kaiser distribution has a free parameter β which

varies the shape of the weights and hence the SSW. $\beta = 0$ gives the rectangular weight distribution, and as β increases, the width of the edge response reduces causing both the SSW and output noise variance to increase.

The decay factor is chosen to ensure that the value of $\hat{W}$ gives optimal performance according to Table 5.1. The results in Figure 5.18 indicate that increasing β increases the ESD since a larger β gives a weight distribution with larger SSW and output noise variance. Increasing β causes the edge response to become narrower, hence the HR increases, as discussed in Section 5.3.2.

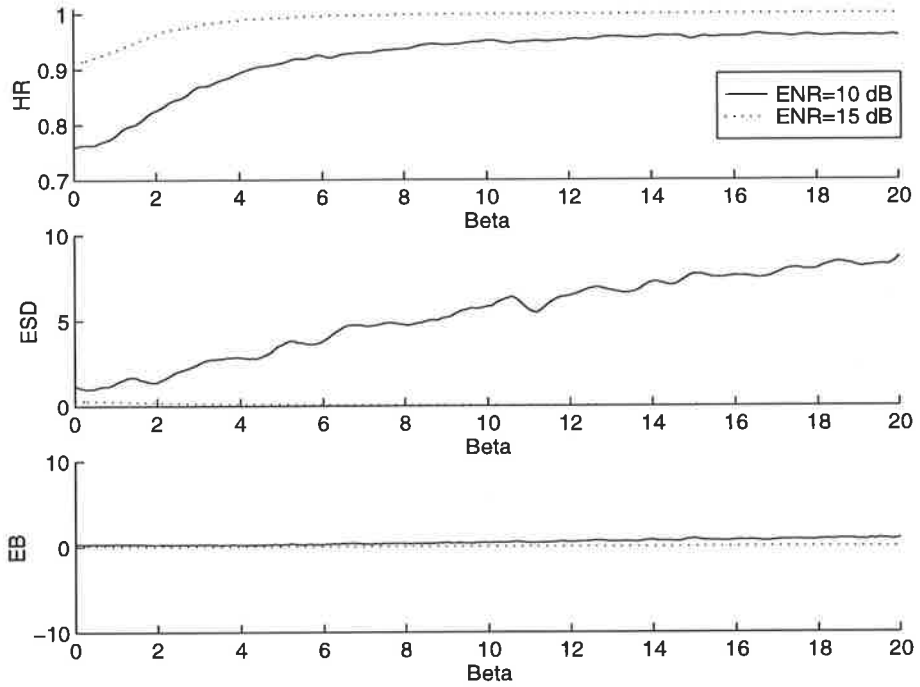


Figure 5.18 SICNN performance as a function of β for the asymmetrical Kaiser weight distribution, with $\hat{W} = 0.239$, $r = 5$, $I_0 = 10$, $c = 0.25$ and additive Gaussian noise.

We can also interpret the results in Figure 5.18 by examining the weight distribution's spectrum. Although the SICNN is not linearly related to that of weight distribution, it is still insightful to examine the characteristics of the weight's spectrum, such as the main-lobe width and side-lobe heights, and quantitatively link them to the performance. When $\beta = 0$ the Kaiser distribution is identical to the rectangular weight distribution, hence the main-lobe width is smallest. We saw in Section 5.3.1 that the SICNN with this rectangular distribution has the smallest ESD which we can see in Figure 5.18. It also has poor HR, which we expect from the shape of the edge response with the rectangular weight

distribution. By increasing β, the main-lobe width increases and the side-lobes decrease, resulting in both the HR and ESD increasing. Thus, β gives a trade-off between spectrum's main-lobe width and the side-lobes, which is reflected in the trade-off between the HR and ESD.

5.4.3 Neighbourhood Size

For any linear filter with a fixed shape of weight, the size or support of the filter determines its frequency characteristics and hence its performance with noisy signals. Consider a linear edge detector, such as Canny's operator; by increasing the neighbourhood size we reduce the output noise, but we also reduce the resolution of the edge response. Thus, there is an inherent trade-off between the amount of noise smoothing and the resolution of the edge response. Likewise for the SICNN, if we change only the neighbourhood size, then both the output noise variance and its resolution to any input edge must be affected.

For a given decay factor and W, the neighbourhood size does not affect the peak response, whereas, from EQ (5.3), it does affect the output noise variance by varying SSW. For example, for the asymmetrical, rectangular weight distribution with constant W, the SSW is $(1/r)$ where r is the neighbourhood size. Hence, we expect that by increasing r, and thus decreasing SSW and the output noise variance, we can improve the HR and ESD. As the neighbourhood size increases, the edge response becomes broader as evident in Figure 5.19, hence the HR decreases.

In Figure 5.20 the HR decreases as its neighbourhood size increases since we are continually increasing the width of the edge response. The ESD also decreases as its neighbourhood size increases since increasing the weight distribution's neighbourhood size with fixed W causes the output noise variance to decrease, which from Section 5.3.1, causes the ESD to decrease. For very noisy inputs, the HR is affected by the output noise variance, so increasing r causes the HR to actually increase slightly.

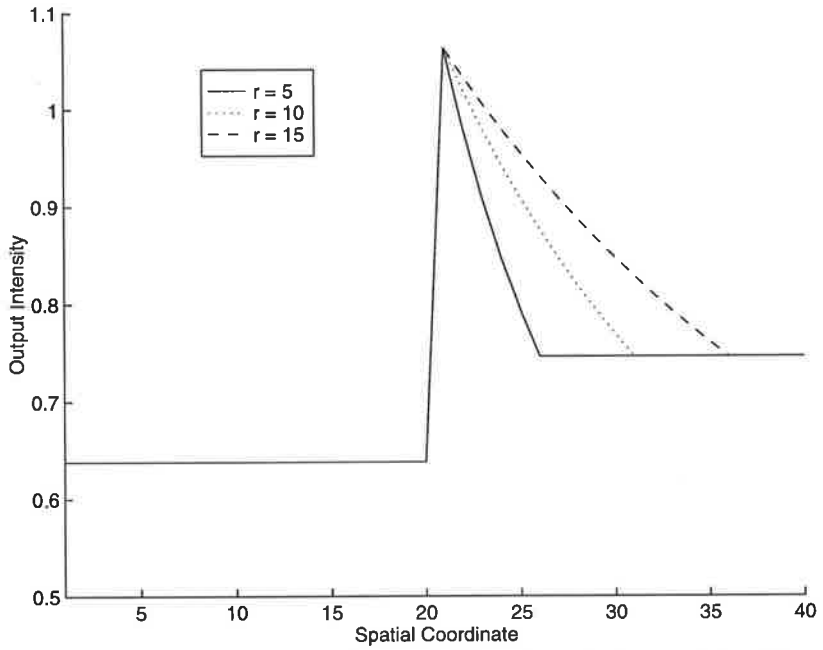


Figure 5.19 SICNN edge response for neighbourhood sizes of 5, 10 and 15. The weight distribution in all cases is asymmetrical and rectangular, with $\hat{W} = 0.239$, $I_o = 10$, and $c = 0.25$.

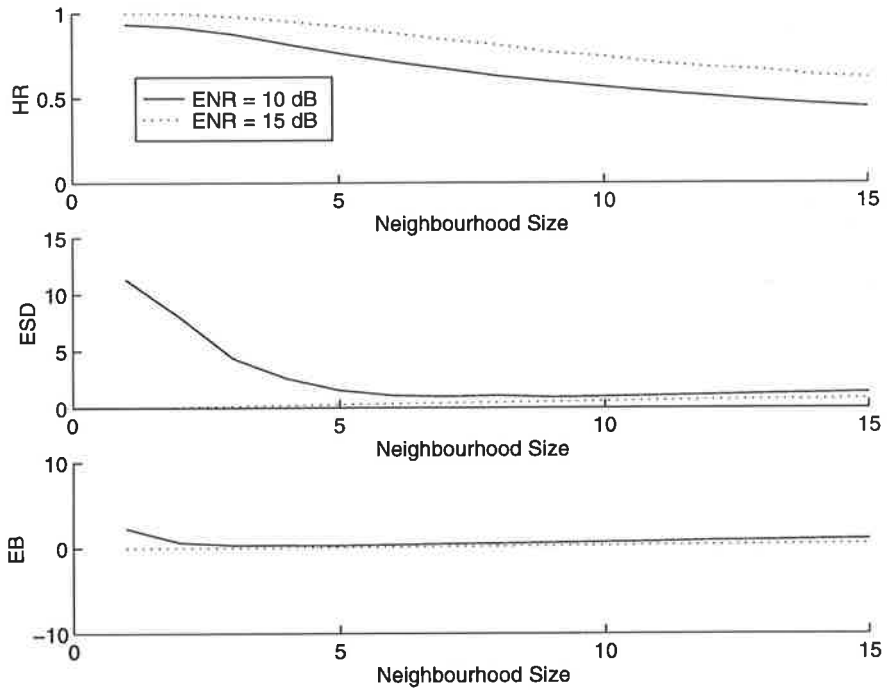


Figure 5.20 SICNN performance as the neighbourhood size varies. The SICNN has an asymmetrical, rectangular weight distribution with $\hat{W} = 0.239$, $I_o = 10$, $c = 0.25$ and additive Gaussian noise.

5.5 Conclusion

This Chapter provided an in-depth analysis of the factors of the SICNN edge response which affect its edge detection performance, and how the SICNN parameters affect these factors, and hence the edge detection performance. We first defined three performance measure, namely the HR, ESD and EB, as well as the parameters associated with an input edge, such as the contrast and ENR.

The edge detection performance is essentially determined by three factors:

- the output noise variance,
- the shape of the weight distribution and hence the edge response, and
- the peak response to noise ratio (PNR).

Increasing the output noise variance increases the likelihood of misdetecting the edge, hence the performance decreases. Making the shape of the edge response narrower increases the HR, while increasing the PNR also improves the HR and ESD. The three factors listed above are not always independent. For example, changing the shape of the edge response also changes the output noise variance and the PNR. The PNR was defined as the minimum of PNR_1 and PNR_2 with respect to the lower and upper output background intensities, respectively. For a given I_0 and W, the decay factor that gives maximum PNR approximates the optimal decay factor, and it was found to be proportional to I_0. The SICNN with the asymmetrical weight distribution, in general, performs better than the SICNN with the symmetrical weight distribution. The weight distribution's SSW determines both the output noise variance and the shape of the edge response. Increasing SSW causes the output noise variance to increase which increases the ESD. Also, increasing SSW gives a narrower edge response, hence the HR increases.

Finally we investigated the effect of the neighbourhood size on the edge detection performance and found that increasing the neighbourhood size decreases both the HR and ESD. For a fixed W and shape of weights (e.g. rectangular weights), increasing the neighbourhood size decreases the SSW and output noise variance, hence the ESD decreases. Increasing the neighbourhood size also increases the width of the edge response, hence resulting in a decrease in the HR.

Chapter 6 Choice of Optimal SICNN Parameters

6.1 Introduction

In the preceding two Chapters we investigated the effects of varying the SICNN parameters on the output noise variance and edge detection performance. In this Chapter we investigate how these SICNN parameters can be chosen to optimise the edge detection performance. We begin with the SICNN weight distribution. We observed in Chapter 5 that, for a given sum of weights, its shape affects both the HR and ESD. The weight distribution which simultaneously optimises both the HR and ESD is constructed as a constrained numerical optimisation problem, and is then solved using Lagrange multipliers. The optimal weight distribution and the corresponding performance is compared to that of the SICNN with the rectangular, triangular and Gaussian weight distributions.

In Section 6.3 we investigate the decay factor, which we showed in Chapter 5 affects the edge detection performance. The experiments in Section 5.3.3 showed that for a given sum of weights, mean input intensity and contrast, there exists a decay factor which maximises the HR, minimises the ESD, and results in zero EB. The optimal decay factor is then defined as the decay factor which gives zero EB. In Chapter 5 the optimal decay factor was found to be approximately equal to the decay factor that maximises the SICNN PNR. However, since the decay factor from the PNR method is slightly different to the experimentally optimal one, we are likely to obtain suboptimal performance. Thus, in this Chapter we investigate another technique of estimating the optimal decay factor by considering the strength of the SICNN output on either side of the discontinuity. Analytical expressions are derived for this decay factor which relate it to the edge and

SICNN parameters, for both multiplicative and additive noise. Empirical estimates for the optimal decay factor are presented for both 1-D and 2-D synthetic edges and for different noise and weight distributions.

Up to this point a linear activation function has been used since it simplifies the mathematics considerably. In Section 6.4 we investigate a number of nonlinear activation functions and their effect on the 1-D edge detection performance.

6.2 Optimal Weight Distribution

We observed in Chapter 5 how the sum of the squares of the weights SSW affects the SICNN output noise variance, and hence its performance, particularly the ESD. The shape of the weight distribution, which is related to SSW for a given W, also has a significant influence on the HR. These two conditions actually oppose one another. In general, if the weight distribution is chosen to maximise the HR, then the output noise variance and ESD will also be large. Thus, in this section we formally define the conditions needed to maximise the HR and minimise the ESD, and derive the weights simultaneously optimise both measures using Lagrange multipliers.

We show that the shape of the resultant optimal weight distribution depends upon the amount we choose to penalise a broad weight distribution in the optimisation process. The edge detection performance is then compared using the optimal weight distribution as well as the Gaussian and rectangular weight distributions.

6.2.1 Optimality Criteria

In the experiments of Chapter 5, the relationship between the shape of the weight distribution and the SICNN edge detection performance was investigated. In particular it was found for a given W that, the smaller the SSW is, the smaller the SICNN output noise variance and hence the smaller the ESD. A narrower weight distribution shape, however, has larger SSW and hence results in a larger HR. These two requirements for the weight distribution are inherently conflicting.

To determine the weights that simultaneously has a small SSW and narrow edge response, we need to

$$\text{minimise } S_1 = \left(\sum_{j=-r}^{r} w_j^2 \right)^{1/2}, \text{ and}$$

$$\text{maximise } S_2 = \sum_{j=-r}^{r} w_j / |j|^p, \; p > 0, \; j \neq 0$$

$$\text{subject to } \sum_{j=-r}^{r} w_j = 1, \; w_j > 0 \;\; \forall j, \; j \neq 0.$$

p is the *weight attenuation factor* or simply the attenuation factor. The first condition seeks to minimise SSW, i.e., minimise the ESD, while the second term attempts to make the weight distribution's shape or edge response as narrow as possible. The first term has a square root to maintain consistency in the dimensions of w_j in the optimisation process.

The term $|j|$ in the above expression to be maximised, measures the distance of pixel j from the centre element of the mask. Thus, p, determines how narrow the derived distribution is: a larger p gives smaller weighting to those weights w_j with large j, hence the distribution is forced to be narrower. In fact, it is easy to show that increasing p increases SSW. Note that when $p = 0$ in the above formulation, there is no longer any means of making the weight distribution narrow since $|j|^p = 1 \;\; \forall j$, hence we only minimise the SSW, which from Section 5.3.1, results in the rectangular weight distribution.

The above optimisation problem can be re-expressed as

$$\text{maximise } \Gamma = \frac{S_2}{S_1} = \frac{\sum_{j=-r}^{r} w_j / |j|^p}{\left(\sum_{j=-r}^{r} w_j^2 \right)^{1/2}}, \; p > 0, \; j \neq 0 \qquad \text{EQ (6.1)}$$

$$\text{subject to } \sum_{j=-r}^{r} w_j = 1, \; w_j > 0, \; w_0 \neq 0.$$

Using the Lagrange multiplier method (Reklaitis et al., 1990), the problem can be written as

$$\text{maximise } F = \Gamma + \lambda \left(1 - \sum_{j=-r}^{r} w_j \right).$$

The equations to solve for the extremum are

$$\frac{\partial F}{\partial w_j} = \frac{\partial \Gamma}{\partial w_j} - \lambda = 0, \qquad \text{EQ (6.2)}$$

$$\frac{\partial F}{\partial \lambda} = \left(1 - \sum_{j \in N_f} w_j\right) = 0. \qquad \text{EQ (6.3)}$$

To show that F is maximised, its Hessian matrix H is first computed, where

$$H = \nabla^2 F = \left[\frac{\partial F}{\partial w_i \partial w_j}\right].$$

The stationary point found from EQ (6.2) and EQ (6.3) is a maximum if H is negative semidefinite, otherwise the stationary point is either a minimum or a saddlepoint (Reklaitis et al., 1990). To solve EQ (6.2) and EQ (6.3), and to compute and check that the Hessian is negative semidefinite, are both tedious and difficult for $r \geq 2$. Instead, these equations are solved numerically using optimisation programs while the Hessian is computed first by using a symbolic maths software (Maple V), evaluated numerically, then checked to determine if it is negative semidefinite.

From the above derivation, it might appear that we can optimise EQ (6.1) with respect to the weight attenuation factor as well. Now, as the attenuation factor only affects the numerator of EQ (6.1), we would only need to maximise the term $w_j/|j|^p$ for $j \geq 1$, or in other words minimise the value of p, which is zero. However, we noted above that if $p = 0$, then the term that penalises a broad weight distribution no longer has any influence in the optimisation process, hence we only minimise SSW, which results in the rectangular weight distribution.

6.2.2 Performance with Varying Weight Attenuation Factor, p

Figure 6.1 shows the optimal 1-D weight distribution obtained from the above optimisation for $r = 5$ and various values of the weight attenuation factor. Note that for the weights derived here, the Hessian is always negative semidefinite. For $p = 0.5$, the weight distribution is almost rectangular - in fact if $p = 0$ we obtain the rectangular weight distribution. As p increases, the weight distribution becomes narrower.

Figure 6.2 shows the edge detection performance for a step edge input with the weight distributions shown in Figure 6.1. The smaller the value of p, the smaller is the resulting ESD and HR. Thus, varying the weight attenuation factor p, provides a trade-off between good ESD performance and good HR performance.

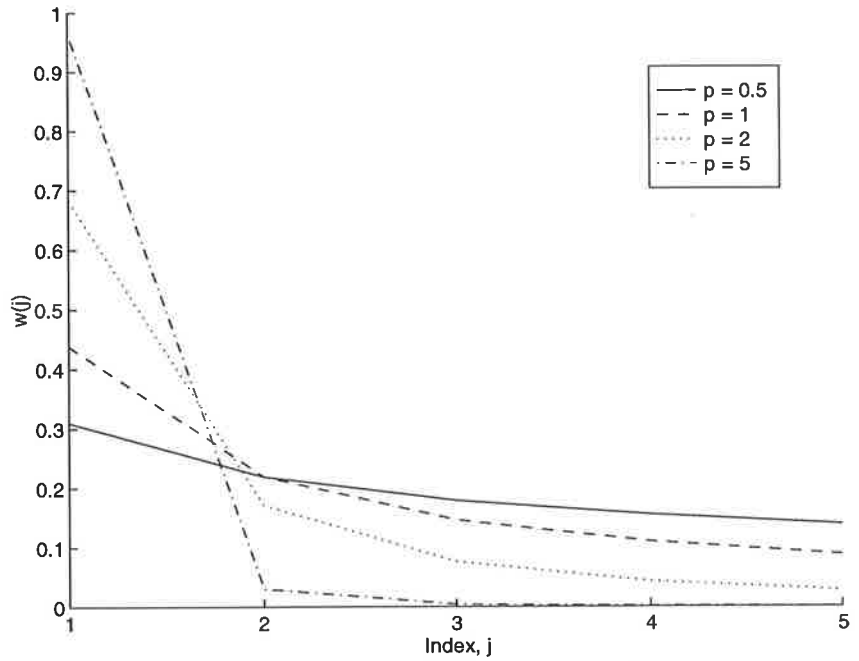


Figure 6.1 Optimal weight distribution with attenuation factor of 0.5, 1, 2, 5, and r = 5.

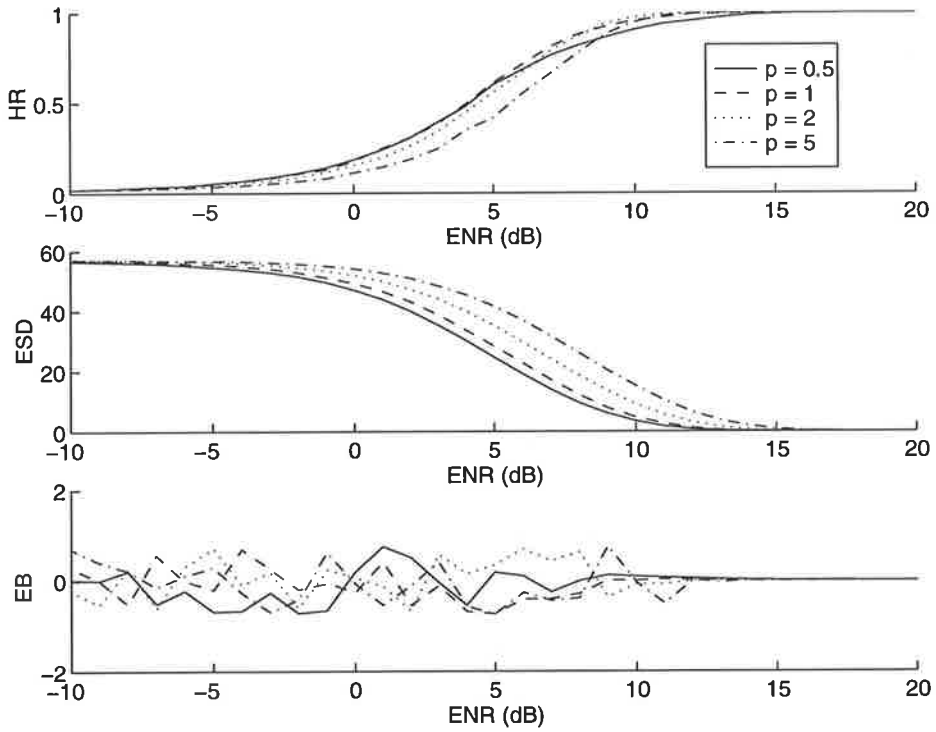


Figure 6.2 SICNN performance with the optimal weight distribution. The SICNN has decay factor chosen to give near zero EB, r = 5, with I_0 = 10, c = 0.25 and additive Gaussian noise.

To further illustrate the trade-off between the HR and ESD, Figure 6.3 shows the edge detection performance of the SICNN with the optimal weight distribution as a function of

the weight attenuation factor, p. In general, as p increases, both the ESD and HR increase. This is expected since for $p = 0$ the optimal weight distribution is in fact rectangular in shape, and we showed in Section 5.3.1 that a SICNN with the rectangular weight distribution has the smallest ESD and HR compared to the SICNN with the triangular and Gaussian weight distributions.

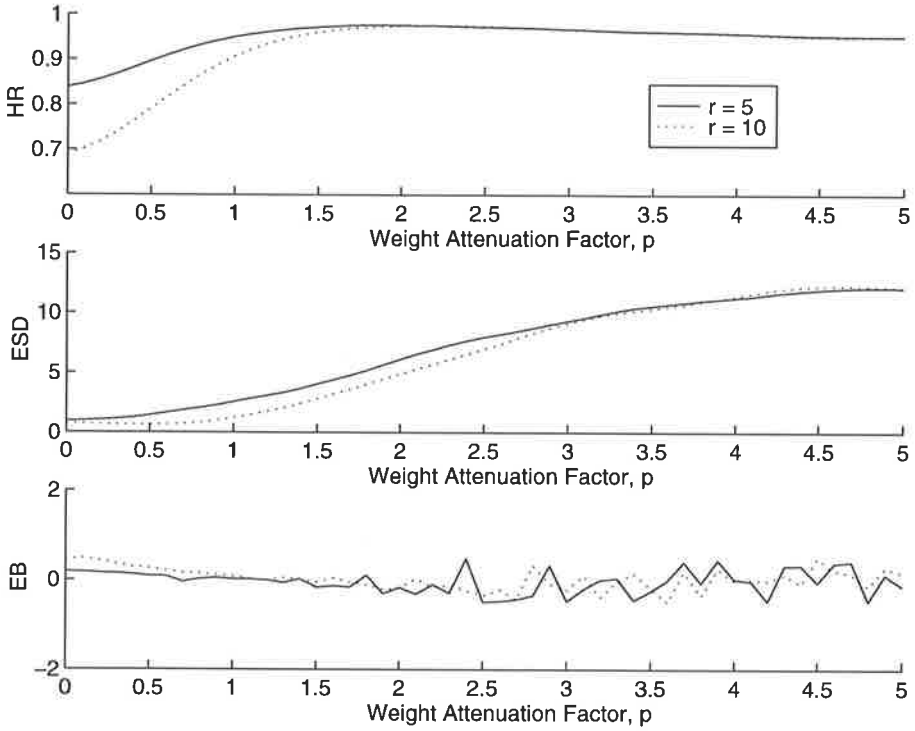


Figure 6.3 SICNN performance with the optimal weight as the weight attenuation factor varies. The input edge has $I_0 = 10$, $c = 0.25$, ENR = 10 dB with additive Gaussian noise.

6.2.3 Comparison with Other Weight Distributions

Now we compare the performance of the SICNN with the optimal weight distribution to that of the SICNN with the rectangular and Gaussian weight distributions. Figure 6.4 shows the rectangular, Gaussian, and optimal weights for $p = 0.5$ and 1. Note that both optimal weight distributions decrease more rapidly than the Gaussian distribution, but tend to level off for large j.

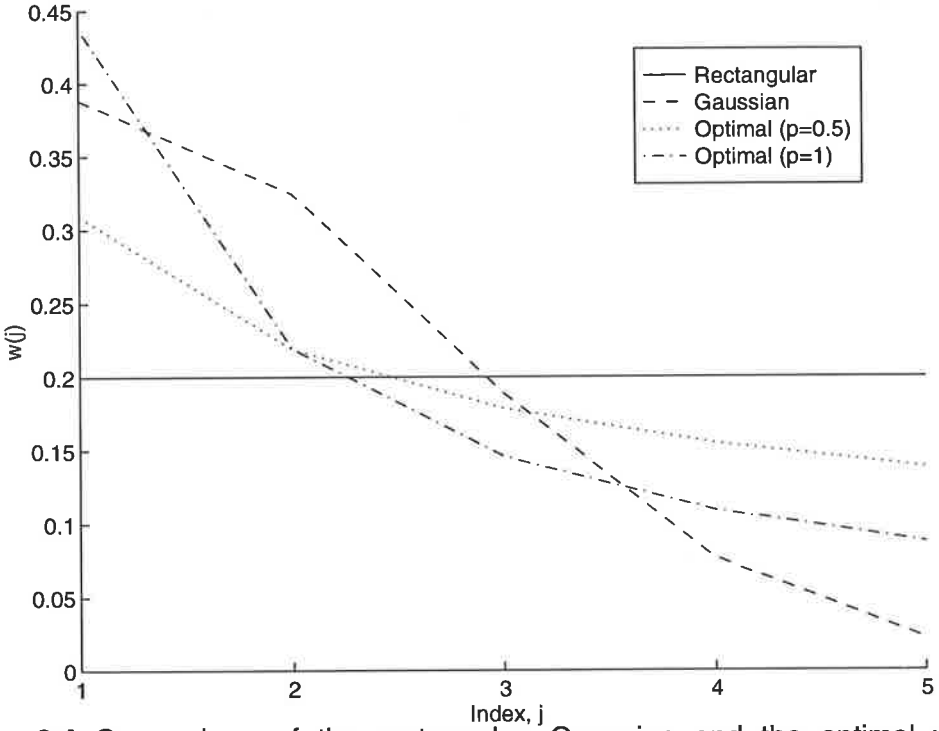


Figure 6.4 Comparison of the rectangular, Gaussian and the optimal weight distributions with attenuation factor of 0.5 and 1, for r = 5.

Figure 6.5 shows a performance comparison of the SICNN with these weight distributions. Overall, the SICNN with the rectangular weight distribution has the smallest HR and ESD, as expected. The SICNN with the $p = 1$ optimal weight distribution gives the largest HR, while the SICNNs with the Gaussian and optimal ($p = 0.5$) weight distributions have almost the same HR. For the ESD, the SICNNs with the Gaussian and optimal ($p = 1$) weight distributions are very similar, whereas the SICNN with the $p = 0.5$ optimal weight distributions has slightly smaller ESD, though still worse than the SICNN with the rectangular distribution.

Thus, the performance of the SICNN with the optimal weight distribution is, in general, in between that of the SICNN with rectangular and Gaussian weight distributions. This comes from our optimisation process for deriving the optimal weight distribution, where varying the weight attenuation factor provided the trade-off between good ESD and good HR. Note, however, that the $p = 1$ optimal weight distribution is better than the Gaussian weight distribution since the SICNN with this optimal weights has the same ESD but larger HR than the SICNN with the Gaussian weights.

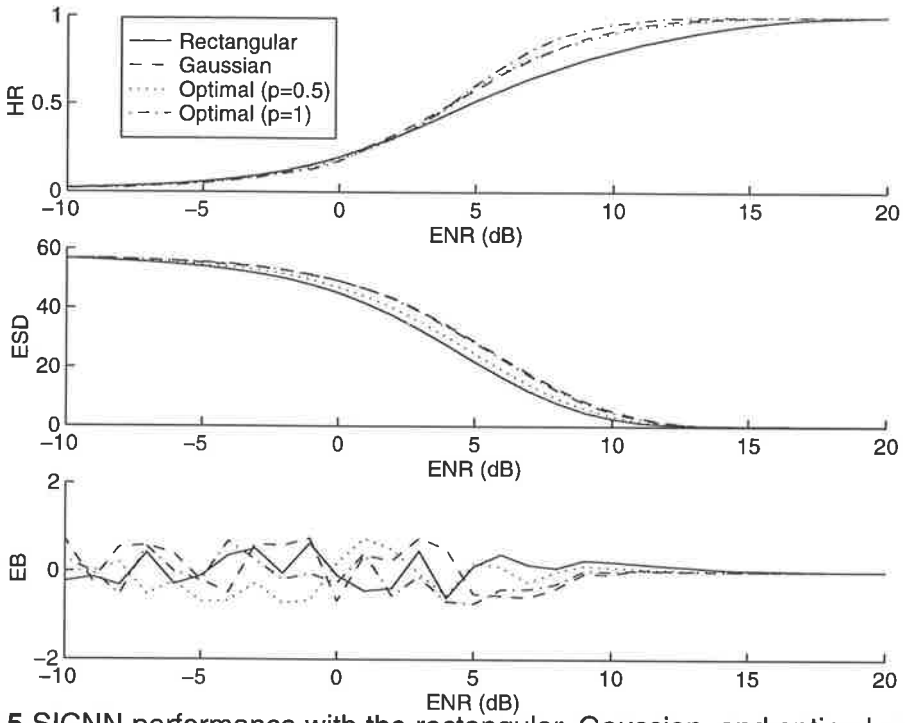


Figure 6.5 SICNN performance with the rectangular, Gaussian, and optimal weight distributions. The SICNN has decay factor chosen to give near zero EB, $r = 5$, with $I_o = 10$, $c = 0.25$ and additive Gaussian noise.

6.3 Optimal Decay Factor

The experimental results in Section 5.3.3 showed that for a given sum of weight and mean input intensity, there is a decay factor that simultaneously optimises the HR, ESD and EB. The optimal HR and ESD occur for the decay factor that gives zero (or near zero) EB. We also showed that this decay factor is approximately equal to the decay factor that maximises the PNR.[1]

Although it is easy to compute the decay factor that maximises the PNR, because it is different to the experimentally optimal value, it would still result in a suboptimal edge detection performance. Therefore, in this section we describe an alternative approach to computing the optimal decay factor that, even though it is computationally more intensive and mathematically not as elegant, can give better performance than the decay factor obtained from the PNR.

1. The decay factor that maximises the PNR is only approximately equal to the experimentally optimal decay factor because the PNR does not take into account the shape of the weight distribution, which affects the HR, and also the expression for the PNR used the linearised SICNN output noise variance which may not hold for very noisy inputs.

We begin with the general derivation of the optimal decay factor and then consider the cases for additive and multiplicative noise. We compare the performance of the SICNN with the empirically optimal decay factor, the decay factor that maximises the PNR, and the decay factor derived in this section. Finally, the analysis of the optimal decay factor is then extended to a 2-D synthetic image consisting of step edges of different mean intensities and contrasts.

6.3.1 Derivation

Recall from the experimental results in Section 5.3.3 that the decay factor that gives zero EB also maximises the HR and minimises the ESD. Our objective now is to find a means of determining the decay factor that gives zero EB, and thereby optimises the performance. A simple approach is to find the decay factor that gives uniform noise intensity across the entire output. When this occurs, it is likely that the noise does not bias the location of the detected maximum neither to the left or right of the true edge position, resulting in near-zero EB.

This is illustrated in Figure 6.6 where (a) shows a noisy step edge input to the SICNN, and (b) shows the output when $a \ll I_0 W$; the EB will be negative[1] for this output. Figure 6.6(c) shows the case when $a \gg I_0 W$, where the output is almost a scaled version of the input and hence the EB will be positive, while (d) shows the output when the decay factor is selected so that the EB is close to zero. The total noise level is roughly uniform across the entire output, resulting in zero EB.

In this derivation, we use the "-" symbol to denote all pixel to the left of the edge discontinuity and the "+" symbol for all pixels top the right of the discontinuity. With the results of Figure 6.6 in mind, the SICNN response to a noisy step edge should approximately satisfy

$$\mu_- + \lambda\sigma_- = \mu_+ + \lambda\sigma_+ \qquad \text{EQ (6.4)}$$

where λ is a constant, and $\mu_- + \lambda\sigma_-$ and $\mu_+ + \lambda\sigma_+$ are the total noise intensity to the left and right of the discontinuity, respectively. μ_- and μ_+ are the lower and upper output background intensities, respectively, and σ_- and σ_+ are the output noise variance to the

1. The EB is measured relative to the edge position, which occurs at pixel number 101 in Figure 6.6.

left and right of the edge discontinuity. Using the shorthand notation $I_- = I_0(1-c)$, $I_+ = I_0(1+c)$, and assuming a linear activation function, we can write

$$\mu_- = \frac{I_0(1-c)}{a + I_0(1-c)W} = \frac{I_-}{a + I_- W}, \qquad \text{EQ (6.5)}$$

$$\mu_+ = \frac{I_0(1+c)}{a + I_0(1+c)W} = \frac{I_+}{a + I_+ W}. \qquad \text{EQ (6.6)}$$

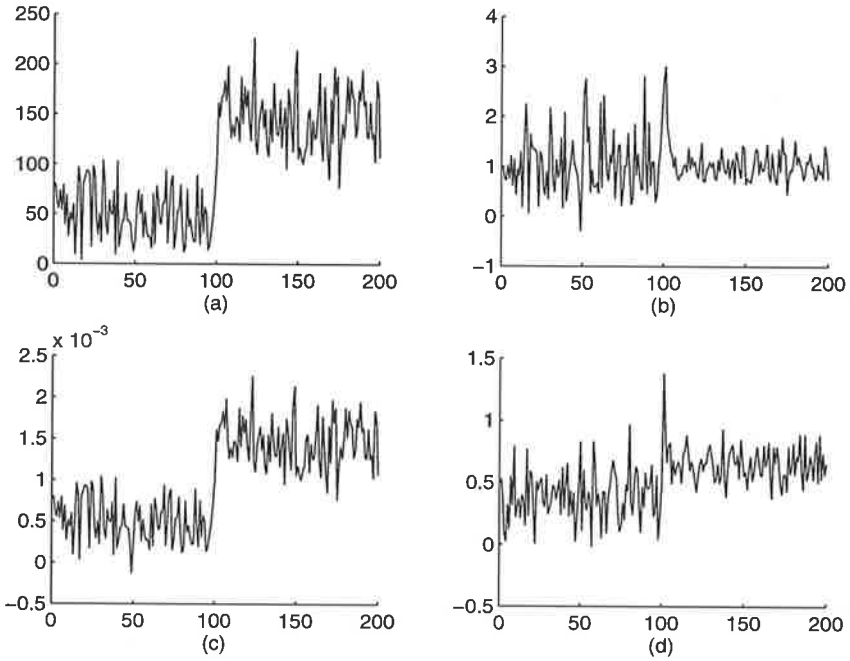


Figure 6.6 SICNN output for different decay factor values. (a) shows the input, and the output is shown when the decay factor is (b) much smaller than the I_0, (c) much larger than I_0, and (d) chosen so that the EB is zero.

Using the expression for the output noise variance derived in EQ (4.17), we can write the output noise variance to the left and right of the discontinuity as, respectively,

$$\sigma_-^2 = \frac{(a + I_- W)^2 + I_-^2 SSW}{(a + I_- W)^4}\sigma_{in}^2 \qquad \text{EQ (6.7)}$$

$$\sigma_+^2 = \frac{(a + I_+ W)^2 + I_+^2 SSW}{(a + I_+ W)^4}\sigma_{in}^2 \qquad \text{EQ (6.8)}$$

where σ_{in}^2 is the input noise variance. Using these expressions, EQ (6.4) becomes

$$\frac{I_-}{a + I_- W} + \lambda\sqrt{\frac{(a + I_- W)^2 + I_-^2 SSW}{(a + I_- W)^4}}\sigma_{in} = \frac{I_+}{a + I_+ W} + \lambda\sqrt{\frac{(a + I_+ W)^2 + I_+^2 SSW}{(a + I_+ W)^4}}\sigma_{in}. \qquad \text{EQ (6.9)}$$

The decay factor that solves this equation is a root of a 5^{th} order polynomial, the coefficients of who depend upon λ, I_0, c, W and SSW. In fact, we have already seen the dependence of the decay factor on I_0, c, and W in Section 5.3.3. Unfortunately, the roots of this polynomial can only be found numerically, but if we assume that SSW is negligible, then the solution to EQ (6.9) is

$$a = \lambda \sigma_{in} W. \qquad \text{EQ (6.10)}$$

Thus, the optimal decay factor, i.e., the decay factor that gives zero EB, is only dependent upon the λ, σ_{in}, and W. The value of W is set by the user, and σ can be estimated from the input, the task now is to estimate λ.

Multiplicative Noise

With multiplicative noise the standard deviation of the input noise's distribution is proportional to the intensity of the underlying signal. Thus,

$$\sigma_{in,-} = QI_-$$
$$\sigma_{in,+} = QI_+$$

where Q is a constant of proportionality, and $\sigma^2_{in,-}$ and $\sigma^2_{in,+}$ are the noise variance to the left and right of the edge discontinuity, respectively. Rearranging EQ (6.7) and EQ (6.8) and inserting the above expressions for the noise, we obtain

$$\sigma_- = \frac{\sqrt{(a + I_- W)^2 + I_-^2 SSW}}{(a + I_- W)^2} QI_-, \text{ and}$$

$$\sigma_+ = \frac{\sqrt{(a + I_+ W)^2 + I_+^2 SSW}}{(a + I_+ W)^2} QI_+.$$

From EQ (6.5), EQ (6.6), and these two equations above, if we set $a = 0$, then

$$\mu_- = \frac{I_-}{I_- W} = \frac{1}{W},$$

$$\mu_+ = \frac{I_+}{I_+ W} = \frac{1}{W} \quad \Rightarrow \mu_+ = \mu_-,$$

$$\sigma_- = \frac{QI_- \sqrt{(I_- W)^2 + I_-^2 SSW}}{I_-^2 W^2} = \frac{Q\sqrt{W^2 + SSW}}{W^2},$$

$$\sigma_+ = \frac{QI_+ \sqrt{(I_+ W)^2 + I_+^2 SSW}}{I_+^2 W^2} = \frac{Q\sqrt{W^2 + SSW}}{W^2} \quad \Rightarrow \sigma_+ = \sigma_-.$$

So, choosing $a = 0$ ensures that both the output background intensities and the output noise variances are equal on both sides of the edge discontinuity, the condition needed for zero EB, as desired.

6.3.2 Determining λ for 1-D Edges

We derived above the decay factor that gives zero EB, and showed that it depends on the constant λ. This constant cannot be derived analytically, so in this section we experimentally or empirically[1] determine its value. These λ values are tabulated for the SICNN with the rectangular, triangular, Gaussian, and optimal weight distributions for inputs with different noise and ENR.

To determine the optimal decay factor using EQ (6.10), the input noise strength needs to be estimated. For a 1-D step edge, the noise is first estimated by computing the local noise variance at each pixel of the input using a sliding window of length $2r + 1$, where r is the neighbourhood size. The noise variance for the entire signal is then taken to be the median value of all these local noise variance estimates.

Additive Noise

Figure 6.7 shows the SICNN edge detection performance for four different λ values used to estimate the optimal decay factor for a step edge with additive Gaussian noise. Different λ values result in slightly different EB, but both the HR and ESD are essentially the same. The optimal λ, i.e., the λ that results in near zero EB, is between 2.8 or 2.9 in this case. Also note that the EB is only large for very noisy inputs, i.e., for very low ENR values.

This result suggests that small perturbations to the value of λ have a negligible effect on the performance, particularly the HR and ESD. Figure 6.8 shows the performance for the rectangular, triangular and Gaussian weight distributions, as a function of λ. As λ or the decay factor increases, the HR decreases slightly, while the ESD decreases by about only 1 pixel. Thus, large variations in λ cause only small changes to the edge detection performance.

1. By empirically determining the decay factor, we vary the decay factor until we obtain zero EB by Monte Carlo simulations.

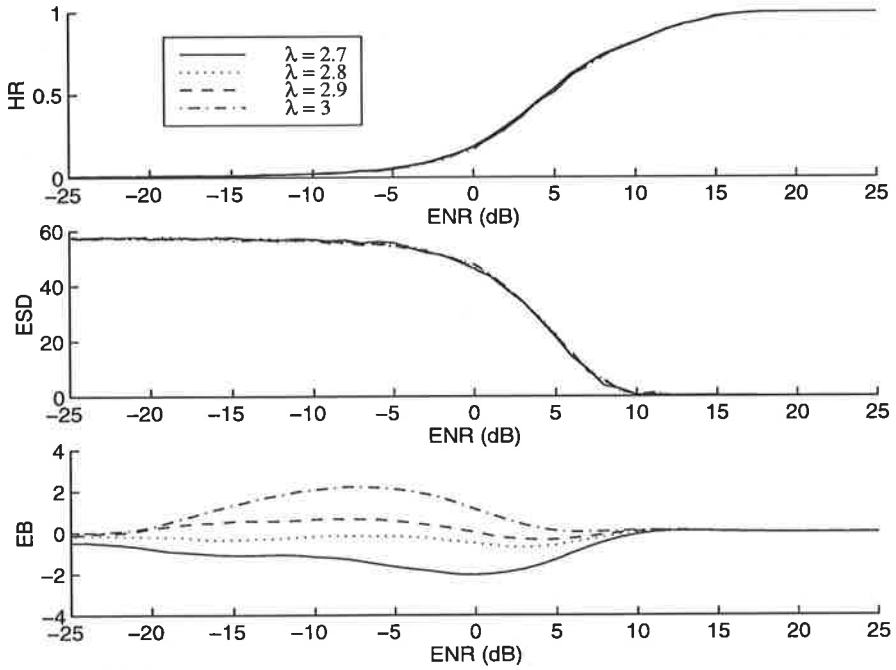


Figure 6.7 SICNN performance when the optimal decay factor is estimated using four values of λ. The SICNN has asymmetrical, rectangular weight distribution, $r = 5$, with $I_o = 10$, $c = 0.25$, and additive Gaussian noise.

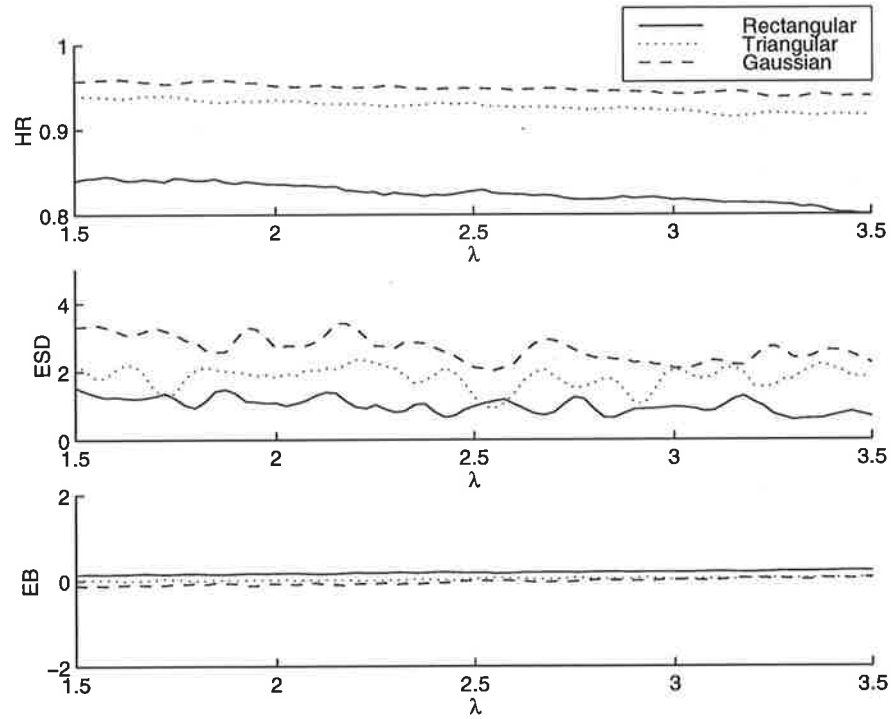


Figure 6.8 SICNN performance when the optimal decay factor is estimated for different values of λ. The SICNN has $r = 5$, with $I_o = 10$, $c = 0.25$, ENR = 10 dB and additive Gaussian noise.

Figure 6.9 shows the edge detection performance for step edges with uniform noise, using the optimal decay factor estimated for four different values of λ. In this particular case, the value of λ which gives zero EB is about 2.3. Again, although the different λ values cause the SICNN to have different EB, both the HR and the ESD are essentially identical.

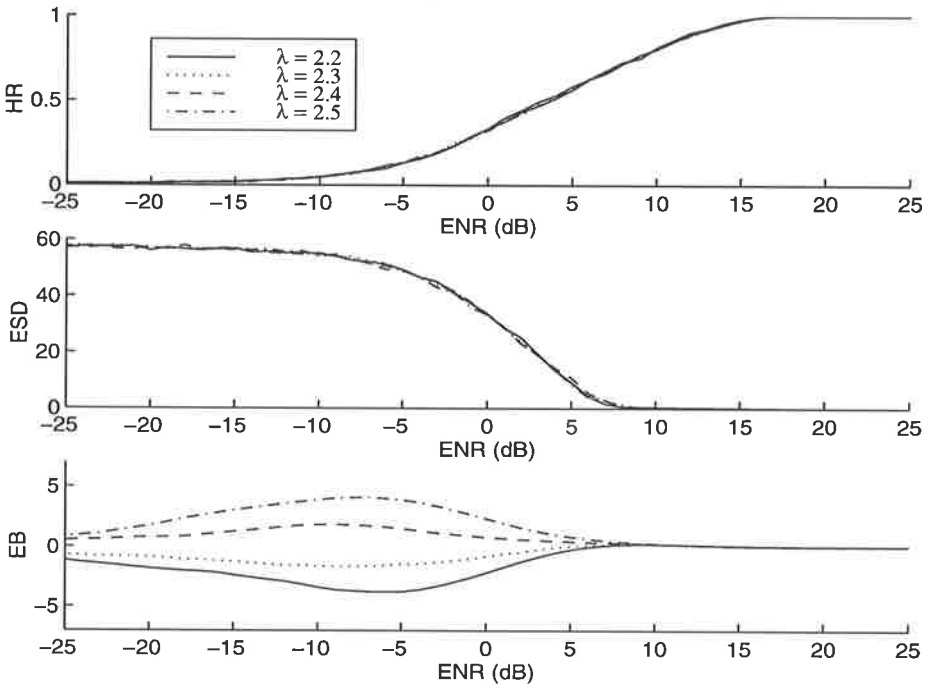


Figure 6.9 SICNN performance when the optimal decay factor is estimated using four values of λ. The SICNN has asymmetrical, rectangular weight distribution, $r = 5$, with $I_o = 10$, $c = 0.25$ and additive uniform noise.

Figure 6.10 shows the performance for the rectangular, triangular and Gaussian weight distributions as a function of λ. The input has uniform noise. Although the EB can vary over a large range, both the HR and ESD change by only very small amounts.

Thus, in this section we have derived a simple equation and method of determining the decay factor from the input signal to give near zero EB for step edges with additive Gaussian and uniform noise. Furthermore, the SICNN edge detection performance is very robust to variations in the value of λ used to estimate the optimal decay factor.

Regardless of this robustness of variations of HR and ESD with respect to λ, we would still like to choose λ so that the EB is as small as possible. Table 6.1 gives the empirical values of λ that result in the smallest EB over a wide range of ENR for the rectangular, triangular and Gaussian weight distributions. Table 6.2 gives the optimal λ values for the

SICNN with the optimal weight distribution with $p = 0.5, 1$ and 2. In each case, the indicated value of λ minimises, in the mean square sense, the aggregate of the EB computed for noisy step edges with ENR varying from -35 dB to 50 dB. This optimal λ value is also given for step edges smoothed with a Gaussian filter of length $2r + 1$. The performance plots from which these values are derived are given in Appendix B.

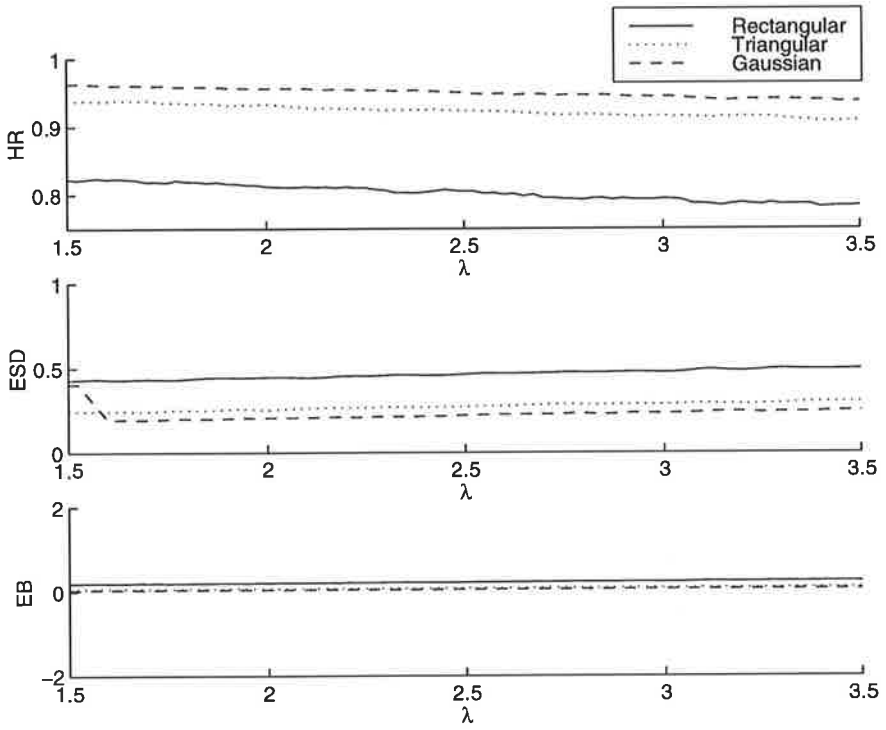


Figure 6.10 SICNN performance when the optimal decay factor is estimated for different values of λ. The SICNN has r = 5, with $l_o = 10$, c = 0.25, ENR = 10 dB and additive uniform noise.

Table 6.1 Optimal λ for rectangular w_R, triangular w_T, and Gaussian w_G weight distributions.

			Gaussian Noise			Uniform Noise		
			w_R	w_T	w_G	w_R	w_T	w_G
No Smoothing	r = 5	c = 0.25	2.8	2.9	3.0	2.3	2.4	2.5
		c = 0.5	2.8	2.9	3.0	2.3	2.4	2.5
	r = 10	c = 0.25	2.7	2.8	2.8	2.1	2.2	2.3
		c = 0.5	2.7	2.7	2.8	2.1	2.2	2.3
With Smoothing	r = 5	c = 0.25	3.1	2.7	2.5	3.0	2.6	2.4
		c = 0.5	3.1	2.7	2.6	3.0	2.6	2.6
	r = 10	c = 0.25	3.0	2.4	2.0	2.7	2.3	2.0
		c = 0.5	3.0	2.4	2.3	2.7	2.3	2.2

Table 6.2 Optimal λ for the optimal weights with attenuation factors 0.5, 1 and 2.

			Gaussian Noise			Uniform Noise		
			p = 0.5	p = 1	p = 2	p = 0.5	p = 1	p = 2
No Smoothing	r = 5	c = 0.25	2.8	2.9	3.1	2.4	2.5	2.7
		c = 0.5	2.8	3	3.1	2.4	2.5	2.7
	r = 10	c = 0.25	2.7	2.8	3.1	2.1	2.3	2.6
		c = 0.5	2.7	2.8	3.1	2.1	2.3	2.6
Smoothing	r = 5	c = 0.25	2.7	2.4	2.2	2.6	2.4	2.1
		c = 0.5	2.7	2.5	2.3	2.7	2.6	2.2
	r = 10	c = 0.25	2.4	1.9	1.3	2.3	1.9	1.3
		c = 0.5	2.4	2.1	1.6	2.4	2.0	1.6

Multiplicative Noise

Figure 6.11 shows the performance on step edges with multiplicative noise when the decay factor is set to zero.Cearly the EB is, in general, less than about 1.5 pixels and often less than 1 pixel. Thus, choosing $a = 0$ appears to be the appropriate choice for zero EB, as predicted from the derivation above. Obviously such a detector is also easy to implement in practice as no noise parameters need to be estimated from the input signal.

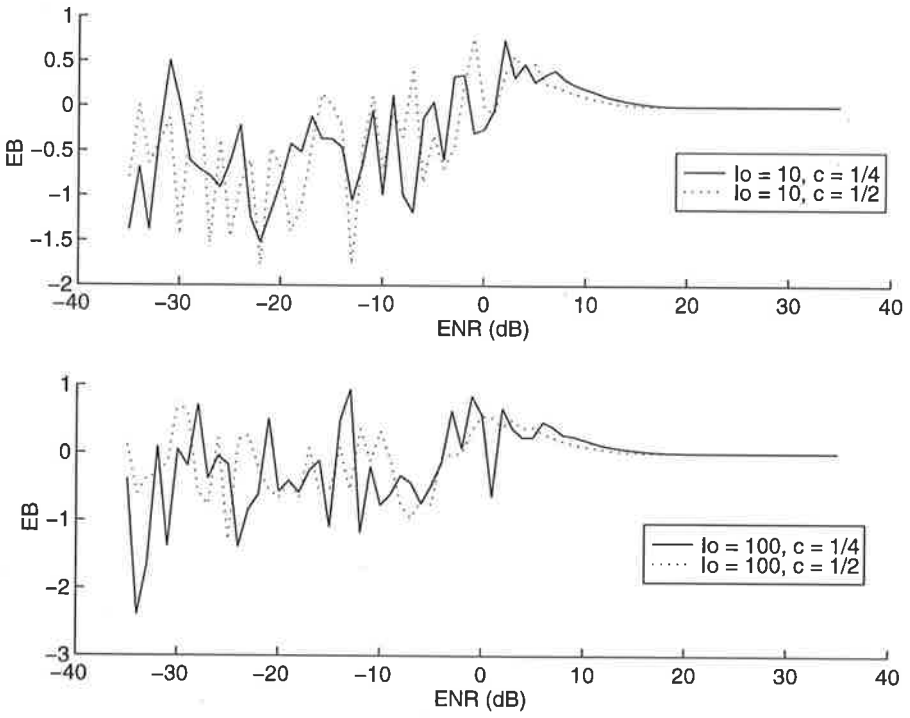


Figure 6.11 SICNN EB for step edges with multiplicative noise. The SICNN an asymmetrical, rectangular weight distribution, r = 5, and a = 0.

These findings should not surprise us since we saw in Section 4.3.1 that the SICNN acts as a nonlinear filter with respect to the input intensity. In particular, from EQ (4.17) or large input intensities, the output noise variance is inversely proportional to the square of the input intensity. Thus, the SICNN has an in-built automatic gain control of sorts. For multiplicative noise, the SICNN tends to "undo" the nonlinearities of the input noise. Later we will see the advantage of this network compared to linear filters for edges with multiplicative noise.

6.3.2.1 Comparison of Optimal Decay Factor Using the λ and PNR Methods

In Section 5.3.3 the decay factor that theoretically maximised the PNR was shown to be close to the empirically optimal decay factor. Although it is easy to solve this equation to obtain the decay factor, the small difference between this value and the optimal one may result in significant suboptimal performance. The "lambda" method of deriving the optimal decay factor as described in this section, however, may be computationally more intensive and not as elegant as the method given in Chapter 5, but it does appear to give a smaller EB over a wide range of ENR.

Figure 6.12 compares the performance with the decay factor from the PNR method, the lambda method with $\lambda = 2.8$, and also using the empirically optimal value. The SICNN with the decay factor estimated using the lambda method has almost identical performance to the SICNN with the empirically optimal decay factor. Both these SICNNs have near zero EB, even for very low ENR. The decay factor estimated from the PNR method gives a smaller HR than either of the other two SICNNs for ENR greater than about 5 dB, and also has a very large negative EB for ENR less than about 5 dB. This EB causes the ESD with the PNR method to be smaller than that of the other two SICNNs, particularly for ENR less than about 5 dB.

6.3.3 Optimal Decay Factor for 2-D Image

The results for the optimal λ for 1-D step edges are extended to the 2-D case by investigating the SICNN performance on the 2-D synthetic image shown in Figure 6.13(a). The image consists of a series of step edges of different contrast. The mean intensity of the entire image is 62, while the contrast averaged over all edges is approximately 0.52. The ENR of the image is computed as per the 1-D edges but with these values of intensity and contrast.

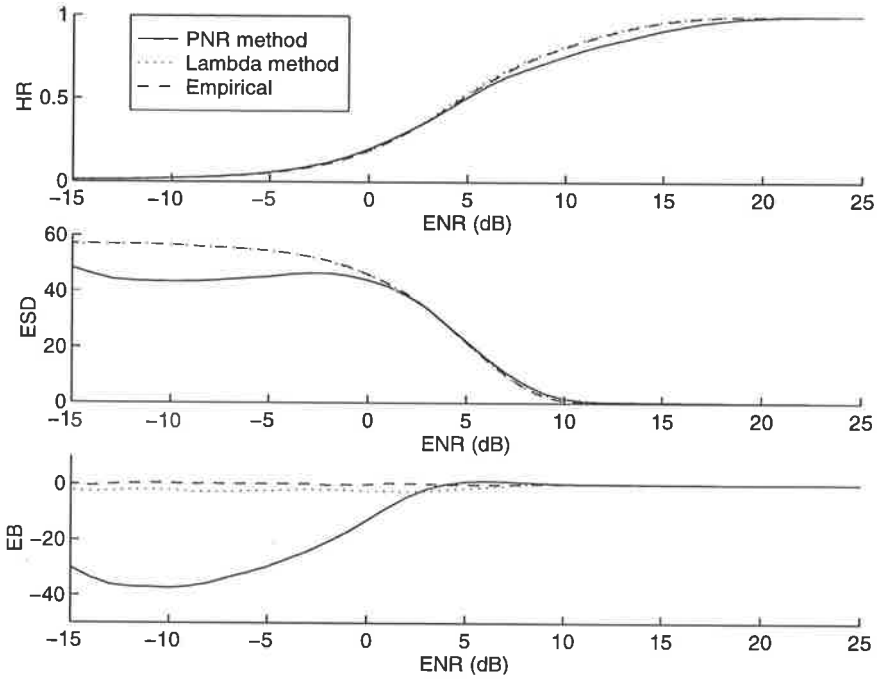


Figure 6.12 SICNN performance with the decay factor computed in three different ways. The SICNN has symmetrical, rectangular weight distribution with $r = 5$, $I_o = 10$, $c = 0.25$ and additive Gaussian noise.

In 2-D, the usual 1-D measures such as the HR, ESD and EB are no longer applicable, so we need a new performance measure. Consider the SICNN output to an input containing one or more edges. If we threshold this output, then all pixels whose intensities are greater than the threshold are declared to be edge pixels. Consider the two hypotheses: H_1, the hypothesis that a SICNN output pixel is an edge pixel, and H_0, the hypothesis that a SICNN output pixel is not an edge pixel. We can then define (Poor, 1988, pp. 31-33):

- ***False alarm (FA) probability or rate*** as the probability that we accept H_1 given that H_0 is true.
- ***Probability of detection (PD) or the detection rate*** is the probability that we accept H_1 given that H_1 is true.

Thus, given that we threshold the output of the SICNN, the FA is the probability that any pixel whose intensity exceeds the threshold is actually a non-edge pixel, while the PD is the probability that the intensity of the true edge pixel in the output will exceed the threshold. Ideally, the FA should be zero and the PD should be one. The PD vs. FA curve can be obtained by varying the threshold and measuring the PD and FA for the output with

each particular threshold. Figure 6.13(b) shows the PD vs. FA curves for two different ENR values.

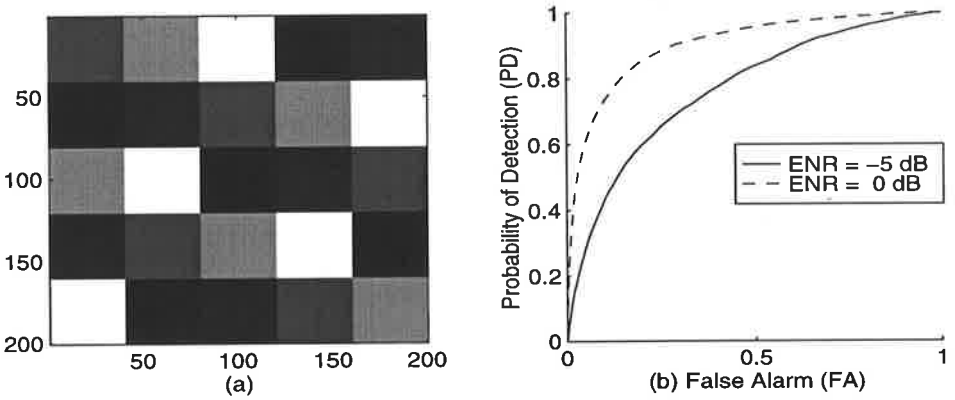


Figure 6.13 (a) The 2-D synthetic image, and (b) shows typical probability of detection vs. false alarm curves for two different ENRs.

In order to compare the 2-D edge detection performance, the area under the PD vs. FA curve for a given ENR is measured, as expressed by EQ (6.11). For a given FA, we would like the PD to be as large as possible, hence curves with larger integrals or areas indicate better edge detection performance[1]. Note that when there is no noise we obtain the maximum possible area of 1, whereas the minimum possible area of 0.5 is obtained when there is so much noise present on the signal that it is equally probable to detect the edge pixel as it is to detect a non-edge pixel.

$$Area = \int_0^1 PDd(FA) \qquad \text{EQ (6.11)}$$

Table 6.3 gives the value of λ which maximises the area under the PD vs. FA curve for a SICNN with the asymmetrical, rectangular weight distribution, while Table 6.4, Table 6.5, and Table 6.6 give the values of λ and also the weight attenuation factor which maximise the area for the SICNN with the asymmetrical, optimal weight distribution. Due to computational considerations it is not possible to test every possible value of λ for the optimal weight distributions (as p must also be varied), hence the performance is only simulated for λ equal to 0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5 and 4. The results are also presented

1. This makes the assumption that the PD vs. FA curves for different ENR do not have irregular shapes, hence the area is a valid reflection of the performance.

for the case when the input is smoothed with a Gaussian filter of length $2r+1$. The relevant Figures used to determine these tables are presented in Appendix B.

Table 6.3 Optimal λ for 2-D synthetic image for the SICNN with the rectangular weight distribution.

		Gaussian Noise				Uniform Noise			
		-5 dB	0 dB	5 dB	10 dB	-5 dB	0 dB	5 dB	10 dB
No Smoothing	r = 5	1.5	1.4	1.0	0.8	1.6	1.5	1.3	1.2
	r = 10	1.0	1.0	0.4	0	1.2	1.0	0.8	0
	r = 15	0.8	0.6	0	0	0.8	0.6	0	0
Smoothing	r = 5	2.4	2.6	1.8	1.6	5	2.8	1.8	1.8
	r = 10	3.0	1.8	1.2	0.8	3.0	1.8	1.4	0
	r = 15	4	1.6	1.0	0	3.0	2.6	1.2	0

Table 6.4 Optimal λ and p for the synthetic 2-D image with multiplicative noise.

		-5 dB		0 dB		5 dB		10 dB	
		p	λ	p	λ	p	λ	p	λ
No Smoothing	r = 5	0	0	0	0	0.7	0	1.6	0
	r = 10	0.4	0	0.55	0	1.2	0	1.9	0
	r = 15	0.5	0	1.0	0	1.3	0	1.5	0
With Smoothing	r = 5	1	0	1.1	0	1.3	0	2.5	1.0
	r = 10	1	0	2.5	0	2	0	1.8	0.5
	r = 15	1	0	3	0	1.5	0	1	0

Table 6.5 Optimal λ and p for the synthetic 2-D image with additive Gaussian noise.

		-5 dB		0 dB		5 dB		10 dB	
		p	λ	p	λ	p	λ	p	λ
No Smoothing	r = 5	1.0	1.5	1.0	1.0	1.0	1.0	1.0	1.0
	r = 10	0.9	1.0	1.0	1.0	1.1	1.0	1.0	0.5
	r = 15	0.6	1.5	1.0	1.0	1.0	1.0	1.1	0.5
With Smoothing	r = 5	0.8	2.5	1.2	2.5	0.2	1.5	0.9	1.5
	r = 10	0	2	0	1.0	0	1.0	1.4	0.5
	r = 15	0	3.0	0	1.5	0	1.0	1	0

Table 6.6 Optimal λ and p for the synthetic 2-D image with additive uniform noise.

		-5 dB		0 dB		5 dB		10 dB	
		p	λ	p	λ	p	λ	p	λ
No Smoothing	r = 5	0.8	1.5	0.8	1.0	1.0	1.0	1.1	1.0
	r = 10	0.9	1.0	0.9	1.0	0.9	1.0	1.1	1.0
	r = 15	0.7	1.5	0.8	1.0	1.1	1.0	1.1	0.5
With Smoothing	r = 5	0.8	3.0	0.8	2.0	1.1	2.0	1.1	1.0
	r = 10	0	2.5	0	2.0	0	1.0	1.2	1.0
	r = 15	1	2	1.1	1.5	1	0.5	1	0.5

6.4 Activation Function

To this point in all our work we have only considered the linear activation function $f(x) = x$. The simplicity of its form not only eases the implementation of the SICNN, but it also greatly simplifies, in many cases, the resulting algebraic equations. By using this function, however, we may be implicitly ignoring other functions which may give better edge detection performance. We now look at two other functions and their effects on the edge detection performance.

The linear activation function, in the biological neural network sense, implies unconstrained firing as the cell's state becomes unconstrained (or infinitely large). A more biologically plausible activation function, and one used more often in neural networks, is the sigmoidal activation function. Here we are particularly interested in the hyperbolic tangent function, $f(x) = \tanh(x/\beta)$ and the saturating exponential function, $f(x) = (1 - \exp(x/\beta))$, where β is a constant related to the slope of the curve at the origin.

For each of these activation functions, we consider three different functions, as shown in Figure 6.14. For each particular function we are interested in one that rapidly reaches saturation, another that reaches saturation extremely slowly, and a third that is somewhat in between these two and similar in shape to the linear activation function. The three different implementations of the functions are produced by varying the slope parameter β. Here, β equal 1, 10 and 100.

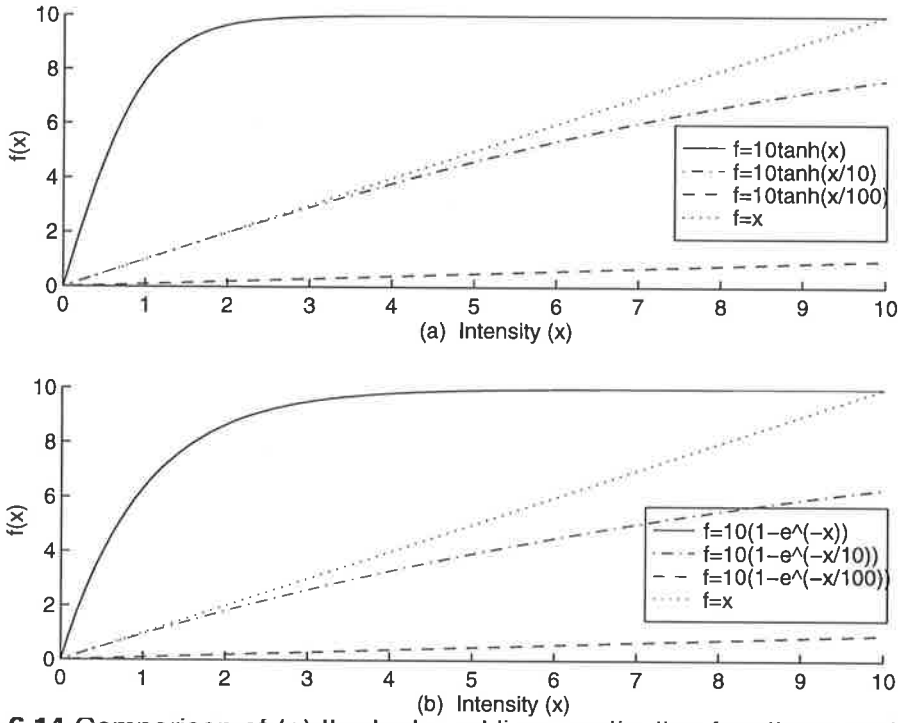


Figure 6.14 Comparison of (a) the tanh and linear activation functions, and (b) the saturating exponential and linear activation functions. In both cases $\beta = 1, 10$ and 100.

6.4.1 The Saturating Exponential Function

Consider first the function $f(x) = (1 - \exp(-x/\beta))$. The SICNN performance with this activation function compared to the linear activation function is shown in Figure 6.15. The decay factor for each SICNN is varied until the smallest possible edge bias is achieved. The SICNN with the linear activation function clearly produces the best results when compared to the performance using the other three functions. Only the performance with $\beta = 100$ is comparable, though still inferior. The results for $\beta = 1, 10$ are so bad that the best possible EB is about 50, resulting in large ESD and zero HR. So the performance with the saturating function is clearly inferior to that with the linear function. The performance also appears to vary significantly with β; a change of one order of magnitude in β causes the performance to change from good to very bad. We also found that the performance curves vary with the mean input intensity, though the performance is still inferior to the SICNN with the linear activation function.

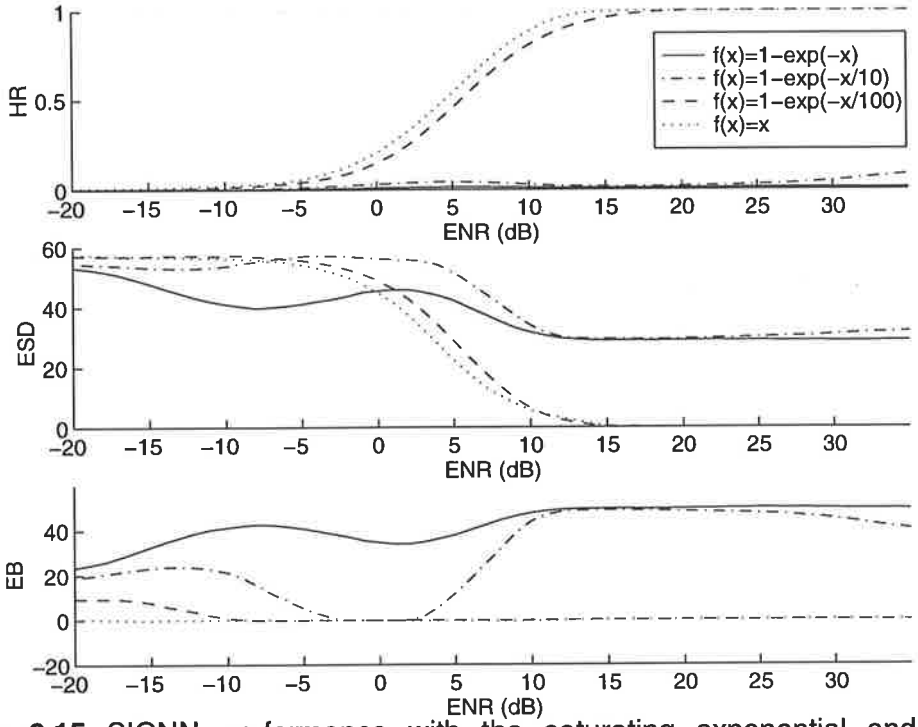


Figure 6.15 SICNN performance with the saturating exponential and linear activation functions. The SICNN has asymmetrical, rectangular weights with $r = 5$, $I_o = 10$, $c = 0.25$ and additive Gaussian noise.

6.4.2 The Hyperbolic Tangent Function

Now we investigate the performance of the hyperbolic tangent function, $\tanh(x/\beta)$. Again three different functions were implemented by varying the value of β. Figure 6.16 shows the results for an input edge with additive Gaussian noise. Once again the edge detection performance of the SICNN with linear activation function is the best of all those considered. The performance with the tanh function with $\beta = 100$ is relatively close, but still inferior. The results with the other two functions, $\beta = 1, 10$, are both very bad with large EB, even for large ENR. This results in a large ESD and zero HR, both of which are bad. Once again the performance of the SICNN with the tanh activation function depends critically on the value of β. Small variations in β can result in large changes in the performance, which is highly undesirable. As with the saturating exponential activation function, the performance curves vary with the mean intensity, though they are still inferior to that of the SICNN with the linear activation function.

In this subsection, we have investigated two different activation functions, both of which exhibit saturation. The SICNN performance is clearly inferior to that of the SICNN with the linear activation function. The performance of these SICNNs also varies greatly with

the value of β, implying that the activation function must be chosen carefully to obtain good results, results which are still not as good as those obtained with the linear function.

Another advantage of the linear activation function is that the derived mathematical expressions are relatively simple, which would not be the case were we to use a nonlinear activation function. We have also seen that using a linear function makes the selection of certain SICNN parameters, such as the decay factor, very easy and straightforward to compute.

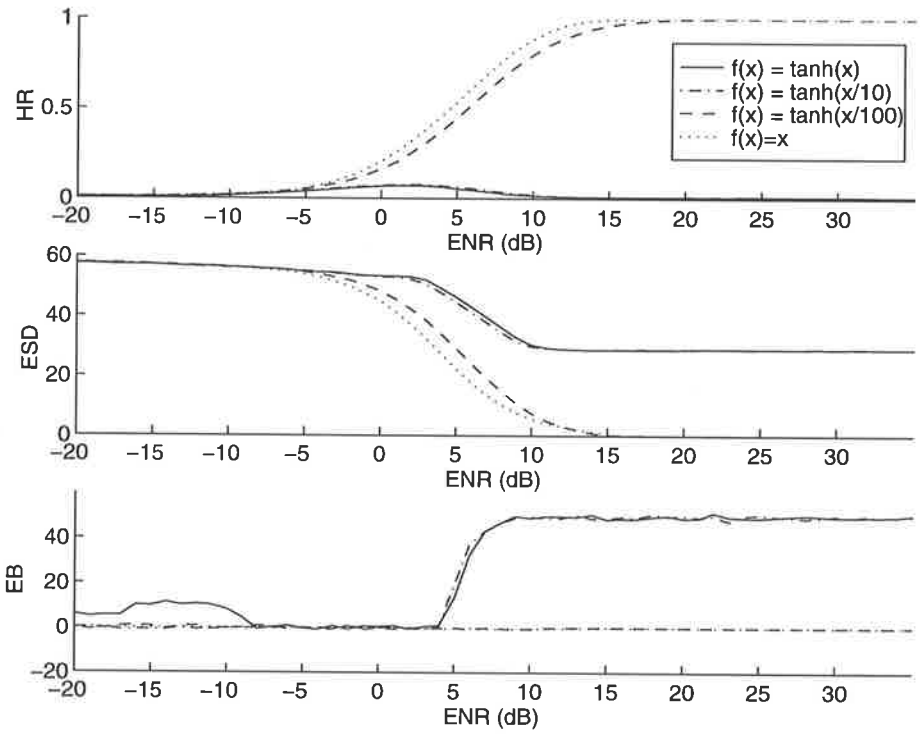


Figure 6.16 SICNN performance with the tanh and linear activation functions. The SICNN has asymmetrical rectangular weights with $r = 5$, $I_o = 10$, $c = 0.25$ and additive Gaussian noise.

6.5 Conclusions

This Chapter has provided algorithms to determine the SICNN parameters which optimise both its 1-D and 2-D edge detection performance. We began with the weight distribution, and using constrained optimisation derived the weight distribution that simultaneously optimises the HR and ESD. The shape of this optimal weight distribution can vary, but is somewhat in between that of the rectangular and Gaussian weight distributions. The shape

of this optimal weight distribution is varied using the weight attenuation factor, thus allowing us to trade-off good HR for good ESD.

Next we derived the decay factor which gives zero EB in the 1-D edge detection process. For multiplicative noise, the optimal decay factor is zero. For additive noise, the optimal decay factor was shown to be proportional to the product of the noise intensity present on the input signal and the sum of weights. The value of this constant of proportionality λ, computed by numerical simulations, was presented for both 1-D and 2-D synthetic edges and for different weight distributions. The optimal value of λ was also found for the optimal weight distribution with various values of the weight attenuation factor. The SICNN performance with the decay factor estimated with this method had comparable performance to that of the SICNN with the empirically optimal decay factor. Both of these decay factors gave better performance than the decay factor estimated using the PNR method from Chapter 5.

Finally, we briefly investigated the performance of the SICNN with nonlinear activation functions, namely the saturating exponential and hyperbolic tangent functions. The SICNN with either of these two weight distributions performed very badly compared to the linear activation function. In fact, in many cases it was not possible to select the decay factor to ensure zero bias. Given that these functions are nonlinear, and that they give poor performance, for the remainder of our work we use the linear activation function.

Chapter 7 Postprocessing Methods for Edge Detection

7.1 Introduction

In this chapter we investigate ways in which the SICNN output can be processed to improve its edge detection performance. For different SICNN parameters, different outputs are obtained, and if these outputs are appropriately combined, then the overall performance can be improved. Some of the methods discussed here are not only relevant to the SICNN, but can also be applied to other edge detectors.

We begin the investigation of scale-combination postprocessing by first briefly reviewing the scale-space literature and showing its relevance to the SICNN. We already know that the SICNN has different edge detection performance depending upon its scale, or neighbourhood size. Indeed, this is also true of most edge detector operators. Large scale operators have good noise reduction properties, whereas small scale operators have good resolution. Thus, by combing them both good resolution and good noise-reduction can be achieved. An algorithm is presented for tracking the edges in the SICNN output or both 1-D and 2-D synthetic images, as its neighbourhood size is gradually and continually decreased.

In a similar manner, the SICNN output, and hence its edge detection performance, varies as a function of the shape of its weight distribution. We investigate tracking the edges in the output as the weight distribution's shape is varied from a very broad one to a very narrow one. Again, for a broad weight distribution, the SICNN has good noise reduction properties, whereas for a narrow distribution it has good resolution. By tracking the edges as the weight distribution is varied, both high accuracy and good noise reduction

properties can be obtained. The optimal weight distribution with varying weight attenuation factor is used, and results are presented for both 1-D and 2-D synthetic edges.

Next we investigate combining the outputs for different weight distributions and thresholding. The SICNN can have either symmetrical or asymmetrical weights, and the edges in either output can be detected by either maximum or zero-crossing thresholding. The increase in the edge detection performance by combining various outputs is investigated.

In Section 7.5 we investigate the increase in the edge detection performance by combining the outputs of SICNNs with identical, but reversed in direction, weight distributions. The edge responses of such SICNNs have opposite polarity compared to the background mean intensity, hence they can be appropriately combined to produce an edge response which is greater than either of the two edge responses alone.

Finally, a simple postprocessing technique which can be used with most edge detectors is investigated. The performance can be greatly improved by identifying and eliminating edge pixels in the output which are isolated from the other edge pixels, or have reasonably few neighbouring edge pixels. Such isolated edge pixels are most likely to arise from noise, and hence can be removed.

7.2 Scale Combination

All objects in the world only exist as meaningful entities over a certain range of scales (or sizes). For example, atoms viewed at a larger scale are water molecules, and at a yet still larger scale may form a cloud. Thus, the object that we observe is intrinsically related to the scale at which we observe it. For real images, edges can exist at many different scales. Scale-space processing can represent the multiscale nature of the edge detection problem.

We saw in Section 5.4.3 that varying the neighbourhood size, or scale, varies its edge detection performance. For 1-D step edges, a SICNN with a large scale has good noise suppression ability, which results in a low ESD. This good localisation, however, comes at the expense of poor resolution, i.e., the HR is low. As the scale of the SICNN is decreased, both the HR and ESD increase.

The basic procedure with scale-combination is to first apply a SICNN with a large scale to the input in order to achieve the best possible edge localisation (at the expense of resolution). Once the edges are located, they are tracked in the output as the scale is gradually decreased. By reducing the scale, the resolution of the edge detection process increases, but we retain the good edge localisation. Thus, this procedure provides a compromise between good resolution and localisation.

7.2.1 Scale-Space Processing with the Laplacian-of-Gaussian

An object in an image can appear as different entities depending upon the scale used to view the object. Consider a single pixel of a real image. If we view it and its immediate neighbours we may only see a short noisy segment. Viewing the pixel at a larger scale (more nieghbouring pixels) we may be able to identify an edge discontinuity with added noise, and at a still larger scale the edge discontinuity and ultimately the original pixel, may form part of a man-made or real object in the image. Thus, the meaning of any object is inherently dependent upon the scale at which it is viewed. Scale-space processing allows us to deal with inherent property of any measured data or object.

Scale-space has developed immensely in the last few years, but we will only review a few of the more important results. For an in-depth review of scale-space processing, the reader is referred to Lindeberg (1994).

Bergholm (1987) was one of the first researchers to apply the basic principle of scale-space to the problem of *edge tracking* or *focusing,* i.e., a coarse-to-fine tracking of edges in a continuous manner. He noted that every edge detector has a built-in conflict, that of achieving high accuracy or resolution while still removing noise, i.e., simultaneously making the HR large and the ESD small.

In his edge focusing algorithm, Bergholm begins by smoothing the input with the coarsest possible Gaussian filter (large scale) and then performs edge detection. A threshold is applied to detect the edges, and then the process is repeated at a smaller scale in only local windows about the previously detected edges. Thus, the edge focusing algorithm attempts to reverse the effects of the initial blurring without increasing noise.

Goshtasby (1994), following on from the work of Bergholm and others, proposed an algorithm to accurately track edges from the Laplacian-of-Gaussian operator, from low to

high resolution. The algorithm is very robust, and can even track edges which may split or merge during the scale-space process. It also allows large scale step sizes to be taken when moving from low to high resolution, which increases the speed of the edge-focusing algorithm.

Lindeberg (1994) provides a rigorous treatment of scale-space filtering. In particular, he gives the conditions necessary for a discrete filter, or kernel, to be a scale-space kernel. In 1-D, he defines a scale-space kernel as one where, for an input signal f_{in}, the number of extrema in $f_{out} = f_{in} * h$ does not exceed the number of local extrema in f_{in}. Furthermore, it must satisfy (Lindeberg, 1984):

$$\sum_{n=-\infty}^{\infty} h(n) z^n = cz^k e^{(q_{-1}z^{-1} + q_1 z)} \prod_{i=1}^{\infty} \frac{(1+\alpha_i z)(1+\delta_i z^{-1})}{(1-\beta_i z)(1-\gamma_i z^{-1})}$$

where $c > 0$, $k \in Z$, $q_{-1}, q_1, \alpha_i, \beta_i, \gamma_i, \delta_i \geq 0$, $\beta_i, \gamma_i < 1$ and $\sum_{i=1}^{\infty} (\alpha_i + \beta_i + \gamma_i + \delta_i) < \infty$. This results in 5 primitive functions:

- two-point weighted average, or generalised binomial smoothing

$$f_{out} = f_{in}(x) + \alpha_i f_{in}(x-1) \qquad (\alpha_i \geq 0),$$

$$f_{out}(x) = f_{in}(x) + \delta_i f_{in}(x+1) \qquad (\delta_i \geq 0),$$

- moving average or first-order recursive filtering

$$f_{out}(x) = f_{in}(x) + \beta_i f_{out}(x-1) \qquad (0 \leq \beta_i \leq 1),$$

$$f_{out}(x) = f_{in}(x) + \gamma_i f_{out}(x+1) \qquad (0 \leq \gamma_i \leq 1),$$

- infinitesimal smoothing, rescaling, and translation.

These requirements, along with a number of others, imply that the only smoothing function that can be used for scale-space processing is the Gaussian. The application of this theory to SICNN scale-processing is discussed in the next section.

7.2.2 Scale Space Processing with SICNN

7.2.2.1 One Dimension

We begin by investigating scale-space combination for a 1-D step edge. The basic approach is to first apply a SICNN with a large neighbourhood size or scale, and then re-apply to the same input the SICNN with continually decreasing neighbourhood size. For each step in this process, the maximum output for a given neighbourhood size is used to determine where the edge can lie in the following output, i.e., the output of a SICNN whose scale is slightly smaller than the previously applied SICNN. By this method, a significant performance increase, in terms of both HR and ESD, can be achieved over the ordinary SICNN.

On the surface, it may appear that the scale-space process described here for the SICNN is very similar to the paradigm discussed above by Bergholm (1989) and Lindeberg (1994). However, a number of significant differences exist. Scale-space involves applying the Gaussian filter to the input, and then tracking the edges as the scale of this filter is reduced. In our approach, we do not necessarily smooth the input with a Gaussian. For the SICNN, the neighbourhood size is reduced without necessarily changing the shape of the weight distribution. Furthermore, the shape of this weight distribution need not be Gaussian. Another significant difference is that the edges are detected as the maxima in the output, rather than the zero-crossings. So, it is obvious that the SICNN does not satisfy the strict conditions to be a scale-space filter as defined by Lindeberg (1994), however, tracking the edges in the output does lead to significant performance improvement, and is still a valid postprocessing technique to use.

The 1-D scale-combination algorithm is:

1. Perform edge detection using a SICNN with the largest desired scale or neighbourhood size.
2. Find the maximum output of this SICNN (assuming asymmetrical weights).
3. Define a local "valid region" about the edge pixel. The edge in the following steps can only be found in the valid region.
4. Reduce the scale or neighbourhood size and re-apply the SICNN with this new neighbourhood size to the original input.
5. Find the maximum output of this SICNN within the valid region.

6. Define a new valid region about this new maximum output.
7. Repeat steps 4 to 6, until the smallest desired scale has been applied to the input.

It was found that selecting only the largest output to define the valid region in the above algorithm, gives the best results. When the two (or more) largest outputs are used, the performance deteriorates markedly. The postprocessing performance for different number of maxima is given in Appendix C.

In Figure 7.1, the performance both with and without scale-combination is shown, for a step edge input with Gaussian noise; further results for different ranges of scales combined and different noise types can be found in Appendix C.

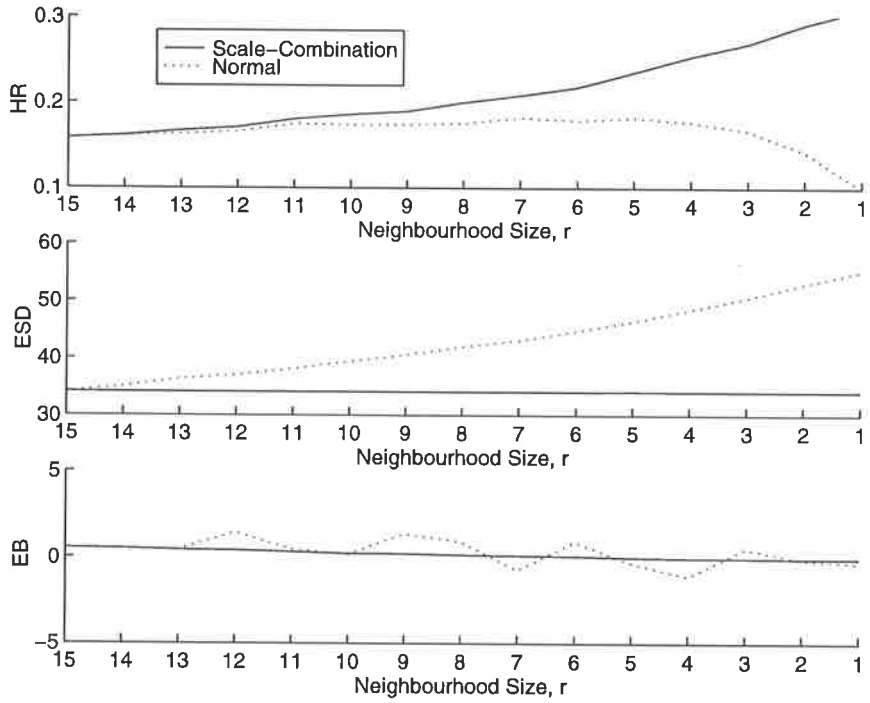


Figure 7.1 SICNN performance with and without scale-combination as the scale is reduced from 15 to 1. An asymmetrical, rectangular weight distribution, with optimal decay factor, $I_0 = 10$, $c = 0.25$, $ENR = 0dB$, and Gaussian noise.

This Figure shows that, as the scale is reduced from 15 down to 1, the HR of both SICNNs with and without scale-combination, increase slightly. The HR of the normal SICNN[1] output, however, decreases towards zero for very small scales, while the HR with scale combination continues to rise to a value of just over 0.3. The maximum HR of the normal

1. The normal SICNN has no postprocessing of its output.

SICNN is less than 0.2. The ESD of the combined SICNN remains constant, while the ESD of the normal SICNN continues to increase as the scale decreases down to 1. Clearly, significant performance improvement can be achieved by combining the outputs of different scales. In particular, the HR can be improved without sacrificing the low ESD. Similar improvements to the performance through scale-combination for inputs with multiplicative and uniform noise are evident and are presented in Appendix C.

7.2.2.2 Two Dimensions

For the 2-D scale-combination processing we investigate the performance of the algorithm on the synthetic image shown Figure 7.2. The 2-D scale-combination algorithm is very similar to the 1-D algorithm, but it differs in a number of ways. In 2-D, edge detection is performed in each of the image's four orthogonal directions (with the weight distribution appropriately chosen), and no limit is imposed on the number of detected edges in each output, rather the number of edges is determined by the arbitrary threshold. Local valid regions are defined about each edge, which are then used to determine where the edges can lie when the following SICNN is applied to the input image, i.e., the SICNN with a scale, or neighbourhood size, slightly smaller than that of the previously applied SICNN.

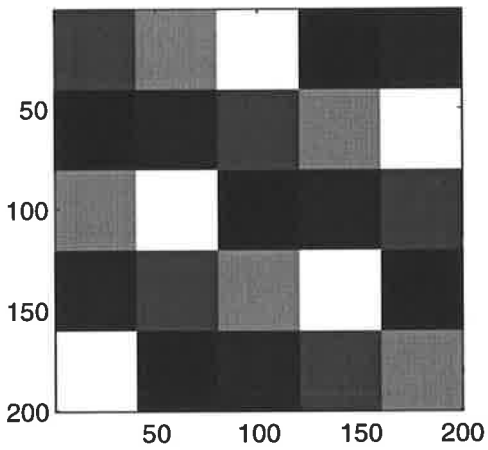


Figure 7.2 Synthetic image with mean intensity 62 and mean contrast 0.52.

Another difference is that, rather than measuring the performance in terms of the HR, ESD and EB, we measure the area under the false alarm (FA) rate vs. probability of detection (PD) curves as discussed in Section 6.3.3. As in the 1-D scale-combination case, we begin the algorithm by applying a SICNN with the largest desirable scale - this ensures that the edges in the output are well localised. Of course, with a large scale the resolution of the

output will also be poor, but this improves as the scale gradually decreases. For the 2-D case, we generally expect the area under the FA vs. PD curves to increase as we decrease the scale. Increasing area implies improving performance. Even if this area does not increase we can still expect the output of the SICNN with scale-combination to appear better than the normal SICNN output, due to the fact that at the termination of the algorithm a small scale SICNN is applied to the input, thus the output will appear more detailed than the output of a large scale SICNN. Thus, tracking the edges in the output as the scale is reduced, ensures that the final output has good noise suppression, like for the output with large scales, and good localisation like the output for small scales.

The scale-combination algorithm for each of the SICNNs applied to one of the image's four orthogonal directions is:

1. Apply the SICNN with the largest desirable scale to the input.
2. Apply a threshold to the output to detect the edges, and then define "valid regions" about each of these edge points.
3. Reduce the scale of the SICNN and re-apply this SICNN to the original input.
4. Apply a threshold to this new SICNN output to detect any edges which lie in the previously defined valid region.
5. Define new valid regions about these edge pixels and return to step 3 until the SICNN with the smallest desired scale has been applied to the input.

Figure 7.3 shows the performance of this algorithm when the input synthetic image has Gaussian noise of $ENR = 5$ dB. The SICNN outputs are combined for scales decreasing from (a) 5 down to 1, (b) 10 down to 1, and (c) 15 down to 1. The normal SICNN output is similar to the combined output for large scales, and as the scale decreases to a low value the area for both SICNNs slowly decreases too. The difference in performance between the SICNNs, however, increases as the scale decreases. Further plots for different ENR and noise types can be found in Appendix C.

Although the absolute numerical difference in the performance of both SICNNs is not large, and the area has not increased during the combination process, the difference in the appearance of the output edge maps is very noticeable. Figure 7.4 shows the edge maps for the output when $r = 10$ (no combination), $r = 1$ (no combination), and $r = 1$ (after

the scale combination process for scales from 10 down to 1). For the $r = 10$ output with no scale-combination, most of the true edges are found, but some of these are quite thick, i.e., not a single pixel wide. On the other hand, the output of the $r = 1$ SICNN with no scale-combination is quite noisy due to the small scale. Also shown is the output for $r = 1$ after applying the scale-combination algorithm. Clearly, this algorithm manages to retain most of the valid edges, but it also removes much of the spurious edges. Thus, the output after scale-combination is better than either output with $r = 10$ or $r = 1$ with no scale-combination.

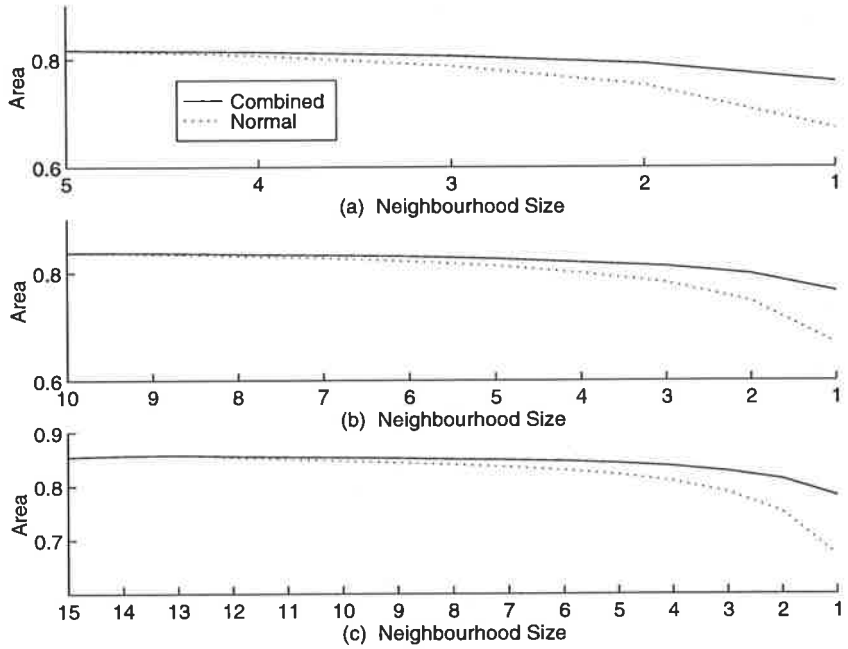


Figure 7.3 SICNN 2-D performance with and without scale-combination for scales from (a) 5 down to 1, (b) 10 down to 1, and (c) 15 down to 1. For each scale, the SICNN has an asymmetrical, optimal weight distribution with the weight attenuation factor chosen to give maximum performance.

Another measure of the amount of improvement in the output after using the scale-combination process is the Noise-to-Signal Ratio (NSR), as defined in Section 2.4.2. It is the ratio of the sum of the total number of false positive edges (FPE) and false negative edges (FNE), to the total number of true positive edges (TPE). For convenience the NSR is repeated here,

$$NSR = \frac{FPE + FNE}{TPE}. \qquad \text{EQ (7.1)}$$

The smaller this value is, the better is the edge detection process. The *NSR* is computed for the outputs shown in Figure 7.4. In (a) for the $r = 10$ output we have $NSR = 3.15$, for the $r = 1$ output in (b) $NSR = 5.60$, while for the postprocessed output in (c), $NSR = 2.11$. These NSR values provide objective evidence of the improvement in the quality of the output edge maps with scale-combination processing.

More convincing evidence of the improvement of the quality of the output edge map is evident for multiplicative noise, as shown in Figure 7.5. For the $r = 10$ output in (a), $NSR = 1.13$, for the $r = 1$ output in (b) $NSR = 1.25$, while for the output after combining the outputs for scales from 10 to 1 in (c) $NSR = 0.38$. There is clearly a significant improvement, which is also evident when inspecting the images.

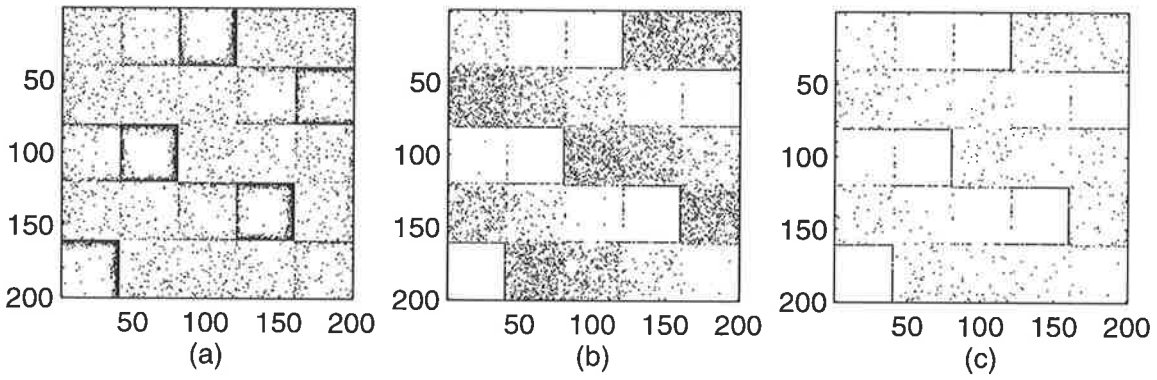


Figure 7.4 SICNN output for (a) $r = 10$ with no combination, (b) $r = 1$ with no combination, and (c) with scale-combination for scales from 10 down to 1. For each SICNN both the optimal weight distribution and decay factor are used. The NSR is (a) 3.15, (b) 5.60, and (c) 2.11. The noise is Gaussian with $ENR = 5$ dB

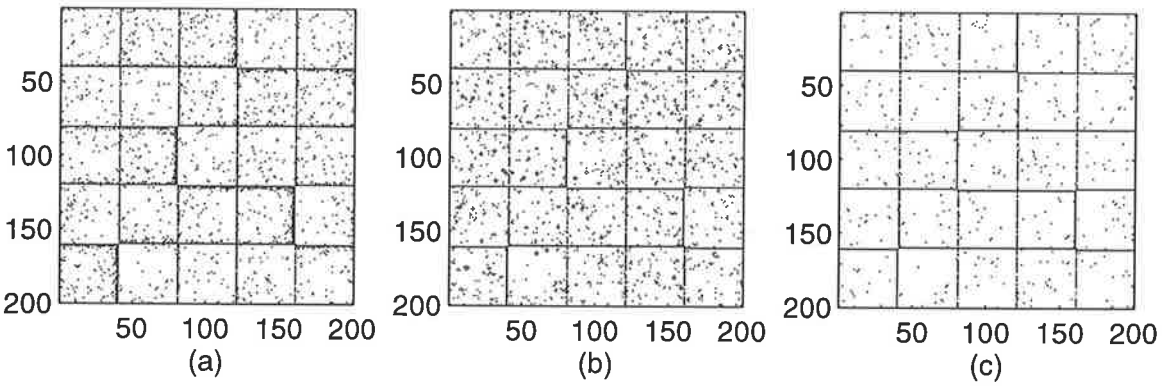


Figure 7.5 SICNN output for (a) $r = 10$ with no combination, (b) $r = 1$ with no combination, and (c) with scale-combination for scales from 10 down to 1. For each SICNN both the optimal weight distribution and decay factor are used. The NSR is (a) 1.13, (b) 1.25, and (c) 0.38. The noise is multiplicative with $ENR = 10$ dB.

7.3 Combination with Variable Weight Distribution

We saw in the previous section how the SICNN performance can be improved by the combination of the outputs with different neighbourhood sizes, or scales. In that case, the weight distribution for each SICNN was the same. Now we look at a slightly different approach to postprocessing - that of combining the outputs of SICNNs of the same neighbourhood size, but with different weight shapes. In Chapter 5 we showed how the shape of the weight distribution affects the performance. For 1-D signals, a very "broad" or flat weight distribution, such as the rectangular one, results in both low ESD and low HR. Also, making the weight's shape or edge response narrower causes both the HR and ESD to increase. This is also evident in Section 6.2.2, where changing the optimal weight distribution's weight attenuation factor causes the width of the edge response to vary, with a corresponding variation to the performance.

Thus, the basic procedure is to apply a SICNN with the broadest possible weight distribution, which ensures that the ESD is as small as possible. The detected edges are used to track edges in the output as the weight's shape is gradually made narrower. By narrowing the weight's shape, the HR of the SICNN increases, and using information about the edge position from the previously applied SICNN (with slightly broader weight distribution), the ESD can be maintained at a low value.

We begin by looking at the performance increase with the optimal weight distribution as the weight attenuation factor is varied, and then investigate the corresponding performance increase with a 2-D synthetic input image.

7.3.1 One Dimension

We saw in Section 6.2.2 that varying the optimal weight distribution's attenuation factor (p) varies both the shape of the distribution and consequently affects the edge detection performance. The SICNN with the optimal weight distribution with a small value of p has a small ESD, but also a small HR, whereas a SICNN with the optimal weight distribution with a large value of p has a large HR, but a large ESD too.

As with scale-combination described in the previous section, by combining the outputs with different optimal weight shapes, a large HR can be achieved whilst maintaining a low ESD. The following algorithm begins by applying a SICNN with the optimal weight

distribution with $p = 0$ (to achieve low ESD). The maximum output is then located, and hence determines where the edge from the following output can be found, i.e., the output of the SICNN with the optimal weight distribution with slightly larger value of p. The constraint of the region where the edge can be located maintains the ESD at its low value during the entire combination process, and by incrementally decreasing p, the HR during the combination process is slowly increased. Thus, the combination process should increase the HR, and maintain the ESD at a low value.

The algorithm is:

1. Choose the "broadest" possible weight distribution (e.g. rectangle or optimal with weight attenuation factor $p = 0$) and perform edge detection with the SICNN.
2. Find the maximum output.
3. Define a "valid region" about this pixel (the edge in the following SICNN output can only be found in this region).
4. Make the shape of the weight distribution slightly narrower (reduce the weight attenuation factor for the optimal weight distribution) and re-apply the SICNN with this new weight distribution to the input.
5. Find the maximum in this output which lies in the valid region of the previous output, i.e., the SICNN whose weight shape was slightly broader than that of the weight distribution used in step 4).
6. Repeat steps 4 and 5 until the weight distribution is as narrow as desired.

In the above algorithm, selecting the largest output in order to define the valid region of step 3 gives the best results. These results and those for selecting a larger number of output pixels to define the valid region, are presented in Appendix C.

Figure 7.6 shows the performance with and without the combination of the outputs of the SICNN with varying weight shape. Note that the combination process begins with a weight attenuation factor equals to zero, i.e., a rectangular weight distribution, which ensures that the ESD is as low as possible to begin with.

The ESD of the normal SICNN increases as the weight attenuation factor increases, but the ESD of the SICNN with the combined output remains approximately constant. Also,

the HR of the normal SICNN also decreases, but the HR of the SICNN with the combined output actually increases slightly for values of p less than about 2, and then levels off.

As with the scale combination algorithm, by combining the outputs with different weight shapes, we are able to increase the HR whilst maintaining a fixed ESD, which is not possible with the normal SICNN output. The improvement in the performance of the SICNN with combination over the normal SICNN is also evident for the inputs with both multiplicative and uniform noise models; these results are present in Appendix C.

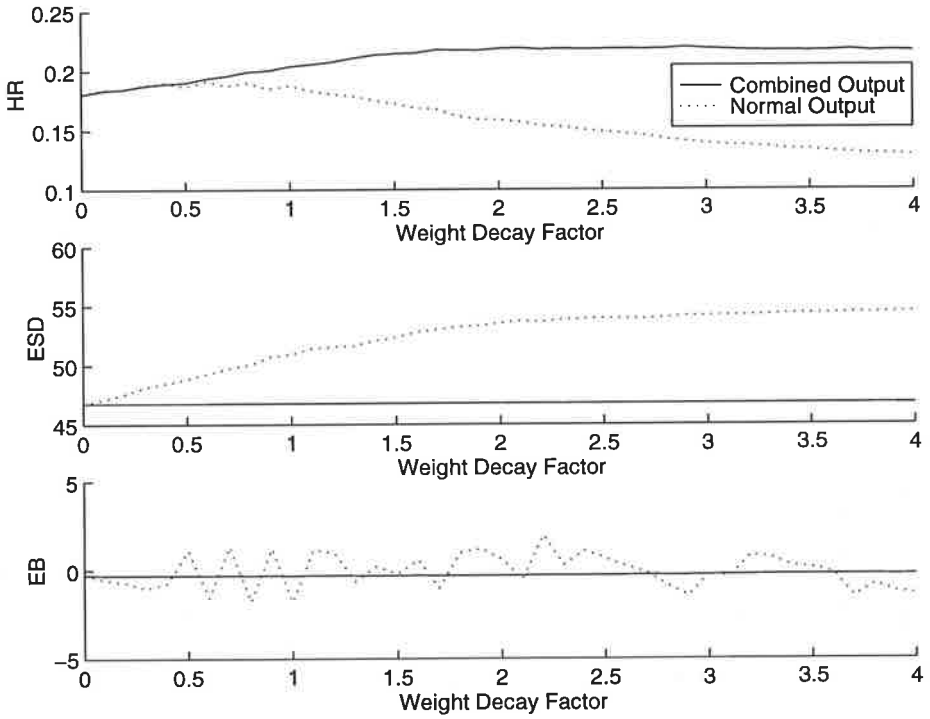


Figure 7.6 SICNN performance with and without combination of the outputs as a function of the attenuation factor. The decay factor is chosen to give near zero EB, and the input has $I_o = 10$, $c = 0.25$, $ENR = 0dB$, with Gaussian noise.

7.3.2 Two Dimensions

As with the 2-D scale-combination algorithm, we investigate the 2-D combination algorithm with varying weight distribution for the standard synthetic image. The performance is again measured in terms of the area under the PD vs. FA curve. Also, four SICNNs are applied to the image, one to each of the four orthogonal directions, with the weight distribution appropriately chosen.

The algorithm begins with the broadest possible weight distribution (with $p = 0$ for the optimal weight distribution), which ensures that all edges in the output are well localised.

The edges are detected by applying an arbitrary threshold to the output. A small local valid region is defined about each edge pixel. A SICNN with a slightly narrower weight distribution, i.e., slightly larger p, is then applied to the same input, and the edges are detected. This time, however, the new edges can only lie in the valid regions as defined by previous output. The new edges in turn define new valid regions, which determine where the edges can be found in the following output. This process repeats until the SICNN with the narrowest desired weight distribution has been applied to the input. By reducing p, the resolution of the detected edges increase, and by tracking the edges good localisation of the detected edges is maintained.

Thus, the algorithm for each SICNN applied to one of the four orthogonal directions is:

1. Apply the SICNN with the broadest desired weight distribution ($p = 0$).
2. Apply a threshold to detect edges in the output, then define a valid region about each edge pixel.
3. Make the weight distribution slightly narrower, and re-apply this SICNN to the input.
4. Apply the threshold, and detect the new edges which lie in the valid region.
5. Define new valid regions about these new edges and go back to step 3 until the SICNN with the narrowest desired weight distribution has been applied to the input.

Figure 7.7 shows the 2-D performance results for an $r = 5, 10,$ and 15 SICNN whose outputs are combined as the weight distribution's shape varies. The image has Gaussian noise of $ENR = 5$ dB. The weight attenuation factor for the optimal weight distribution is varied from 0 up to 4. For p beyond about 1.25, the performance of the SICNN with the combination of the outputs outperforms the SICNN with no combination of outputs, although for low p the reverse is true. As p increases from zero, the performance of the normal SICNN improves quickly, but the performance with postprocessing does not improve as quickly since the location of the detected edges are constrained by the valid regions of the previous combined output. Conversely for large p, the performance of the normal SICNN decreases rapidly, whereas the performance of the combined output decreases gradually since, once again, the position of the detected edges are constrained by the valid region of the previous combined output. Further plots for different ENRs and noise are given in Appendix C.

Although the numerical difference in the performance between the two outputs with and without postprocessing is not large, the appearance of the edge maps is significantly different. Figure 7.8 compares the 2-D edge maps of the SICNN both with, and without the output combination. Figure 7.8(a) shows the output of the SICNN with the optimal weight distribution with $p = 0$. Many of the true edges are detected, but they are quite thick and there is also noise present. The reason for this is that, for this value of p, the weight distribution is rectangular, and we have seen before that the SICNN with a rectangular weight distribution has good localisation of edges, but poor resolution, resulting in thick edges with few spurious edges. The NSR is 0.342.

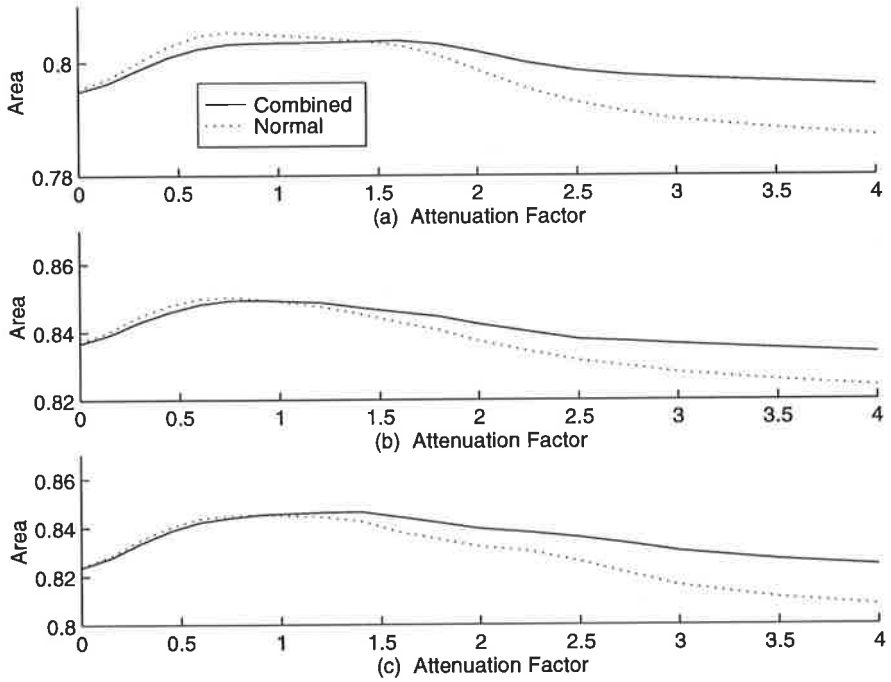


Figure 7.7 SICNN performance with and without combination of the outputs as a function of the attenuation factor. The SICNN has (a) $r = 5$, (b) $r = 10$, and (c) $r = 15$, the optimal decay factor is used and $ENR = 5$ dB with Gaussian noise.

Figure 7.8(b) shows the output with the optimal weight distribution for $p = 4$, and no postprocessing. Some of the true edges are detected, but the image is very noisy, even noisier than the output in (a). This is expected, because in this case the weight variation is very narrow - thus, the SICNN has good resolution but poor noise performance. This is reflected in the NSR which has increased to 0.401.

Figure 7.8(c) shows the SICNN output with the optimal weight distribution for $p = 4$, when the outputs with p increasing from 0 to 4 are combined using the above algorithm.

By beginning the combination process with the output for $p = 0$, which has some noise but thick edges, and tracking these edges as p increases to 4, where this output has thin edges with considerable more noise, an output can be obtained which is both relatively noise-free and has thin, well-localised edges. Although the output with postprocessing still has some edges missing, there is clearly an improvement over the outputs shown in (a) and (b), and this is reflected in the NSR of 0.178, which is smaller than the NSR of the other two outputs.

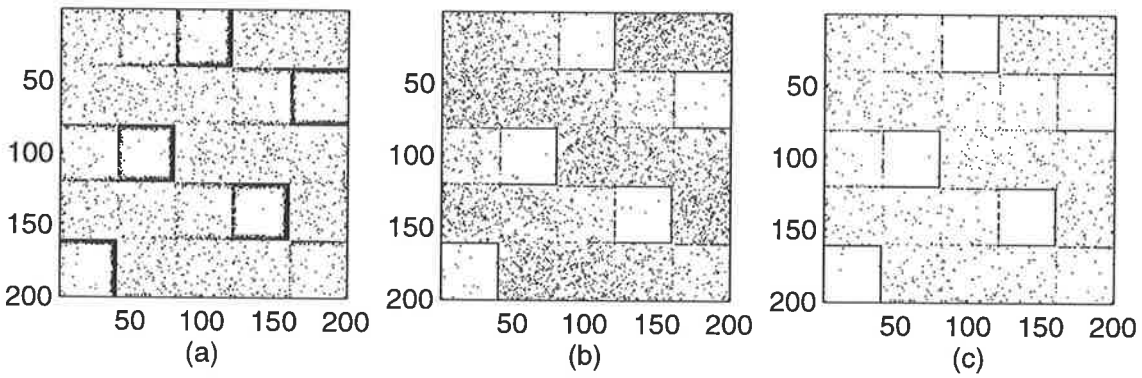


Figure 7.8 SICNN output an $r = 5$ SICNN for (a) $p = 0$ with no combination, (b) $p = 4$ with no combination, and (c) $p = 4$ with combination of the outputs. The optimal weight distribution and decay factor are used. The input has Gaussian noise with $ENR = 5$ dB. The NSR is (a) 0.342, (b) 0.401, and (c) 0.178.

The improvement is even more evident for the image with multiplicative noise shown in Figure 7.9. The output in (a) for $p = 0$ is very clean, but the detected edges are quite thick. The output in (b) for $p = 4$ is very noisy, however, most of the edges are thin. Finally, the output after postprocessing in (c) is both clean and the detected edges are thin and well-localised. It's NSR is 0.30, which is clearly superior to 1.63 and 1.40, the NSR of the outputs in (a) and (b), respectively.

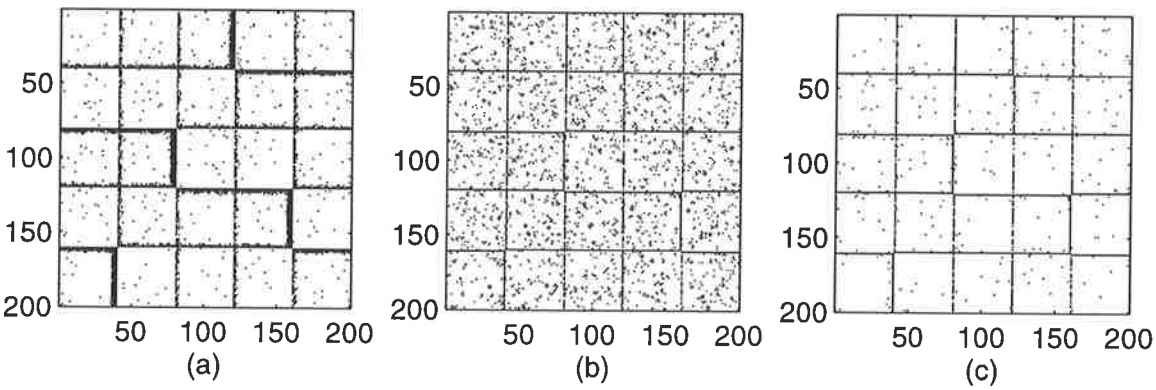


Figure 7.9 SICNN output an $r = 5$ SICNN for (a) $p = 0$ with no combination, (b) $p = 4$ with no combination, and (c) $p = 4$ with combination of the outputs. The optimal weight distribution and decay factor are used. The input has multiplicative noise with $ENR = 10$ dB. The NSR is (a) 1.63, (b) 1.40, and (c) 0.30.

7.4 Combination of SICNN Outputs

We have already mentioned in Section 5.4.1 that the SICNN can perform edge detection in a number of different ways, i.e., by detecting the maximum output with asymmetrical weights, detecting the maximum output with symmetrical weights, and the maximum zero-crossing in the output with symmetrical weights. If any two of these outputs contain information about the same source, then combining them may give better performance than using either output alone. This is the basic principle of data-fusion, where multiple observations are combined to obtain improved results.

7.4.1 One Dimension

We begin by looking at the combination of the SICNN outputs for 1-D step edge inputs. Consider any two outputs, i.e., the outputs from any of the two different SICNNs mentioned above. The basic procedure is to locate the n largest outputs in one of the output (where $n \geq 1$), and then define small local "valid regions" about each of these n edge pixels. The edge is then located in the valid region of the output of the 1st SICNN. By introducing valid regions about the n maxima of the output, we limit the search space where the edge is sought in the 2nd output, thereby increasing the performance.

Four different combination schemes of the SICNN outputs are investigated:

- combining the maximum output of the SICNN with asymmetrical weights, with the n largest zero-crossings of the SICNN with symmetrical weights (we call this "MAW - nZCSW", which denotes Maximum Asymmetrical Weights and n- Zero-Crossing Symmetrical Weights).

- combining the largest zero-crossing of the SICNN with symmetrical weights with the n largest maxima in the output of the SICNN with asymmetrical weights (we call this "ZCSW - nMAW"),

- combining the maximum output of the SICNN with asymmetrical weights, with the n largest maxima in the output of the SICNN with symmetrical weights (we call this "MAW - nMSW"),

- combining the maximum output of the SICNN with symmetrical weights with the n largest maxima in the output of the SICNN with asymmetrical weights (we call this "MSW - nMAW").

The algorithm for the **MAW-nZCSW** scheme, for example, is:

1. Apply the SICNN with symmetrical weights.
2. Find the n largest zero-crossings in this output, and then define valid regions about each one.
3. Find the maximum output with asymmetrical weights, which lies in the valid region defined in 2).

The algorithm for the nZCSW-MAW, MAW-nMSW and MSW-nMAW combinations are analogous to this one.

For each combination scheme, selecting approximately the 10 largest values in the output to define the valid region, gives the best results. The combination scheme's performance for different number of maximum outputs can be found in Appendix C.

Figure 7.10 shows a comparison of the edge detection performance for the four different combination schemes. The MAW-nZCSW combination of the outputs has the largest HR, and also the lowest ESD. The other combination schemes all have comparatively smaller HR, and equal or larger ESD. Further results of combinations for inputs with multiplicative and uniform noise are given in Appendix C.

Figure 7.11 shows the improvement in the performance using the combination technique. The performance with asymmetrical weights (maximum thresholding), and the performance with zero-crossing with symmetrical weights, are both clearly inferior to the performance of these two outputs combined (MSW-nZCSW), as given by the algorithm above. Appendix C shows the improvements of the other three combination schemes over the performance with no combination.

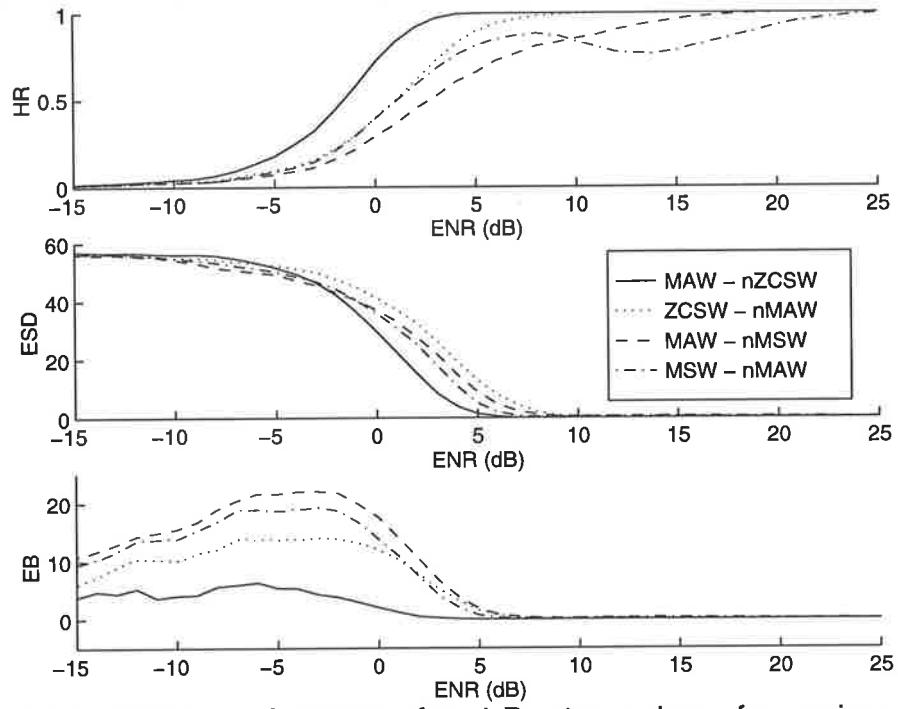


Figure 7.10 SICNN performance for 1-D step edges for various output combinations. The SICNN has either a symmetrical or asymmetrical rectangular weight distribution, with $r = 5$, $I_o = 10$, $c = 0.25$, and Gaussian noise.

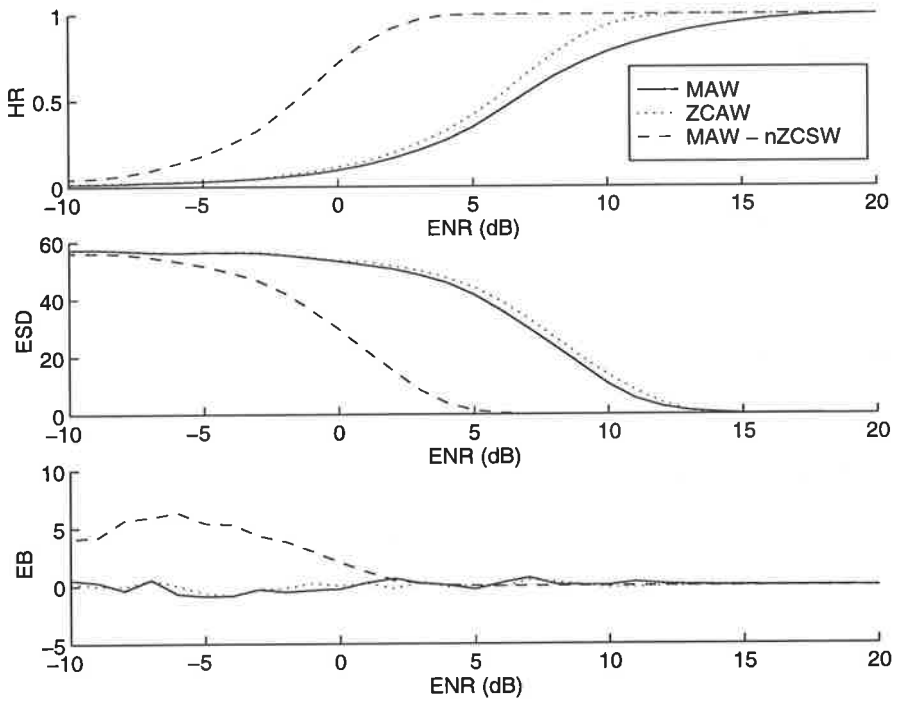


Figure 7.11 SICNN performance for MAW, ZCSW, and combination of MAW and ZCSW. The SICNN has rectangular weight distributions with $r = 5$, $I_o = 10$, $c = 0.25$, and Gaussian noise.

7.4.2 Two Dimensions

We have just seen how combining different SICNN outputs can greatly improve its 1-D edge detection performance. Now we investigate the same combination schemes, but for the 2-D synthetic image. Thus, for any two SICNNs one output is thresholded to detect the edges, and about each edge a local valid region is defined. We then apply a threshold to the second SICNN output, and detect any edges which lie in the valid region of the first SICNN output.

As with the 1-D case, there are four possible combination schemes: MAW-nZCSW, ZCSW-nMAW, MAW-nMSW, and MSW-nMAW. Thus, the edges in the 1st output are sought in the valid region of the 2nd output. In the 2-D implementation of each scheme, we apply a SICNN to each of the four orthogonal direction of the image. Thus, the algorithm for the SICNN applied to one of these directions, for the MAW-nZCSW, is given by

1. Apply a SICNN with symmetrical weights to the input.
2. Apply a threshold to the magnitude of the zero-crossings to detect edges.
3. Define valid regions about each edge output.
4. Apply a SICNN with asymmetrical weights and find the edges which lie in the valid region.

The edge detection performance for the four possible schemes was computed and by comparing all of them, the MAW-nZCSW combination scheme performed the best. The performance comparison between the four schemes can be found in Appendix C.

Figure 7.12 shows the performance comparison between the MAW-nZCSW scheme and the individual performance of the SICNN with MAW and the SICNN with ZCSW. We can clearly see that combining the MAW and ZCSW outputs produces gives the same performance as MAW. Thus, there appears to be no benefit in combining the output of the SICNN with MAW and ZCSW. (Further plots of the combination performance with different schemes and noise types can be found in Appendix C).

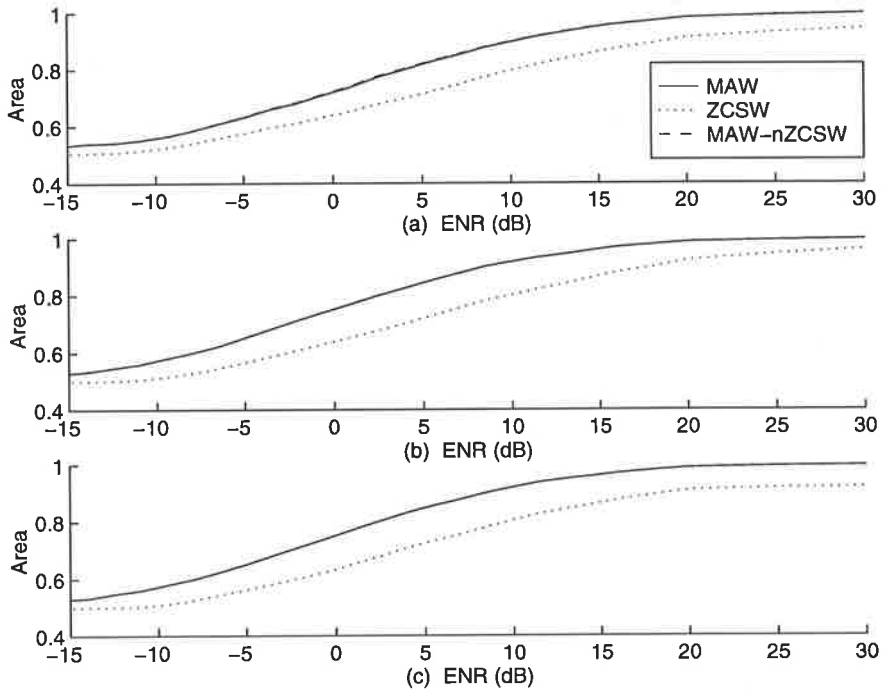


Figure 7.12 SICNN 2-D performance for MAW, ZCSW, and the combination of MAW & ZCSW with (a) $r = 5$, (b) $r = 10$, and (c) $r = 15$. The input has Gaussian noise.

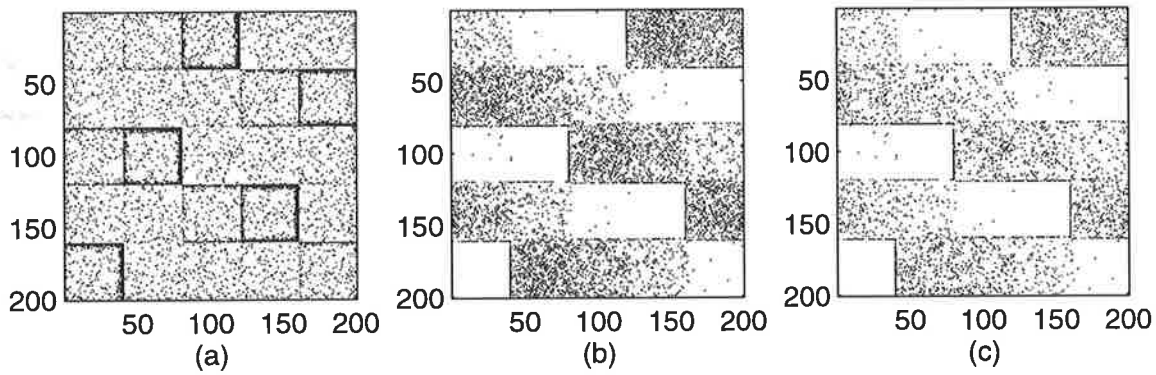


Figure 7.13 SICNN output for an $r = 5$ with (a) MAW, (b) ZCSW, and (c) combining the MAW and ZCSW outputs. The input image has Gaussian noise of $ENR = 5$ dB, and the NSR is (a) 4.34, (b) 8.59, and (c) 5.77.

Figure 7.13 shows the edge map outputs for the two SICNNs (MAW and ZCSW) as well as the combination scheme's output. Figure 7.13(a) shows the output with MAW, (b) shows the output with ZCSW, while (c) shows the output after combining the MAW and ZCSW outputs. All three outputs have detected some of the edges very well, but the amount of noise present in all outputs is very large. The image in (a) has noise uniformly spread within it, while both outputs in (b) and (c) have some regions which are very clean, but other regions which are very noisy. Unfortunately, by combining the SICNNs outputs

we cannot improve the quality of the detected edges at all. The NSR values for the three outputs are (a) 4.34, (b) 8.59, and (c) 5.77. These values again show the inability of the 2-D combination scheme to improve the edge detection performance. Similarly bad results are obtained for different combination schemes and noise types.

7.5 Complementary Output Processing

It is easy to show that the peak response to an edge relative to the background intensity of a SICNN with asymmetrical weights depends upon the direction of the weights. For example, the weight distribution $[0\ 0\ 1]$ is "reversed" in direction compared to $[1\ 0\ 0]$. Figure 7.14(a) and (b) show the SICNN responses to a step edge for two weight distributions, which are identical in shape, but reversed in direction. Note that the background intensities are identical in both cases, while the position of the edge response is displaced by only 1 pixel. We call these two outputs *complementary outputs*, and it is possible to combine these two to obtain an overall better output, i.e., an output which results in better edge detection performance.

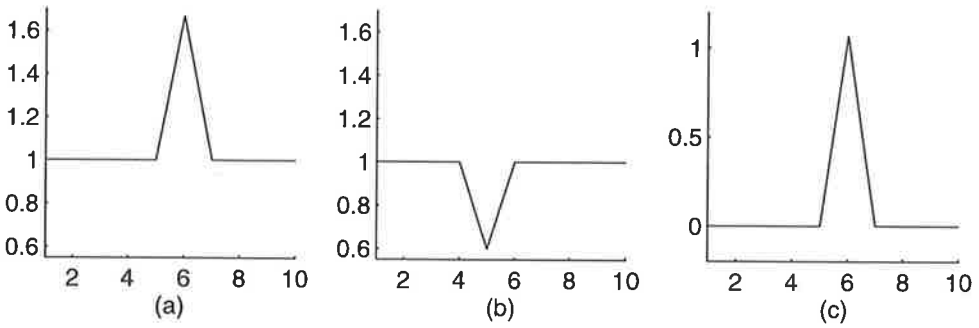


Figure 7.14 SICNN output to a step edge input for a weight distribution of (a) $w = [0\ 0\ 1]$, and (b) $[1\ 0\ 0]$. (c) shows the result of shifting the output shown in (b) by one pixel and then subtracted from the output shown in (a).

7.5.1 One Dimension

From Figure 7.14, the only difference in the SICNN output with reversed weight distribution to a step edge is the magnitude of the edge responses, and the 1-pixel separation between the responses. Thus, a simple approach to enhancing the output is to subtract from each pixel, such as the signal in (a), by the adjacent pixel intensity of the complementary output, as shown in (b). This subtraction process not only removes the

background intensity, but it increases the edge response compared to the background intensity. Unfortunately, if noise is also present, then the noise variance of the output after the subtraction process may also increase.

It can be shown that for a SICNN with asymmetrical weights summing to 1 and a step edge input of contrast c, subtracting the complementary outputs (after shifting) increases the peak edge response relative to the background noise intensity if the input edge's contrast satisfies

$$\frac{1}{8}(c-3)^2 \geq 1$$

$$\Rightarrow c \leq 3 - 2\sqrt{2} \approx 0.17 . \qquad \text{EQ (7.2)}$$

Figure 7.15 compares the edge detection performance of the normal SICNN with the performance after subtracting the complementary outputs. Subtracting the complementary outputs certainly improves the overall performance as the HR is larger and the ESD is smaller. The EB, however, is much larger for the output after postprocessing than the normal output, but this occurs only for very low ENR. Further results for different weight distributions and noise model types can be found in Appendix C.

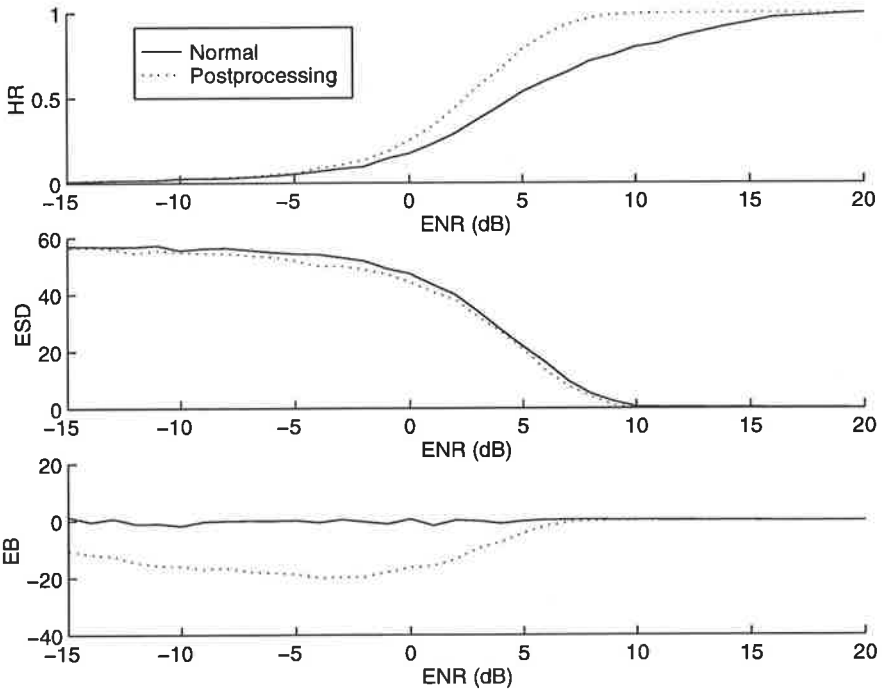


Figure 7.15 SICNN performance with and without complementary output postprocessing. The SICNN has an asymmetrical, rectangular weight distribution, with optimal decay factor, $r = 5$, $I_o = 10$, $c = 0.1$, and Gaussian noise.

7.5.2 Two Dimensions

We now extend the 1-D complementary output postprocessing technique to the 2-D case. In 2-D, the SICNN is applied to each of the two orthogonal directions of the image. The asymmetrical weight distribution is appropriately selected to perform edge detection in these two directions. The complementary output images are formed by applying the same SICNN in each direction, but with the weight distribution reversed. Each complementary output for each direction is then shifted and subtracted from the normal output for that particular direction.

Figure 7.16 shows the 2-D performance on the synthetic image both with and without complementary postprocessing. The performance of the output with postprocessing is always equal to or worse than the normal output's performance. The most likely reason for this was described in Section 7.5.1. We showed that, for step edges, the noise strength decreases for complementary postprocessing only when the contrast of the edge is less than about 0.17. However for the synthetic image, the average contrast is over 0.52, so it is clear that complementary postprocessing will not yield better performance than the normal SICNN output for this particular image. The comparison for different noise types is presented in Appendix C.

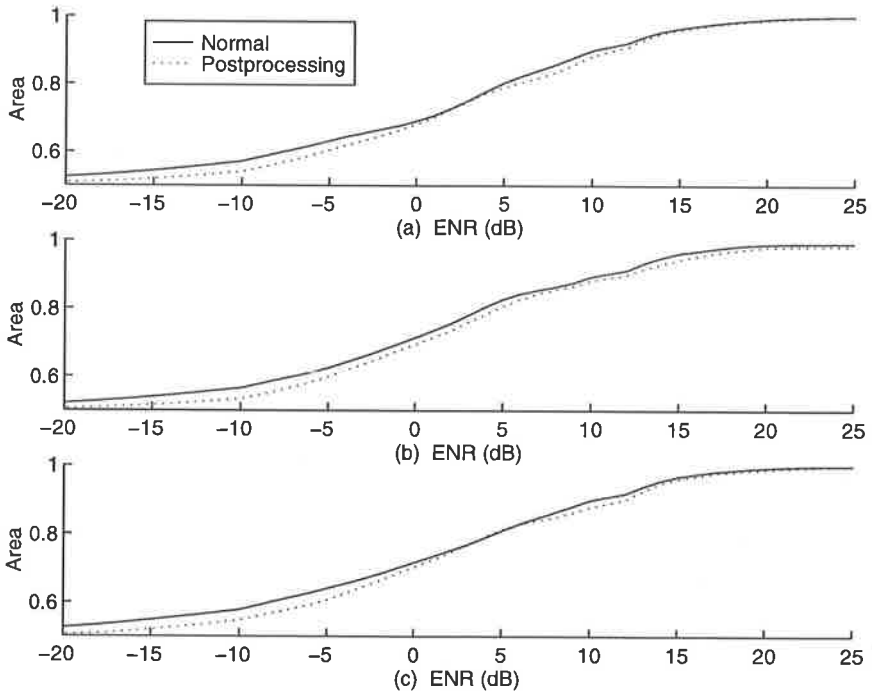


Figure 7.16 SICNN 2-D performance with and without complementary output postprocessing for (a) $r = 5$, (b) $r = 10$, and (c) $r = 15$. The SICNN has an asymmetrical, optimal weights and decay factor. The image has Gaussian noise.

Figure 7.17 shows the comparison of the edge maps for the normal output and also the output after complementary postprocessing. The normal output is very noisy, but some edges are still identifiable, however, they are somewhat broad. The output after postprocessing has also identified these edges, but they now much thinner and well localised. This output also has regions which are clean and other regions which are very noisy. The NSR values of 3.86 and 3.62 for (a) and (b) respectively, indicate that the output after postprocessing is slightly better than the normal output.

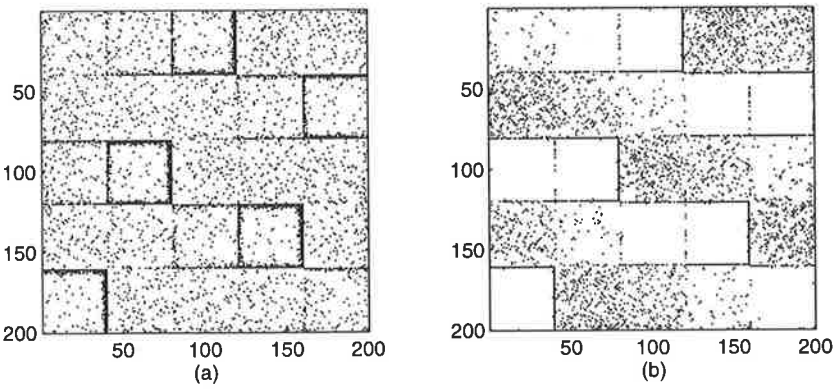


Figure 7.17 SICNN output with (a) no postprocessing, (b) complementary output postprocessing. The SICNN has an asymmetrical optimal weights and decay factor, $r = 5$, $ENR = 5$ dB and Gaussian noise. The NSR is (a) 3.86 and (b) 3.62.

7.6 Neighbourhood Processing

The algorithms described above to improve the SICNN 1-D and 2-D edge detection performance are both quite sophisticated and computationally expensive. There is, however, a simple postprocessing technique which can greatly improve the performance. In what we call "neighbourhood processing" we assume that we have the output edge map, and we then simply look at the number of neighbours in the vicinity of each edge pixel to determine if that edge pixel should be retained or discarded.

The rationale is that, if an edge pixel has very few neighbours or none at all in its vicinity, then it is likely that the edge pixel is noise and is not part of a valid edge segment in the input. Of course, such an assumption is highly dependent upon the input, but if the local window size is kept small and the number of edges required in each of these local windows is not large, then we can assume that the above rationale is reasonable. For

example, if there is an isolated edge pixel in a local 5×5 window, then it is likely to be a spurious edge and is not a valid edge pixel, thus it can be eliminated.

The algorithm for neighbourhood processing is the following.

1. For each edge pixel in the input image, find the number of edge pixels in its $N \times N$ neighbourhood.
2. If the number of neighbouring edges in the local window is greater than or equal to n, where $n \geq 0$, then it is deemed to be part of a valid edge structure and thus retained, otherwise, the edge pixel is declared not to be a valid edge pixel and is removed.

Figure 7.18 shows the performance using 2-D neighbourhood processing. The input image has Gaussian noise and a 3×3 local neighbourhood is used. The performance is measured for different number of local neighbours, and for various neighbourhood sizes.

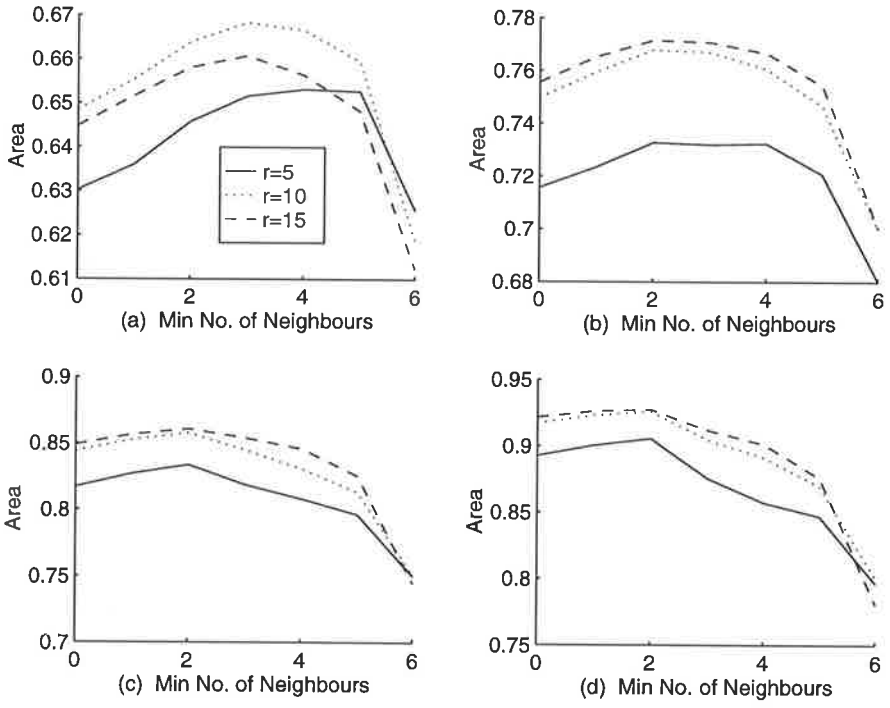


Figure 7.18 SICNN performance with 3×3 local neighbourhood postprocessing as the minimum number of neighbours is varied. The input has Gaussian noise with ENR of (a) -5 dB, (b) 0 dB, (c) 5 dB, and (d) 10 dB. The SICNN has an asymmetrical optimal weights and decay factor.

For large ENR, the performance peaks when thresholding for a minimum of two neighbouring edge pixels, while for low ENR the performance peaks for 2, 3 and 4 neighbouring pixels. Thresholding beyond these minimum number of edge pixels only

decreases the performance. Further plots for 5×5 local neighbourhood processing and different noise types are given in Appendix C.

The improvement in the performance by using neighbourhood processing is evident from the output edge maps in Figure 7.19 when the input has multiplicative noise of $ENR = 10$ dB. Figure 7.19(a) shows the normal output which is very noisy but has identified some of the edges. With 3×3 local neighbourhood processing with a minimum of 2 neighbour as shown in (b), most of the noise has been removed, whilst the edges have been retained. A similar output is obtained in (b) for 5×5 local neighbourhood processing and a minimum of 4 neighbouring edge pixels. The NSR values for (a), (b) and (c) are 3.42, 1.89, and 1.98, respectively, indicating significant performance improvement with this simple postprocessing technique.

It is clear that neighbourhood processing improves the output, as edges in the image are part of some line, i.e., there are no isolated true edges. All isolated edges are very likely to be from noise. For an arbitrary input image, the improvement would not be so dramatic, but there would still be an improvement, nonetheless.

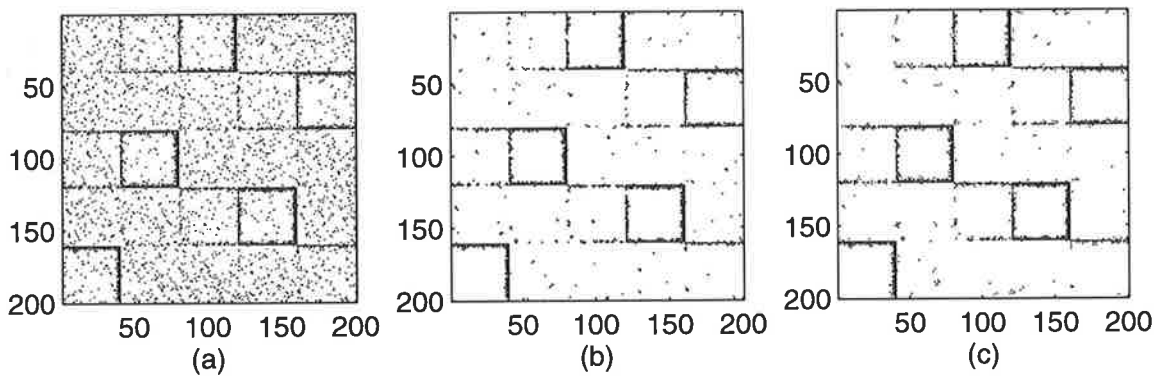


Figure 7.19 SICNN output for (a) no postprocessing, (b) 3×3 local neighbourhood processing with a minimum of 2 neighbouring pixels, and (c) 5×5 local neighbourhood postprocessing with a minimum of 4 neighbouring edge pixels. The SICNN has an asymmetrical optimal weight distribution, $r = 5$, and the input has Gaussian noise of $ENR = 5$ dB. The NSR is (a) 3.42, (b) 1.89, and (c) 1.98.

7.7 Conclusions

In this Chapter we investigated a number of postprocessing methods which can be used to improve the SICNN edge detection performance. Some of the methods are quite general and can be applied to a broad range of edge detectors, such as the Laplacian-of-Gaussian.

We began by reviewing the scale-space literature, and implemented a similar algorithm for the SICNN output. In the output, the edges were tracked as the scale was continually decreased. For 1-D step edges, the HR improved dramatically without increasing the ESD. Although the area under the PD vs. FA curve for the 2-D synthetic image did not increase as the scale was reduced, the area was always greater than the area for the normal output. The quality of the detected edges did improve with postprocessing as was evident from the edge maps. The output after scale-space processing was much cleaner with better quality edges than the output of the normal SICNN. For one particular ENR, the NSR of the postprocessed output was 2.11 compared to 3.15 for the normal output, indicating significant performance improvement.

Next we looked at tracking the edges in the output as the width of the optimal weight distribution was gradually decreased by decreasing the attenuation factor. By tracking the edges, the HR increased whilst the ESD was kept constant. For the 2-D synthetic image, the output after weight combination process gave, in general, greater area under the PD vs. FA curve than the normal output. Furthermore, the edge maps were much cleaner and the edges far better defined for the output after postprocessing than the normal output, particularly for multiplicative noise. For Gaussian noise, the NSR was 0.178 with postprocessing, and 0.342 without postprocessing, while for multiplicative noise, the NSR was 0.30 and 1.63 for the outputs with and without postprocessing, respectively.

We then looked at improving the SICNNs performance by combining the outputs of SICNNs with different weight distributions and different thresholding schemes. The best combination performance was achieved when the maximum of the output with asymmetrical weights was combined with the zero-crossings of the output with symmetrical weights. Also, combining these two outputs gave far better performance than either output alone. In 2-D, the best combination of outputs was again the output with asymmetrical weights and maximum thresholding and the output with asymmetrical weights and zero-crossing thresholding. The performance after combining the two outputs, however, was always equal to or worse than maximum thresholding the output

with asymmetrical weights, hence there is no benefit in combining these outputs. This was illustrated for an image with Gaussian noise, where the NSR values were 4.34, 8.59 and 5.77 for the output with maximum thresholding, zero-crossing thresholding and the combination of these two, respectively.

Next we investigated the combination of the outputs with the same weight distribution shape but reversed in direction. This scheme improved the performance, particularly for inputs of low contrast. For this reason, when this technique was applied to the synthetic image, which has an average contrast of about 0.52, the performance was always slightly smaller than the performance of the output with no postprocessing. A small improvement was observed, however, in the output edge maps for Gaussian noise with a particular ENR, where the NSR values were 3.62 and 3.86 for the output with and without postprocessing, respectively.

Finally, we investigated a simple technique to improve the appearance of the output. We counted the number of edge in a local neighbourhood, and then discarded all edges which had fewer than a certain number of neighbouring edge pixel. Although this method is very simple, it dramatically improved the appearance of the output, particularly when the output was very noisy. Using a 3×3 local neighbourhood postprocessing improved the NSR from 3.42 to 1.89, for example.

Chapter 8 Edge Detection Comparisons

8.1 Introduction

Throughout the previous Chapters we described how to apply the SICNN to edge detection and how to choose its parameters to maximise the edge detection performance. Now we combine that work and understanding in order to compare the edge detection performance of the SICNN with and without Gaussian pre-smoothing, with that of the discrete differentiator, and the Deriche operator. The differentiator is the simplest linear edge detector, while our experiments showed[1] that the Deriche operator outperforms even Canny's operator, which is widely regarded as being one of the best edge detector. See Section 2.5.4 for a review of the Deriche operator.

We begin by comparing the performance of the various edge detectors on 1-D step edges with multiplicative and additive noise. The performance is measured using the HR, ESD, EB, the PD vs. FA curve and the area under this curve, and also another measure of performance (MOP). Then the performance of these edge detectors on the 2-D synthetic image with multiplicative and additive noise is compared. Again, we measure the performance in terms of the PD vs. FA curve and the area enclosed by it, and also the MOP.

Finally in Section 8.4, we compare the output edge maps of the edge detectors on two real images: a SAR image, which has multiplicative noise, and the standard Lenna image, which has additive noise. Both of these images are fairly typical of what we might encounter in the real world.

1. The actual comparison between the Deriche and Canny operators is not presented in this thesis.

8.2 One Dimensional Comparison

We begin with a comparison of the SICNN with two standard edge detectors for 1-D step edges. The four edge detectors investigated are:

- The SICNN with the asymmetrical, optimal weight distribution with attenuation factor $p = 1$, an $r = 5$ neighbourhood size, and a linear activation function.
- A SICNN, as above, but with Gaussian filtering or smoothing of the input. The Gaussian's support size is $2r + 1$, where r is the neighbourhood size.
- A discrete approximation to the ideal differentiator. The length of the filter is $2r + 1$, and is defined by:

$$D(n) = \begin{cases} \left(\frac{1}{n}\right)(-1)^n, & \text{if } n = -r, \ldots, -1, 1, \ldots, r \\ 0, & \text{otherwise} \end{cases}$$

- the Deriche 2nd order 1st derivative of length $2r + 1$. This operator was discussed in Section 2.5.4, and is defined by EQ (2.10) as

$$De(n) = kn\exp(-\alpha|n|) \qquad n = -r, \ldots, -1, 0, 1, \ldots, r$$

where $k = \frac{-(1 - \exp(-\alpha))^2}{\exp(-\alpha)}$. The parameter α is given by EQ (2.11) and is related to the width of the Gaussian filter used for smoothing the input.

To obtain a fair numerical and empirical comparison between these different edge detection schemes, a number of different performance measures are used to compare the edge detection performances. These measures are computed from numerical simulations of the edge detection process. They are:

- the HR, ESD, and EB as a function of the ENR,
- the FA rate and the PD for a given ENR,
- the area under the PD vs. FA curve as a function of the ENR, and
- a ***Measure of Performance (MOP)***, which is related to the PD and FA, and defined as

$$MOP = \frac{(1 + PD - FA)}{2} \qquad \text{EQ (8.1)}$$

for a given ENR. This is a modified version of the performance measure presented by Azevedo & Longini (1980) and Efeachor & Jervis (1993, pp. 704). Note that the

maximum MOP is 1 when the PD is one and the FA is zero. This measure gives a larger value for a greater number of true edge pixels detected (PD) and for a fewer number of non-edge pixels detected (FA).

8.2.1 Multiplicative Noise.

Figure 8.1 shows the results for a step edge input[1] with multiplicative noise. Figure 8.1(a), (b) and (c) show the HR, ESD and EB, respectively. The EB of the linear differentiator and Deriche operator are both very large for low to medium ENR. This is due to the nonuniform intensity of the multiplicative noise, which is larger for larger input intensities, resulting in inherent biasing. Thus, their ESD is fixed at about 30 pixels for low ENR. The SICNNs, on the other hand, have nonlinear characteristics which enable them to cope well with the multiplicative nature of the noise. The SICNNs can be thought of as inverting the nonlinear noise, which results in a uniform noise intensity across their output, hence giving zero EB. For the HR, the SICNN with smoothing performs best, followed by the Deriche operator, the SICNN, and then the discrete differentiator. In Figure 8.1(d), the MOP is plotted as a function of the threshold multiplier, such that the overall threshold T is

$$T = \mu + (\text{threshold multiplier})\,\phi$$

where μ and ϕ are the mean and standard deviations of the edge detector output. The results in this Figure, for an ENR of -5 dB, show that both the SICNN with smoothing and the Deriche operator have similar peak values of the MOP.

Figure 8.1(e) shows the PD vs. FA curve for the corresponding step input with ENR of -5 dB. Again, the curve of the SICNN with smoothing is slightly above that of the Deriche operator, followed by the curves for the SICNN and differentiator. Figure 8.1(f) shows the area or integral under the PD vs. FA curves. For large ENR, the SICNN with smoothing performs slightly better than the Deriche operator, whereas the reverse is true for low ENR. The areas for the SICNN and discrete differentiator indicate that these two edge detectors have inferior performance compared to the other two.

The most obvious reason why the SICNN with smoothing and the Deriche operator perform better than the other two edge detectors, is that they incorporate some form of

1. The input has a total length of 200 pixels, with the edge located halfway along this.

smoothing, which clearly must improve their performance compared to the other two edge detectors which have no smoothing of the input at all.

8.2.2 Gaussian Noise

Figure 8.2 gives the performance comparison for the Gaussian noise case. Figure 8.2(a) shows that the HR of the Deriche operator is best for large ENR, closely followed by the SICNN with smoothing, and then the SICNN and differentiator. In (b) the ESD of the SICNN with smoothing is smallest, followed by that of the Deriche operator, then the SICNN and differentiator. In (c), the EB of all four operators is almost zero, except for the SICNN with smoothing. (For both SICNNs, the optimal decay factor is estimated in a local window of size $2r+1$, using the value of λ as given in Section 6.3). Although the EB is less than zero for the SICNN with smoothing, we showed in Section 6.3.2 that the SICNN HR and ESD are not adversely affected by this.

Figure 8.2(d) shows the MOP plots. The Deriche operator peak MOP is clearly the largest, followed by that of the SICNN with smoothing, the SICNN and then the discrete differentiator. Consequently the PD vs. FA curve in (e) for the Deriche operator has the largest area, followed by the curves for the SICNN with smoothing, the SICNN and then the discrete differentiator.

In (f), both the Deriche operator and the SICNN with smoothing have the largest area under their PD vs. FA curve, with both curves being very similar. For medium to high ENR, the Deriche operator area is slightly larger than that of the SICNN with smoothing, whereas for low ENR the SICNN with smoothing area is greater than that of the Deriche operator. The normal SICNN and the differentiator both have much smaller areas.

Thus, both the Deriche operator and the SICNN with smoothing appear to perform equally well. Some measures indicate that the Deriche operator is slightly better, while other measures indicate that the SICNN with smoothing is better. The performance of the SICNN and differentiator is inferior to that of the other two.

8.2.3 Uniform Noise

Figure 8.3 shows the performance results for uniform noise. The Deriche operator has slightly better HR than the SICNN with smoothing, whereas for the ESD measure, the

SICNN with smoothing always has smaller ESD than the Deriche operator. For both the HR and ESD, the SICNN and differentiator perform worse than the other two operators. The EB of the four edge detectors are all very close to zero.

Although for an input with ENR of -5 dB, the Deriche operator MOP has the largest peak value, though the MOP of the SICNN with smoothing is somewhat close. Also, the area under the Deriche operator PD vs. FA curve for the same input is greater than that of the others. Consequently, the area under the Deriche operator PD vs. FA curve in (f) is greater than the other edge detectors over the range of ENR. The SICNN with smoothing area is comparable for large ENR, but the remaining two edge detectors have significantly smaller areas under their PD vs. FA curves.

Thus, it appears that for uniform noise, the performance of the Deriche operator and SICNN with smoothing are very similar, though some performance measures indicate that the Deriche operator is slightly better. The performance of the SICNN and differentiator are clearly inferior to the other two operators.

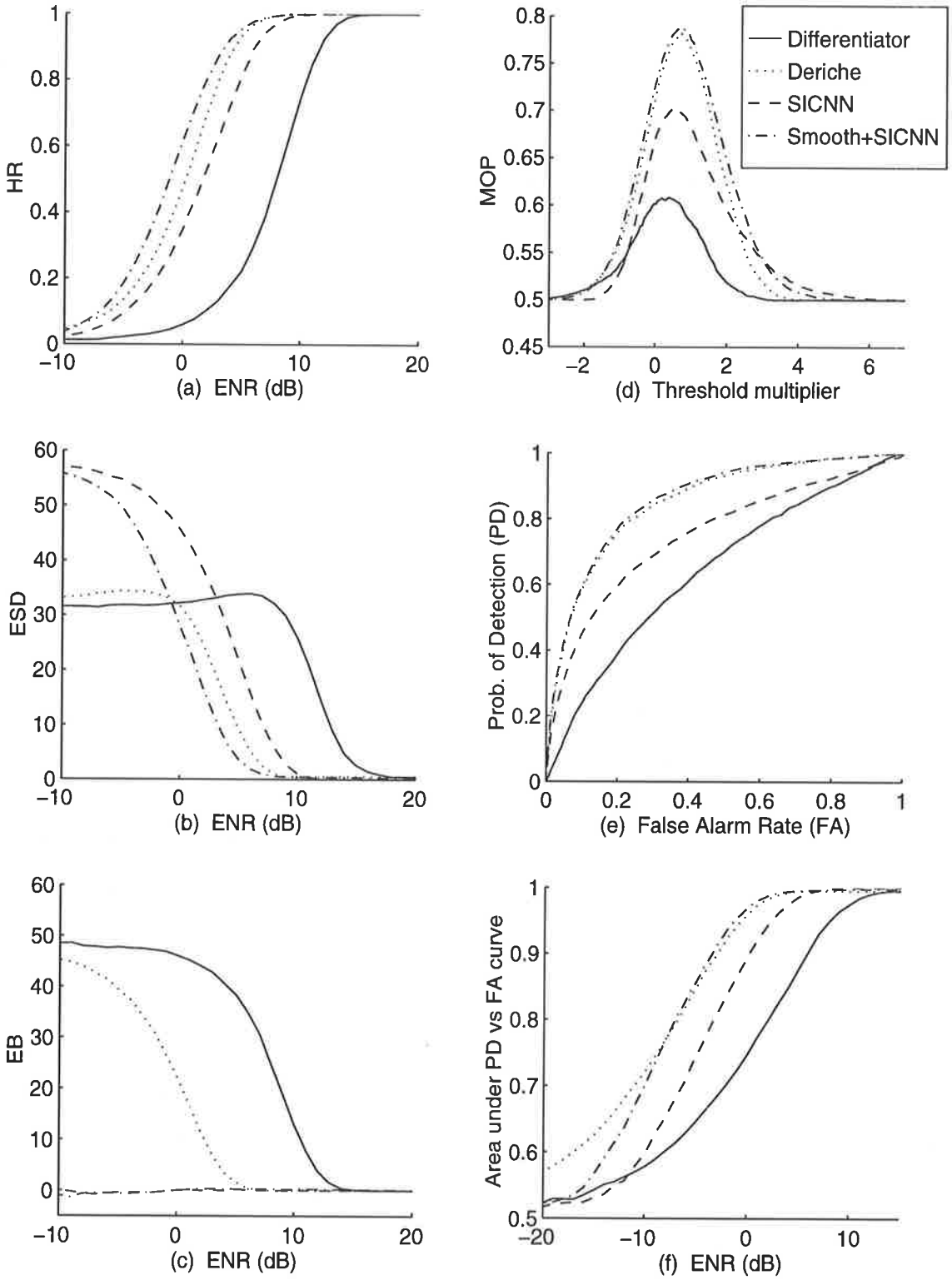


Figure 8.1 Performance comparison between the Deriche operator, discrete differentiator, SICNN, and SICNN with smoothing, for $I_o = 10$, $c = 0.25$ and **multiplicative** noise.

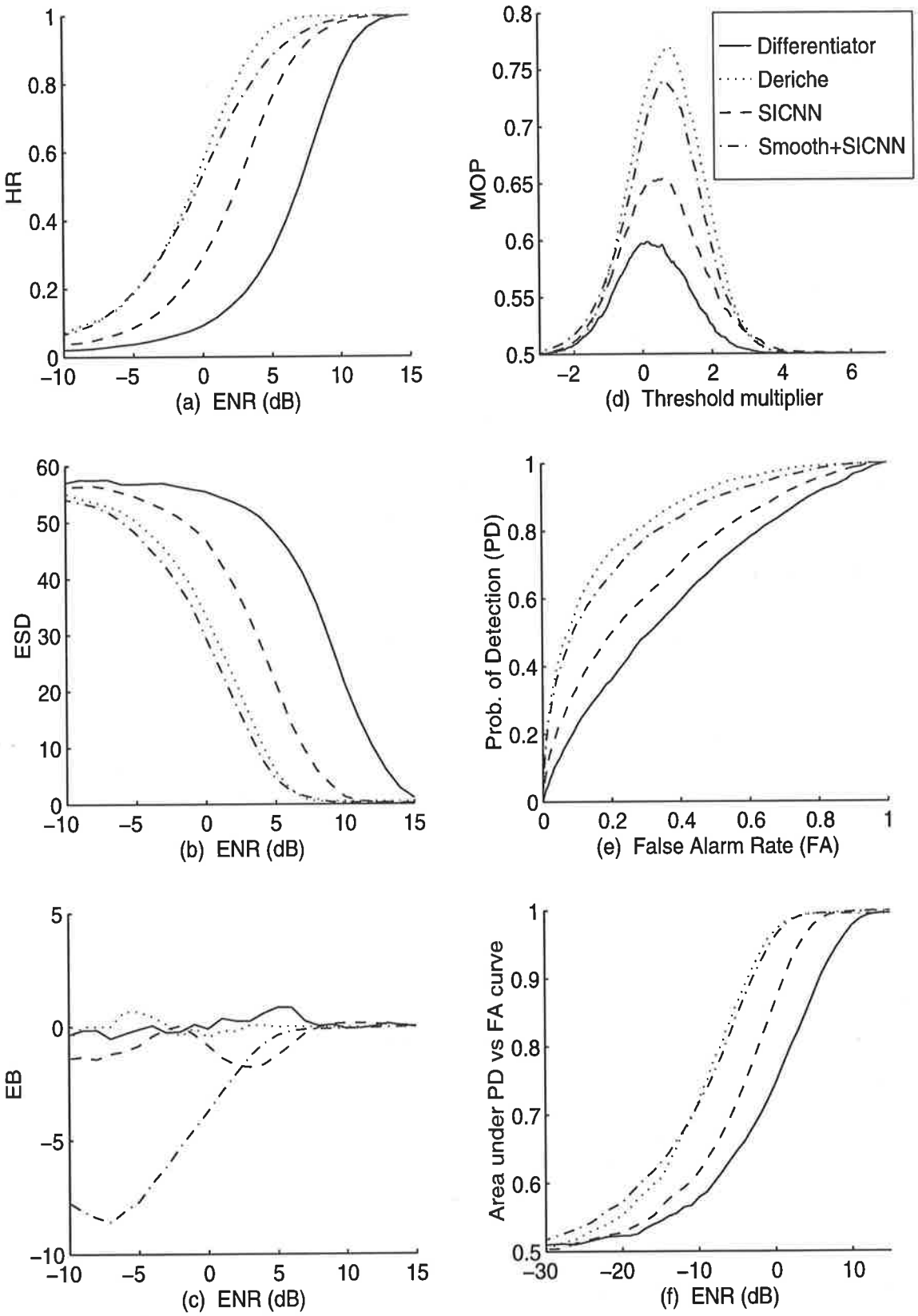


Figure 8.2 Performance comparison between the Deriche operator, the discrete differentiator, the SICNN, and the SICNN with smoothing for $I_o = 10$, $c = 0.25$ and **Gaussian** noise.

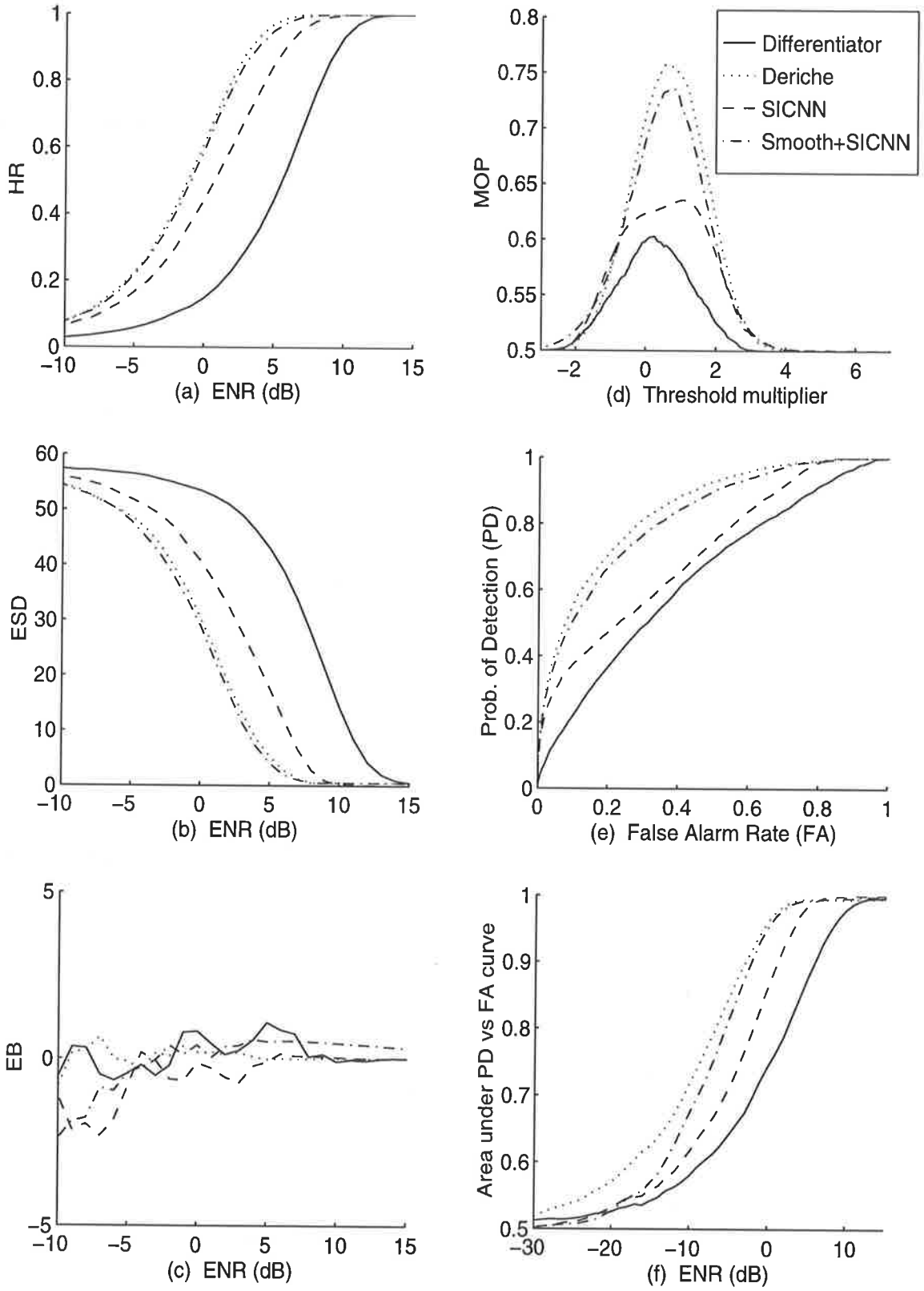


Figure 8.3 Performance comparison between the Deriche operator, the discrete differentiator, the SICNN, and the SICNN with smoothing for $I_0 = 10$, $c = 0.25$ and **uniform** noise.

8.3 Two-Dimensional Comparison on Synthetic Images

In this section we investigate the performance of the differentiator, the Deriche operator, the SICNN and the SICNN with Gaussian smoothing on the 2-D synthetic image shown in Figure 6.13. For fairness of comparison, the support size of each edge detector is $2r+1$, where r is the neighbourhood size of the SICNN. The support size of the Gaussian smoothing filter is also $2r+1$.

8.3.1 Two-Dimensional Algorithm

For 2-D edge detection the algorithm for each edge detector is:

1. Choose appropriately the edge detector so that it detects edges in each of the four orthogonal image directions, corresponding to $0°$, $90°$, $180°$ and $270°$. Thus, there are four output images for each edge detector.
2. Apply a global threshold to each of these output images, giving four binary edge maps or images for each edge detector. The threshold for each image is:

$$T = \mu + \eta\phi$$

   where μ and ϕ are the mean and standard deviations of the image, and η is the so-called threshold multiplier.
3. Logically OR the four binary edge maps to produce the overall edge map.

For an input image with a given ENR, the threshold is varied to produce the PD vs. FA curve, which is the primary performance measure that we use. The area under this curve can be computed, and the whole process repeated for a different ENR. This area, as the ENR varies, is another measure that we use to compare the edge detection performance of the four edge detectors. For a given ENR, the MOP, as is given by EQ (8.1), is also computed for different thresholds, while the NSR given by EQ (2.1) is used to compare the edge maps.

The discrete differentiator and the Deriche operator are calculated as in the 1-D case. The parameter α of the Deriche operator is computed according to EQ (2.11), assuming that $\sigma = r/3.5$, where σ is the width of the Gaussian filter used to smooth the input of the SICNN. Both SICNNs have linear activation functions, with optimal weight distribution and decay factor, as described in Section 6.3.

8.3.2 Multiplicative Noise

Figure 8.4 shows the performance results for the synthetic image with multiplicative noise. The neighbourhood size of the SICNN is $r = 5$, so the mask size of all four edge detectors is $2r + 1 = 11$ pixels. Figure 8.4(a) shows the MOP of each edge operator as the threshold multiplier is varied, where the curves for both SICNNs have larger peak values than those for the differentiator and Deriche operators. Interestingly, the MOP curve for the SICNN is better than that of the SICNN with smoothing. Figure 8.4(b) shows the FA vs. PD for an ENR of 5 dB. Again, the curves for both SICNNs are better than those of the other two linear edge detectors. For low FA rates, the SICNN with smoothing has a greater PD than the SICNN, but the reverse is true for large FA rates.

Figure 8.4(c) shows the area under the FA vs. PD curve as a function of ENR. Both SICNNs have the best performance for ENR greater than about -2 dB, followed by the Deriche operator and then the differentiator. For medium ENR, the SICNN with smoothing gives better performance than the normal SICNN, whereas the reverse is true for larger ENR. Thus, it appears that Gaussian smoothing only improves the SICNN performance when the input is very noisy.

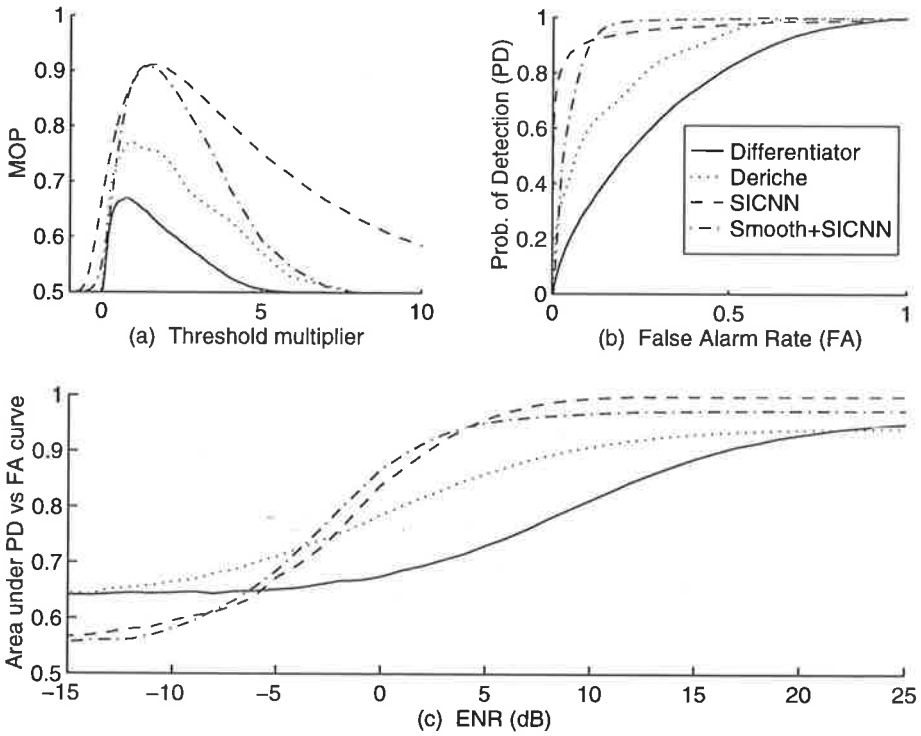


Figure 8.4 Performance comparison for differentiator, Deriche operator, SICNN and SICNN with smoothing using the (a) MOP, (b) PD vs. FA for ENR of 5 dB, and (c) the area under the PD vs. FA curve. The input image has multiplicative noise.

Before investigating the performance of the edge detectors for the input with additive noise, a number of important points need to be made for Figure 8.4(c). For large ENR, the area is less than 1, which is the expected maximum area. An area of 1 is obtained only when the PD is 1 for all FA.

The synthetic image consists of many edges of differing contrasts. Thus, when the threshold is set to detect the small responses from the low-contrast edges, it will not only detect the large response from large contrast edges, but also the neighbouring pixels on either side of these large responses (the total width of each edge response is equal to $2r + 1$, the size of the edge detector mask size). These false alarms (since the edge is only defined as being 1 pixel wide) cause the PD vs. FA to lose its sharp step-like shape, so the area is less than one. Furthermore, smoothing the input compounds the problem by making the edge responses even broader, as evident for the SICNN with smoothing. Note that, a neighbourhood size of $r = 1$ would avoid this problem.

Figure 8.5 shows the output edge maps for each edge detector with the threshold set to maximise the respective MOP, as given in Figure 8.4(a). Due to the multiplicative noise, both the differentiator in (a) and the Deriche operator in (b) detect large patches or regions of the input as edges. The corresponding input regions have large intensity, thus large noise variance too, so most of these output points will lie above the threshold, particularly if the threshold is set low to detect the input edge discontinuities of low contrast.

The SICNN in (c), however, detects most of the edges well, but the overall output is noisy, though not as noisy as in (a) or (b). The SICNN with smoothing in (d) is able to eliminate most of the spurious edges, but the detected edges are very thick, which is expected since the input is smoothed. This smoothing effect is also observable in the Deriche operator output.

The NSR values for these edge maps are (a) 12.30, (b) 8.63, (c) 2.24, and (d) 4.22, which clearly indicate the SICNN output is the best. From these NSR values, Figure 8.4, Figure 8.5, and from the results for different ENR as presented in Appendix D, both SICNNs perform best, followed by the Deriche operator and then the differentiator.

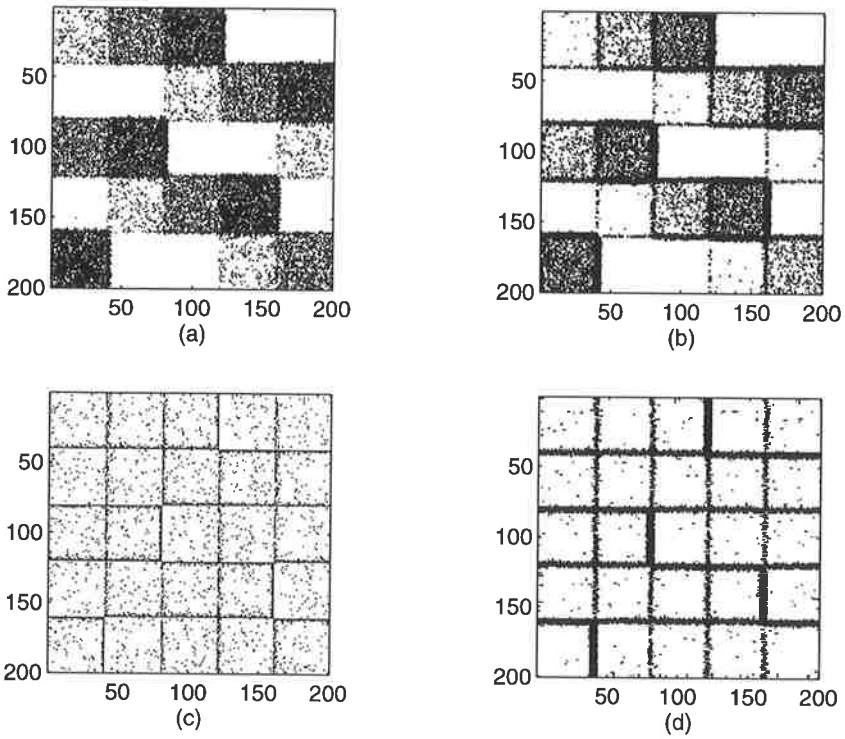


Figure 8.5 Edge map outputs of the (a) differentiator, (b) Deriche, (c) SICNN and (d) SICNN with smoothing for an input with multiplicative noise with *ENR = 5* dB. The outputs are chosen for thresholds according to Figure 8.4(a). The NSR values are (a) 12.30, (b) 8.63, (c) 2.24, and (d) 4.22.

Using these edge maps we can point out and explain a number of features of the curves in Figure 8.4. The area is not 0.5 for low ENR, primarily because the output noise is not uniform in intensity across the entire output. For a linear edge detector operating on an input with multiplicative noise, the output noise variance is much stronger in regions where the input intensity is large compared to where the input intensity is small. Thus, for a given threshold, the majority of edges are detected only in certain regions of the input, as evident in Figure 8.5 (a) and (b). The SICNNs area, however, would be 0.5 for low enough ENR. The differentiator and Deriche operator would not have such a problem for inputs with additive noise.

Another interesting point is that the MOP curves for both SICNNs have similar peak values, whereas this was not the case for the 1-D results observed before. It is difficult to state exactly why this is the case since the 2-D edge detection algorithm and outputs are not as easy to analyse as those for the 1-D case. Although the peak MOP values of both SICNNs are similar, their edge outputs are certainly different as evident from Figure 8.5.

Thus, the numerical results do not tell the complete story, so the output edge map should be observed.

8.3.3 Gaussian Noise

The comparison is now presented for the input synthetic image with Gaussian noise. The edge detection algorithm is identical to that used for the inputs with multiplicative noise, as are the edge detectors and the Gaussian smoothing filter.

Figure 8.6 shows the performance curves for the edge detectors for the synthetic image with Gaussian noise. The curves in (a) and (b) are computed for an ENR of 10 dB. Figure 8.6(a) shows the MOP curves, where the SICNN for this particular ENR is far better than the other edge detectors since it has a higher peak value.

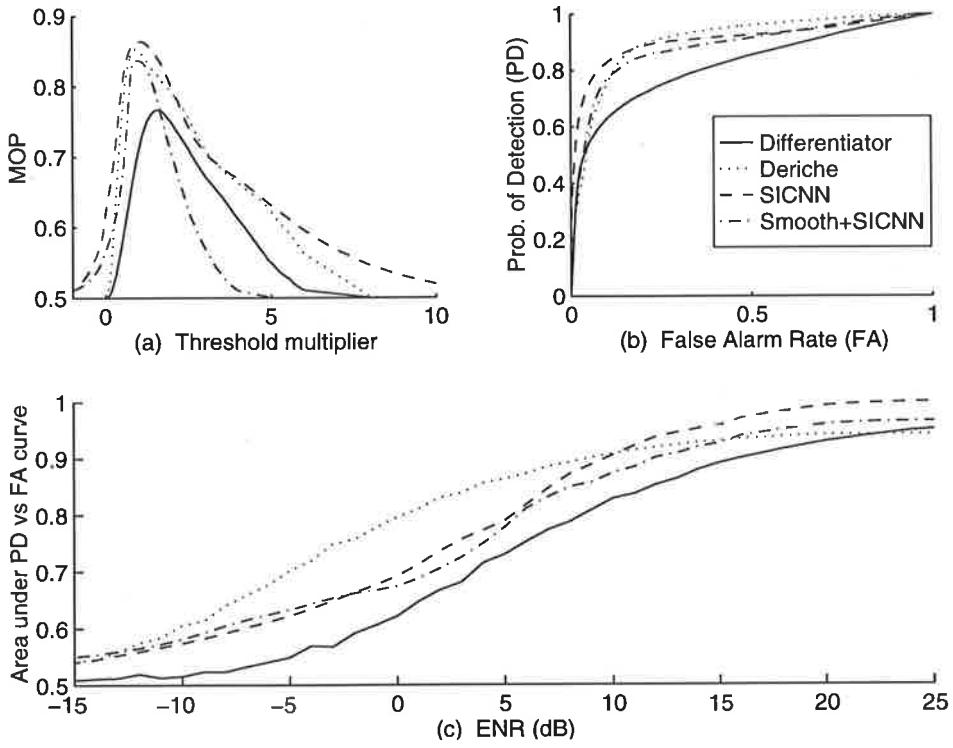


Figure 8.6 Comparison of the differentiator, Deriche operator, SICNN and SICNN with smoothing using the (a) MOP and (b) PD vs. FA for ENR of 10 dB, and (c) the area under the PD vs. FA curve. The input has additive Gaussian noise.

Figure 8.6(b) shows the PD vs. FA curves, and as expected from the MOP, the PD vs. FA curves for the Deriche operator and both SICNNs are somewhat similar, so it is difficult to rank their performance. The curve for the differentiator, however, is clearly inferior to the other edge detectors. Figure 8.6(c) shows that the Deriche operator has the largest area for

ENR less than about 10 dB. The performance of the two SICNNs are very close, but the SICNN with smoothing appears to be slightly better for low ENR, while the SICNN is better for large ENR. The curve for the differentiator is far below those of the other edge detectors.

Figure 8.7 shows the edge maps of all four edge detectors for a threshold selected such that the MOP is maximised as shown in Figure 8.6(a). The outputs of all four edge detectors are generally not very good. The output of the differentiator is noisy, with many edges missing, whereas the output of the Deriche operator is not as noisy but still has many edges missing and the detected edges are very thick. The output of the SICNN is very noisy in some regions, and many edges are also missing. The same is also true for the SICNN with smoothing, although the output is less noisy and the detected edges are quite thick and not well-defined. To enable us to objectively compare the outputs, we compute their NSR. The NSR values for these outputs are (a) 5.23, (b) 4.64, (c) 3.91, and (d) 4.47, which clearly indicates the superior quality of the SICNN output for the input with ENR of 10 dB over the outputs of the other edge detectors.

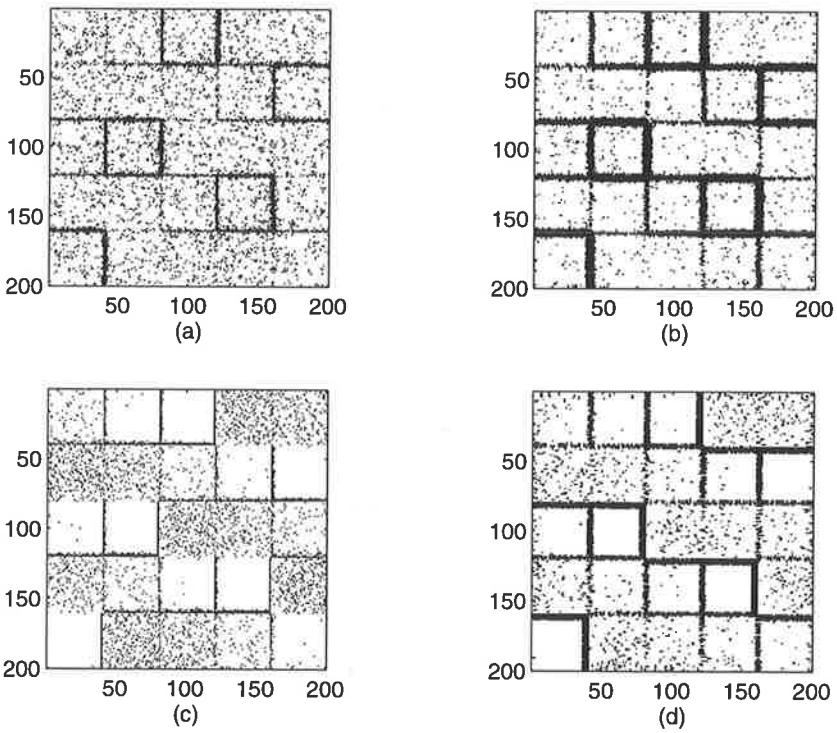


Figure 8.7 Edge map outputs of the (a) differentiator, (b) Deriche, (c) SICNN and (d) SICNN with smoothing. for an input with Gaussian noise with *ENR = 10* dB. The outputs are chosen for thresholds according to Figure 8.6(a). The NSR values are (a) 5.23, (b) 4.64, (c) 3.91, and (d) 4.47.

Some regions of both SICNNs outputs are noisy compared to other regions, since the output noise intensity is inversely related to the input intensity. Hence, input regions of low input intensity have very large output noise as compared to regions of the input with large intensity. Thus, when a global threshold is applied, in order to detect the responses from small contrast edges, the noise pixels in the regions of low input intensity are also unavoidably detected, hence we obtain regions in the output which are very noisy.

In any case, from the measure of the area under the PD vs. FA curve for the synthetic input image with Gaussian noise, the Deriche operator is best, followed by the SICNN and the SICNN with smoothing, and finally the discrete differentiator. Nevertheless, the edge map of the SICNN may be superior to those of the other edge detectors for a particular threshold, and ENR.

8.3.4 Uniform Noise

As for both the multiplicative and additive Gaussian noise results, we compare the edge detection performance of the discrete derivative, Deriche operator, the SICNN and the SICNN with Gaussian smoothing. Again, the support size of all edge detectors and the Gaussian filter is $2r+1$, where r is the neighbourhood size of both SICNNs.

Figure 8.8 shows the comparison using the usual performance measures. Figure 8.8(a) compares the MOP curves for an ENR of 10 dB. The peak MOP value for the Deriche, SICNN and SICNN with smoothing are all very similar, while the peak value for the differentiator is much smaller than the other three.

Figure 8.8(b) compares the corresponding PD vs. FA curves for an ENR equal to 10 dB. For large FA rate, the curves for the Deriche operator and the SICNN are very similar, but the SICNN has a much greater PD for small FA rates. The SICNN with smoothing PD is similar to that of the Deriche operator for small FA rates, but is slightly less for large FA rates. The curve for the differentiator is clearly inferior to those of the other three edge detectors.

Figure 8.8(c) compares the area under the PD vs. FA curve as a function of ENR. For inputs with low to medium ENR, the Deriche operator has the best performance, followed by the SICNN with smoothing, the SICNN, and then the differentiator. For large ENR, however, the SICNN has the best performance, followed by the SICNN with smoothing,

the Deriche operator and then the differentiator. Further results for different ENR are presented in Appendix D.

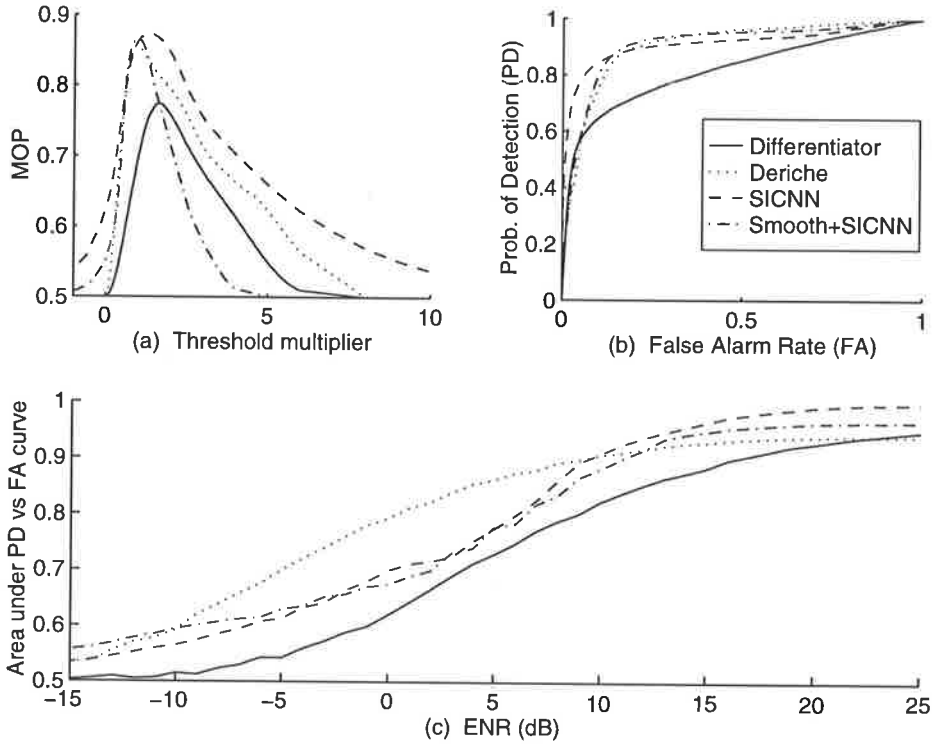


Figure 8.8 Performance comparison for differentiator, Deriche operator, SICNN and SICNN with smoothing, using the (a) MOP and (b) PD vs. FA for *ENR = 10* dB, and (c) the area under the PD vs. FA curve. The input image has additive uniform noise.

Figure 8.9 shows the output edge maps of all four edge detectors for inputs with ENR of 10 dB, and thresholds selected to maximise the MOP shown in Figure 8.8(a). The output of the differentiator in Figure 8.9(a) is very noisy, although some true edges are clearly visible, while the output of the Deriche operator is less noisy with more true edge detected, though these edges are very broad. The SICNN output in (c) has some of the true edges well localised, but the output is still quite noisy in some regions. The output of the SICNN with smoothing, as shown in (d), is not as noisy as the normal SICNN, but the detected edges are very thick and not localised well (the reason why the output of the SICNNs is noisy in certain regions only, was explained above for the 2-D results for Gaussian noise). To quantitatively compare the edge maps, the NSR values are (a) 4.31, (b) 4.38, (c) 2.81, and (d) 4.93. These values indicate that the SICNN output is much better than that of the other outputs.

Thus, it appears that, using the measure of the area under the PD vs. FA curve, both the SICNN and Deriche operator have similar performance, followed closely by the SICNN with smoothing, and then the discrete differentiator. Nevertheless, for a given ENR and threshold that maximises each edge detector's MOP, the SICNN output is clearly the best.

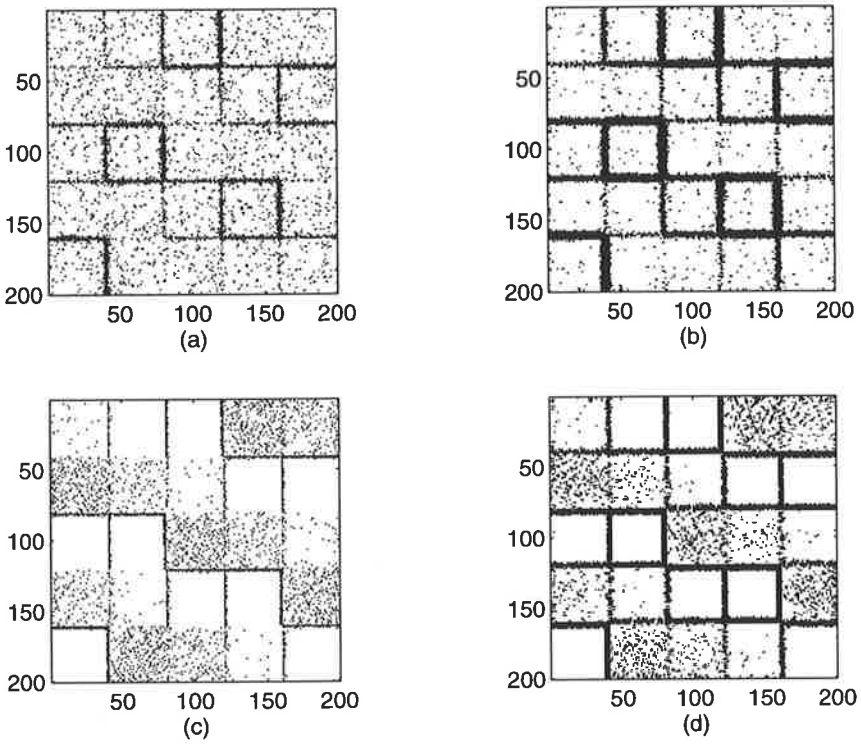


Figure 8.9 The edge map outputs of the (a) differentiator, (b) Deriche, (c) SICNN and (d) SICNN with smoothing for the input with uniform noise with *ENR = 10* dB. The outputs are chosen for thresholds according to Figure 8.8(a). The NSR values are (a) 4.31, (b) 4.38, (c) 2.81, and (d) 4.93.

8.4 Two-Dimensional Comparison of Real Images

All of the investigations and experiments to this point have been performed on either 1-D or 2-D synthetic edges, which allowed us to quantitatively analyse the SICNN performance and compare it to other edge detectors. We now compare the four edge detectors with 2-D real images. Figure 8.10 shows two real images used for this purpose. Figure 8.10(a) is a SAR image of a road junction located somewhere in South Australia. Due to the coherent imaging nature of SAR, it is well known that the dominant noise is multiplicative in nature. Figure 8.10(b) shows the Lenna image which is popularly used to compare the performance of edge detectors. Its noise is mostly assumed to be additive.

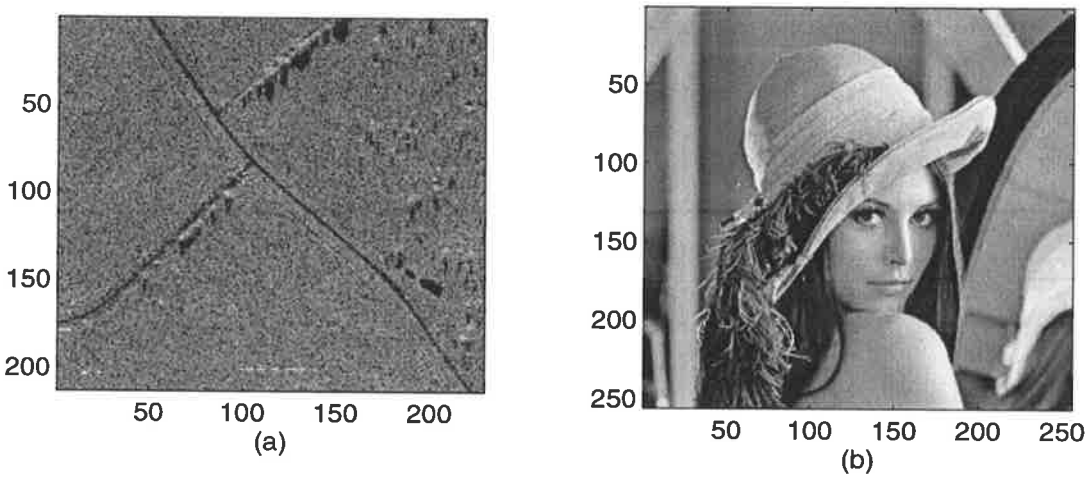


Figure 8.10 (a) shows a SAR image of a road junction, and (b) shows the Lenna image.

8.4.1 Results for SAR Image

It is well known that synthetic aperture radar (SAR) images have multiplicative noise, i.e., the noise intensity is proportional to the intensity of the pixel. Consequently, linear edge detectors that work well with images with additive noise, are not particularly suited to radar images. The advantages that the nonlinear properties of the SICNN give it over linear filters for synthetic 1-D and 2-D edges has already been observed, especially for inputs with multiplicative noise.

Figure 8.11 shows the comparison between the different edge detectors for the road junction SAR image as given in Figure 8.10(a). For both SICNNs the neighbourhood size is $r = 5$, thus the support size for each edge detector is $2r+1 = 11$ pixels. Since the image has predominately multiplicative noise, a decay factor of zero was used for both SICNNs. The optimal weight distribution is also used, with attenuation factor set to 1. The threshold for each edge detector's output is selected such that approximately equal number of edges are detected in each output.

Figure 8.11(a) shows the output of the differentiator, which is very noisy because we are differentiating an image which is very noisy, hence the noise is amplified, resulting in a poor probability of detecting a true edge. Figure 8.11(b) is the Deriche operator output. The road junction is clearly visible, although the road edges are quite noisy and broken. There is also some noise throughout the image, and there are many edges which appear to identify peripheral objects, but the image is too noisy to assert this with any confidence.

Figure 8.11(c) shows the output of the SICNN. The road is clearly identified and, like the output of the Deriche operator, the quality of detected edges is not good with the overall image being noisy. Interestingly, the main cross-diagonal road is a single pixel wide, whereas for the Deriche operator it appears as two sets of parallel lines very close together. The output of the SICNN with smoothing is shown in (d). This output is the least noisiest of the four edge detector outputs. Furthermore, the edges corresponding to the road are clearly visible, and it is likely here that the peripheral objects would not be confused with noise. Like the Deriche operator output, the road is detected as two parallel lines of edge pixels. The only problem with this SICNN output is that the edges appear quite thick, which is due to the fact that the input is smoothed with a Gaussian filter.

Thus, for both the synthetic 1-D and 2-D edges, the SICNN with smoothing has the best performance as compared to the SICNN with no smoothing and the linear edge detectors. Again, this is due to the ability of the SICNN to nonlinearly adapt to the varying intensity in its input.

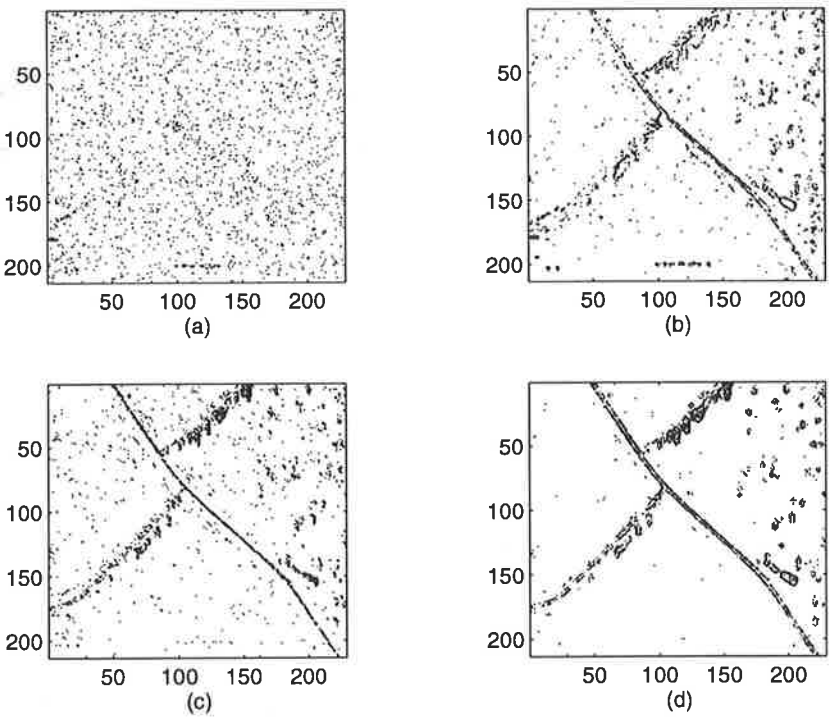


Figure 8.11 Comparison of the four edge detectors on the road junction SAR image. The outputs are for (a) the differentiator, (b) the Deriche operator, (c) the SICNN, and (d) the SICNN with Gaussian smoothing.

8.4.2 Results for Lenna Image

We now compare the performance of the edge detectors on the Lenna image shown in Figure 8.10(b). The differentiator and Deriche operator are identical to the ones used for the SAR image. We assume that the image has additive noise, so the optimal decay factor for both SICNNs is estimated in small, local regions. Once again, for a fair comparison the threshold for the outputs of each edge detector are selected so that the number of detected edges is approximately equal for each output.

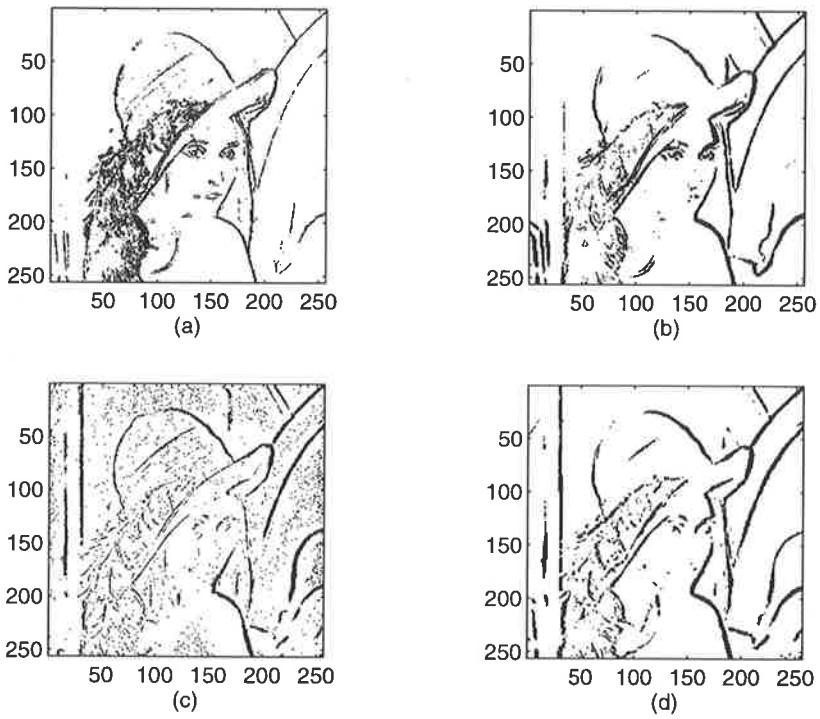


Figure 8.12 Comparison of the edge detector outputs on the Lenna image. The outputs are for the (a) differentiator, (b) Deriche operator, (c) SICNN, and (d) SICNN with Gaussian smoothing.

Figure 8.12(a) shows the output of the discrete differentiator. The edges in both the face and hat region are identified well, but many of the background features are missing. Similarly for the output of the Deriche operator shown in (b), the face and hat are detected, but not as well as the differentiator. The Deriche operator, however, is able to detect more background features. Figure 8.12(c) shows the output of the SICNN. In this case, the decay factor with a value of $\lambda = 1$ was found to give a good output. Although most of the edges in the face, hat and background regions are detected, the image is quite noisy. Figure 8.12(c) shows the output of the SICNN with Gaussian smoothing. Its best output is achieved with $\lambda = 0.5$. Clearly, because of the smoothing, this output is much less noisier

than that of the other SICNN, but some of the fine details of the face and hat have been lost. Nevertheless, most of the background features are still detected.

For the Lenna image, it is difficult to say which edge detector is best. The main problem is that the ground truth is not known, i.e., the position of the true edges in the image, thus it is difficult to accurately compare the different detectors. Nevertheless, it appears that the SICNN is able to detect most of the edges, even though there are many spurious edges compared to the other outputs. The SICNN with smoothing output is much cleaner and has also detected most of the edges, although these edges are now thicker due to the Gaussian smoothing. The outputs of the Differentiator and Deriche operator are very clean, but they do not detect some edges, particularly in the background regions.

8.5 Conclusions

In this Chapter we compared the performance of various edge detectors on both 1-D and 2-D synthetic edges, and two real images. For the 1-D step edges with multiplicative noise, the SICNN with smoothing performed the best followed by the SICNN with no smoothing, and then the Deriche operator and the differentiator. The SICNNs have an advantage over the linear edge detectors in that they can nonlinearly adapt to their input. For inputs with additive noise, both the Deriche operator and the SICNN with smoothing gave the best performance followed by the SICNN, and then the differentiator. Some performance measures indicated that the Deriche operator was best, while other measures indicated the SICNN with smoothing was best.

For the 2-D synthetic image with multiplicative noise, the SICNN with smoothing performed the best for low ENR, but for large ENR the SICNN performed best. Following these two were ranked the Deriche operator and the differentiator. The edge maps of both SICNNs were clearly better than those of the Deriche operator and the differentiator. The SICNN with smoothing output has less noise and thicker edges, while the outputs of the Deriche and differentiator were quite bad, due to the multiplicative nature of the input noise. The NSR values were 12.30, 8.63, 2.24, and 4.42 for the differentiator, Deriche operator, SICNN, and SICNN with smoothing, respectively. These values clearly indicate the superiority of the SICNN outputs over the outputs of the linear edge detectors.

For the input image with additive noise, the Deriche operator performed best in terms of the area under the PD vs. FA curve, followed by the SICNN with smoothing for low ENR, and then the SICNN and the differentiator. For ENR greater than about 10 dB, the normal SICNN performed the best. Looking at the output edge maps for an ENR of 10 dB, all outputs had many spurious edges, though the SICNN output had many clean regions, and localised the edges well. The NSR values for Gaussian noise were 5.23, 4.64, 3.91, and 4.47 for the outputs of the differentiator, Deriche operator, SICNN, and SICNN with smoothing, respectively. The corresponding values for uniform were 4.31, 4.38, 2.81, and 4.93. These values objectively indicate that the SICNN output was clearly the best.

These results also indicate that for the SICNN, smoothing the input with a Gaussian improved the performance, particularly for low ENR by removing much of the noise. For large ENR, the blurring effect of the filters appeared to be more significant and it caused the performance to deteriorate, so in this case it is better to use the SICNN without smoothing.

Finally we compared the outputs of the edge detectors on a number of real images. For a SAR with multiplicative noise, the SICNN with smoothing performed the best, while the SICNN and Deriche operator gave outputs of similar quality. The differentiator was not able to detect any significant edges at all. For the Lenna image, which has additive noise, the SICNN was able to detect most of the edges in the face and background region, though there were many spurious edges. The SICNN was smoothing output was very clean with most of the edges detected too, though they were thick due to smoothing. The outputs of the linear edge detectors were very clean, but missed many of the edges in the background regions.

Chapter 9 Edge Enhancement using the SICNN

9.1 Introduction

When a real scene is acquired by a camera of some sort, there is invariably a degradation in the image quality. This may be due to bad lighting, poor camera calibration, or low dynamic range in the viewing device. Enhancement is used to generate a more visually pleasing and informative image, though not necessarily a more accurate image of the original scene. Of all the image enhancement techniques, we are only interested in those that perform edge enhancement.

We begin by reviewing a number of image enhancement techniques, such as histogram processing, point processing, highpass filtering and homomorphic filtering. In particular we look at contrast transformation methods and nonlinear unsharp masking. Different image enhancement measures are reviewed, and we shall show that these are not entirely useful to us. Thus, a new measure is defined, the *edge enhancement product* (EEP), which measure both the enhancement of the edge pixel's intensity and the difference between the background intensity levels.

The edge-enhancing SICNN is then presented in Section 9.4. For the recurrent SICNN, we use an iterative approach to obtain its steady-state response, and it is the output of this iterative process which we use for edge enhancement. For the SICNN output after 1 iteration we show how the decay factor can be chosen to maximise the output EEP. The SICNN enhancement for different iterations is also investigated, and finally the performance of the SICNN (with and without smoothing) is compared to two other edge

enhancement schemes: the *gradient-like enhancement* and *hyperbolic tangent contrast enhancement* schemes.

9.2 Review of Image Enhancement

When capturing an image of any real-world scene, we can expect many degradations in the resulting image. These degradations may be due to the environment, such as poor lighting, or they may arise from inadequacies and limitations of the actual imaging device, such as poor camera calibration and insufficient dynamic range. These degradations result in a direct reduction of the image quality.

To generate a more perceptually pleasing and informative image, enhancement is used. Enhancement improves the quality of the original image - making it more suitable for a particular application or purpose. Note that enhancement does not restore the degraded image to its original state - this is image restoration. We are only interested in enhancement techniques that improve the edge contrast of the original image.

9.2.1 Spatial Domain Methods

Histogram Processing

We now review the techniques that attempt to modify the image so that its histogram has a desired shape. The histogram of an image represents the relative frequency of occurrence of the gray levels in the image. Mathematically, the histogram of an image is given by

$$h_j = n_j / N$$

where N is the total number of pixels, and n_j is the number of pixels in the image with gray-level j.

A popular and simple technique is *histogram equalization*, which transforms the input image in such a way that the output image's histogram is roughly uniform. Histogram equalization tends to increase small contrasts, and decrease large ones. If we let the output after histogram processing be I', and the output histogram to be uniform over the range of desired intensities $[I'_{min}, I'_{max}]$, then the desired transformation for an input pixel of intensity I is then (Sonka et al., 1993):

$$I' = \frac{(I'_{max} - I'_{min})}{N} \sum_{i=I_{min}}^{I} h_i + I_{min}$$

where N is the total number of pixels, and I_{min} is the minimum input gray level.

A simple adaptive version of this technique would apply equalisation over small local regions rather than over the entire image. More complex adaptive schemes can be found in Paranjape et al. (1992), and Pizer et al. (1987), while an evaluation of histogram equalisation for medical purposes was described by Zimmerman et al. (1988).

A slightly different approach is *linear histogram stretching*, where the smallest input intensity in the original image is stretched down to a desired minimum I'_{min}, and the maximum input intensity is linearly scaled up to a desired maximum I'_{max}. Thus, the required transformation for an input pixel of intensity I and for a desired output gray scale range $[I_{min},I_{max}]$ is

$$I' = I'_{min} + \frac{(I - I_{min})}{(I_{max} - I_{min})}(I'_{max} - I'_{min}) .$$

Once again an adaptive approach can be implemented by processing the image over a small, local neighbourhood.

Point Processing

There exist many spatial filtering enhancement schemes such as mean and median filtering, but we shall not consider those here as they primarily remove noise and do not enhance edges. A simple, but effective, algorithm to enhance edges is known as *high-frequency boosting* (Gonzalez & Woods, 1992) or *unsharp masking (UM)*, where the high-frequency contents of the input are partially added to the input. The enhanced output is given by

$$\begin{aligned} I_{Boost} &= \alpha \cdot I - I_{LP} \\ &= (\alpha - 1) \cdot I + I - I_{LP} \\ &= (\alpha - 1) \cdot I + I_{HP} \end{aligned}$$

where I, I_{Boost}, I_{LP} and I_{HP} are the original input, the boosted output, and the low-pass and high-pass versions of the input image, respectively. The parameter α controls the amount of the original image added to its high-passed version. A typical mask used to obtained the high passed version of the original input image is

$$\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & w & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

where $w = 9\alpha - 1$, $\alpha \geq 1$. This technique relies on the fact that edges are usually of high frequency in nature, hence this approach increases their strength compared to the background; of course noise is also boosted since it is of a high-frequency nature.

9.2.2 Frequency Domain Methods

Enhancement in the frequency domain is, in principle, straightforward. The Fourier transform of the image to be enhanced is computed, then it is multiplied by the filter's Fourier transform, and finally the inverse Fourier transform of this product is taken to obtain the enhanced image in the spatial domain. The filter is a high pass one, which implies that it only enhances the high frequency components of the input signal, such as edges and noise.

Highpass Filtering

To sharpen an image, the image's Fourier transform can be multiplied by the transform of a high-pass frequency filter such as the Butterworth highpass filter.

$$H(\omega_x, \omega_y) = \frac{1}{1 + \left[\frac{\sqrt{\omega_x^2 + \omega_y^2}}{\omega_o}\right]^{2n}},$$

where ω_x, ω_y are the frequency variables, ω_o is the cutoff frequency, and n is the filter order. Again, this approach emphasises the high frequency components (edges) of the image, which invariably includes noise too.

Homomorphic Filtering

A typical *homomorphic* approach to image enhancement is illustrated in Figure 9.1 (see Oppenheim et al., 1968; Jernigan & McLean, 1992); it is a special case of a class of systems known as homomorphic systems. The main motivation for this field of work is to apply concepts and structures of abstract linear algebra to image processing. It has found applications in image restoration, speech processing, and seismic signal processing, among others, and has also been psychologically connected to Fechner's law (see Pinoli, 1997).

Here it is assumed that the input image $f(x, y)$ can be written in terms of its illumination, $i(x, y)$, and reflectance, $r(x, y)$, components

$$f(x, y) = i(x, y)\, r(x, y).$$

It is also assumed that the reflectance component changes abruptly at discontinuities, while the illumination component does not. Thus, the reflectance component is associated with high frequency edges and the illumination component is associated with the slow-changing background. Homomorphic filtering allows us to separate both of these components of the input. The filter $H(u, v)$ is designed to affect the low and high frequency components differently, and is typically a highpass filter in order to accentuate the edges. The natural logarithm allows us to apply classical linear image processing techniques to each component of the input.

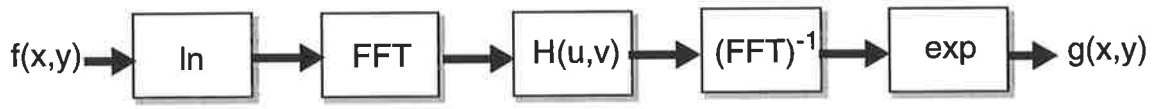


Figure 9.1 Homomorphic filtering for image enhancement (see Gonzalez & Woods, 1992).

9.2.3 Contrast Transformation Methods

A number of schemes (Gordon & Rangayyan, 1984; Dhawan et al., 1986; Beghdadi & Le Negrate, 1989; Dash & Chatterji, 1991) aim to explicitly vary the local contrast of an edge in an image. All of the proposed methods begin by computing the local contrast in a small window, and then changing that contrast according to some function or transformation. The intensity of the central element of the local window is then recomputed according to the new contrast value for that pixel. Thus, the function or transformation determines the relationship between the output and input contrasts.

The local contrast of the original image at pixel (i, j) is defined as (Peli, 1990):

$$c_{ij} = \frac{|I_{ij} - I_0|}{|I_{ij} + I_0|} \qquad \text{EQ (9.1)}$$

where I_{ij} is the intensity of the pixel and I_0 is the local mean intensity. This is also known as Michelson's formula. Numerous transformations for this contrast have been proposed. Dhawan et al. (1986) investigate a number of transformation functions, including the tangent ($\tan(nc_{ij})$), hyperbolic tangent ($\tanh(nc_{ij})$), exponential ($1 - \exp(-nc_{ij})$), natural logarithm ($\ln(1 + nc_{ij})$), and the square root ($\sqrt{c_{ij}}$) transformations, where n is a real

number. The only necessary property of the transformation is that it must map the input contrast to output contrasts in the range of $[0\ 1]$, the extreme limits of the contrast value for any edge.

These transformations not only increase the contrast but they also increase the noise intensity, so the choice of the transformation function is usually a trade-off between the amount of contrast enhancement desired and the maximum allowable increase in noisc intensity. Later we show how the value of n can be chosen to maximise the output EEP. Whichever contrast transformation $F(c_{ij})$ is used, where $F(c_{ij}) \geq c_{ij}$ and $F(c_{ij}) \in [0, 1]$ for $c_{ij} \in [0, 1]$, the pixel's intensity is modified as

$$I'_{ij} = \begin{cases} I_0 \dfrac{(1 + c'_{ij})}{(1 - c'_{ij})} & \text{if } I_{ij} \leq I_0 \\ I_0 \dfrac{(1 - c'_{ij})}{(1 + c'_{ij})} & \text{otherwise.} \end{cases}$$

where $c'_{ij} = F(c_{ij})$.

9.2.4 Nonlinear Unsharp Masking

A classical contrast enhancement filter is the so-called *Unsharp Mask (UM)* which increases the contrast by adding a high-pass version of the input signal to itself, as we discussed above. Naturally, the edges are enhanced, but so is any noise present in the signal. Guillon et al. (1996) developed a new family of non-stationary filters, whose impulse response at each point depends upon the neighbouring pixels' intensities, i.e., it is adaptive to the input signal. The filter mask then processes the pixel by a combination of high-pass and low-pass filter versions of the input.

Consider a filter mask M of size $K \times L$ centred on the pixel (i, j). Each coefficient m_{ij}^{kl} of this mask is viewed as a level of confidence of the current pixel belonging to the mask. Thus, $m_{ij}^{kl} \to 1$ for pixels similar to the centre pixel and $m_{ij}^{kl} \to 0$ for others, and $M = \{m_{ij}^{kl} \in [0, 1] / (k, l) \in K \times L\}$.

Let I_{kl} be the intensity of pixel (k, l) in the mask M, and I_{ij} be the intensity at the current position, which is at the centre of the mask. The algorithm needs a discriminating function that tends to 1 when the pixel values are similar, and tends to zero otherwise. A suitable function is,

$$m_{ij}^{kl} = \exp\left(\frac{-(I_{kl} - I_{ij})^2}{2\sigma^2}\right)$$

where σ controls the width of the Gaussian curve. The mask M is then computed for each pixel in the image. The proposed filter structure is shown in Figure 9.2. The multiplier α is a weighted factor driving the contrast enhancement effect. The low and high pass versions of the input are computed as, respectively,

$$I_{ij}^{LP} = \frac{\sum_{(k,l) \in M} m_{ij}^{kl} I_{ij}}{\sum_{(k,l) \in M} m_{ij}^{kl}}$$

$$I_{ij}^{HP} = \sum_{(k,l) \in M} \left(m_{ij}^{kl} - \bar{m}_{ij}\right) I_{ij}$$

with $\bar{m}_{ij} = \frac{1}{KL} \sum_{(k,l) \in M} m_{ij}^{kl}$. This particular technique is called the *Gradient-Like Enhancement* (GLE) technique.

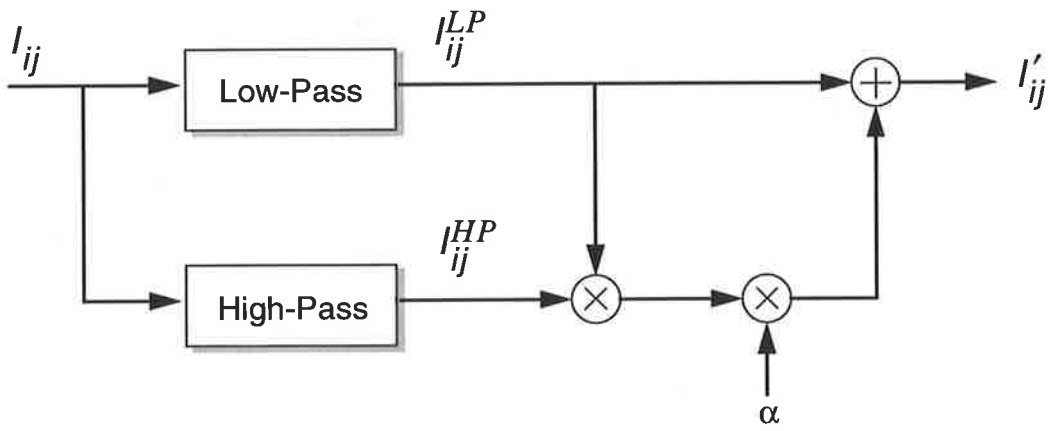


Figure 9.2 Gradient-Like Enhancement (GLE) as proposed by Guillon et al. (1996).

The low-pass component I_{ij}^{LP} is a weighted mean over the mask M. The high-pass component I_{ij}^{HP} can be interpreted as local estimate of the gradient at that pixel. The overall output of the system is

$$I'_{ij} = I_{ij}^{LP} \left(1 + \alpha I_{ij}^{HP}\right). \qquad \text{EQ (9.2)}$$

This techniques attempts to improve the performance over the usual Unsharp Masking algorithm by multiplying the low-pass version of the input with the high-pass version of the input, rather than multiplying the input directly with the high-pass version of the input. Thus, the same edge enhancement can be achieved with this method as with the classical

unsharp masking, but the effect of noise can be reduced. Later, we show how the value of α can be selected to maximise the output EEP.

9.3 Edge Enhancement Measures

We have seen that there exists a large number of image enhancement algorithms, there are few means or methods that can quantify the amount of enhancement. Many papers use visual inspection to compare different enhancement schemes - which is a qualitative measure and not a quantitative one.

Visual inspection is ultimately the best means of rating the improvement in the quality of an image compared to the original, but the means for determining the performance are not entirely adequate. For example, humans are subjective - people may measure the quality of an image in different ways. We desire a means of rapidly and automatically measuring the improvement in an image with good repeatability in the measures. For these reasons, we shall only consider quantitative measures of enhancement, and not subjective ones. Furthermore, the quantitative measure will not be based on any properties of the human visual system, but solely on the statistics of the image. The enhancement schemes investigated here will primarily enhance the contrast, or edges, of the input image, thus we review and describe only measures which relate to such enhancement schemes.

9.3.1 Enhancement Measures

Contrast Improvement Index

Assuming that there is a step edge in a local neighbourhood, the contrast at pixel (i, j) is defined as (Peli, 1990)[1]:

$$c = \frac{I_{max} - I_{min}}{I_{max} + I_{min}}$$

where I_{max} is the maximum background intensity of the edge, and I_{min} is the minimum background intensity of the edge. This is sometimes referred to as Michelson's formula.

1. The contrast, as defined here, measures the contrast in a small window centred on pixel (i,j), whereas the contrast defined in EQ (9.1) is the contrast of the actual pixel (i,j).

A popular measure of enhancement is to compute the contrast of the enhanced image, and compare it to the original image (Dash & Chatterji, 1991; Dhawan et al., 1986; Gordon & Rangayyan, 1984; Beghdadi & Le Negrate, 1989). Laine et al. (1994) define the contrast improvement index (CII) in a region of interest as:

$$CII = \frac{c_{out}}{c_{in}},$$

where c_{out}, and c_{in} are the contrast at an edge in the output (enhanced) image and the original input image, respectively.

Relative Edge Enhancement

Paradis & Jernigan (1994) used a measure called the relative edge enhancement (REE) for 1-D step edges. This measure is primarily used when the intensity of the edge points themselves increase relative to the background intensity. For enhancement of a step edge, as shown in Figure 9.3, Paradis & Jernigan (1994) define the REE as

$$REE = \frac{\Delta y_p}{\Delta y}$$

where Δy_p is the distance between the edge peaks, and Δy is the difference between the background intensities.

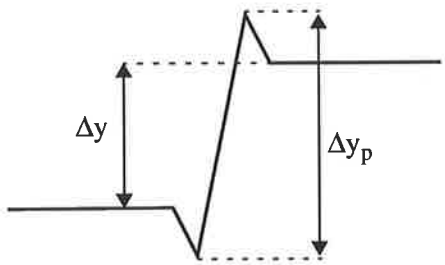


Figure 9.3 A step edge whose edge pixels are enhanced relative to the background.

When measuring the enhancement in an image compared to the original image we can use a variation of the above formula:

$$REE = \frac{\Delta y_p / \Delta y}{\Delta x_p / \Delta x}$$

where Δy_p is the peak-to-peak edge variation and Δy is the background difference in the vicinity of the enhanced edge in the enhanced image, while Δx_p and Δx are the corresponding measures for the unenhanced original input.

Gradient Enhancement Measure

Another possible image enhancement measure is the Gradient Enhancement Measure (GEM) as proposed by Harris (1977). This is simply a measure of the increase in the gradient of the edge after image enhancement.

9.3.2 New Performance Measure

For SICNN enhancement, we are only interested in the enhancement as shown in the edge in Figure 9.3, i.e., enhancement of the edge pixel intensity relative to the difference in the background intensities. The measures described above are inappropriate for measuring this type of enhancement. The CII measures only the enhancement of the background intensities - it does not take into account the enhancement of the edge pixels relative to the background. Conversely, the GEM only measures the enhancement of the edge pixel's gradient, with no account of the change in the background intensity. The REE measure is possibly the best of these measures as it takes into account both the edge responses and the background intensities, however it can increase without bounds as $\Delta y \to \infty$. Thus, we can make the REE arbitrarily large by simply forcing the difference in the background intensities (Δy) to be as small as possible. We would prefer a measure which gives a high value when both the edge enhancement and the difference between the background intensities are as large as possible.

A better measure for the type of enhancement that we desire, and related to the REE, is given by EQ (9.3), which we call the Edge Enhancement Product (EEP). The multiplier of 4 is a normalisation constant. The EEP measures the ratio of the peak-to-peak intensity difference (Δy_p) relative to the background intensity difference (Δy) as shown in Figure 9.3. By maximising this measure we make the enhancement of the edge pixel's intensity as large as possible and also keep the difference between the background intensity levels large too. Thus, by maximising this measure we trade-off edge enhancement with background contrast enhancement.

$$EEP = 4\left(1 - \frac{\Delta y}{\Delta y_p}\right)\frac{\Delta y}{\Delta y_p} \qquad \text{EQ (9.3)}$$

Figure 9.4 shows the EEP as the ratio $\Delta y/\Delta y_p$ varies. We assume here that the maximum value of the difference in background intensities does not exceed the peak-to-peak intensity magnitude, i.e., $\Delta y \leq \Delta y_p$. When Δy or the difference in the background intensities is zero, we have no background contrast, hence the EPP is zero. When $\Delta y/\Delta y_p$ is 1, then the background intensity levels are equal to the intensity levels of the edge pixels, thus there is no edge enhancement, and hence the EEP is also zero. Maximum enhancement occurs when $\Delta y = 0.5\Delta y_p$, in which case the difference in background intensities is exactly equal to half of the peak-to-peak intensity difference between the edge pixels. The constant of 4 in EQ (9.3) ensures that the maximum EEP is 1 and not $1/4$.

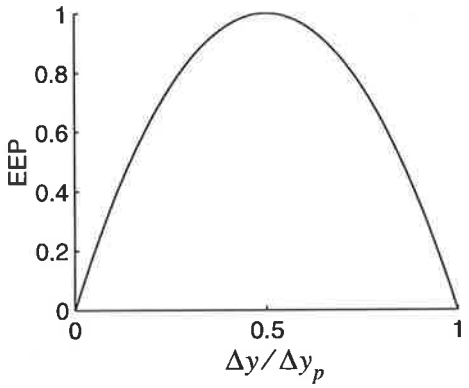


Figure 9.4 EEP as a function of the ratio of $\Delta y/\Delta y_p$.

9.4 The SICNN Edge Enhancer

In this section we present our analysis of the SICNN for edge enhancement. We begin with the recurrent SICNN and show that an iterative approach can be used to calculate its steady-state values. It is the output of this iterative approach that we can use for edge enhancement. Using the EEP measure defined in EQ (9.3), the SICNN decay factor that maximises the EEP of its output for 1 iteration is found. We then compare the optimal decay factor to the decay factor which maximises the experimental EEP of both 1-D and 2-D edges. Finally, we obtain by simulations the maximum EEP of the SICNN for different iterations, and discuss why it is advantageous to use the 1st iteration output only.

9.4.1 Defining the Edge-Enhancing SICNN

Much of the background theory used in this Chapter was presented in Section 4.2, so the reader is referred to that section of the thesis for more details.

The differential equation for each cell of the 1-D SICNN is given by EQ (4.1) and repeated here:

$$\frac{dx_i}{dt} = I_i - a_i x_i - \sum_{j \in N_r(i)} w_{ij} f(x_j) x_i \qquad i = 1, 2, \ldots, M \qquad \text{EQ (9.4)}$$

where x_i is the state of cell i, I_i is its input, a_i is its passive decay factor of excitation, f is the activation function, w_{ik} is the interaction weight between cells j and i, N_r is the neighbourhood function, and M is the total number of nodes, which corresponds to the total length of the input. Clearly, this equation is nonlinear as each cell state x_i depends upon the output states of its neighbours, who in turn are also a function of x_i. The steady-state solution to EQ (9.4) is given by EQ (4.2):

$$x_i = \frac{I_i}{a_i + \sum_{j=-r}^{r} w_j f(x_{i+j})} \qquad i = 1, 2, \ldots, M. \qquad \text{EQ (9.5)}$$

This equation can be solved using an iterative approach where an initial estimate of x_{i+j} in the denominator of EQ (9.5) is used to compute the value of x_i on the left-hand side of the equation, whose value in turn replaces the value of x_i in the numerator. Eventually, we expect this value of x_i to approach the steady-state value of EQ (9.4). This iterative approach can be cast as the discrete-time dynamical system as given in EQ (4.3), which is slightly modified here so that the time index k corresponds directly to the number of iterations:

$$x_i(k) = \frac{I_i}{a_i + \sum_{j=-r}^{r} w_j f(x_{i+j}(k-1))} \quad \text{for } k = 0, 1, 2, \ldots, \infty. \qquad \text{EQ (9.6)}$$

The initial value of $x_i(0)$ is chosen to be I_i, the input intensity to node i. Thus, our initial "guess" for the SICNN steady-state is its input intensity. Consider, for example, a constant input of intensity 10, and a SICNN with decay factor $a_i = 1$, neighbourhood size $r = 5$, and a symmetrical rectangular weight distribution. Figure 9.5 shows the value of $x_i(k)$ as

k increases. $x_i(k)$ initially oscillates greatly, but as k increases it eventually converges to its steady-state value.

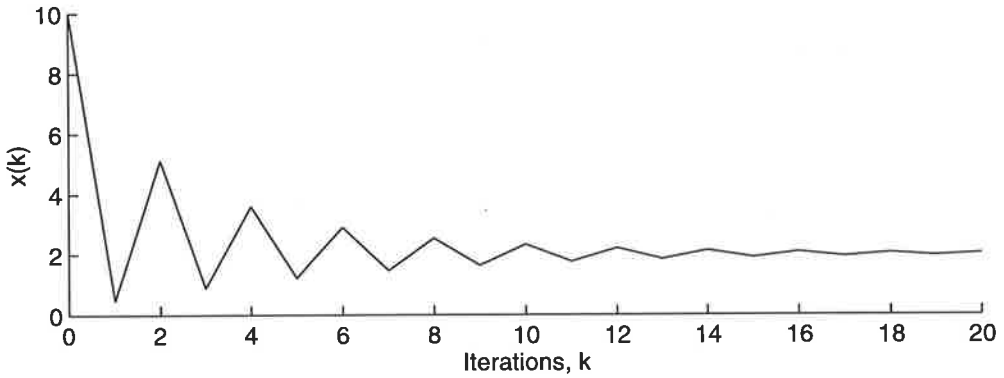


Figure 9.5 $x_i(k)$ as given by EQ (9.6) for different number of iterations. The input has $I_o = 10$, and the SICNN has a symmetrical rectangular weight distribution, $r = 5$ and $a = 1$.

9.4.2 Enhancement with 1 Iteration

We now look at the edge enhancement capabilities of the SICNN output after only 1 iteration, i.e., when $k = 0$ in EQ (9.6), which is, in fact, identical to the one that has already been used for edge detection. For a given input, by varying the decay factor, the shape of the its output can be varied, as shown in Figure 9.6. By varying the decay factor, both the edge enhancement and the background intensities of this output are varied. Thus, by appropriately selecting the value of the decay factor, an output which maximises the EEP measure as given in EQ (9.3) can be obtained.

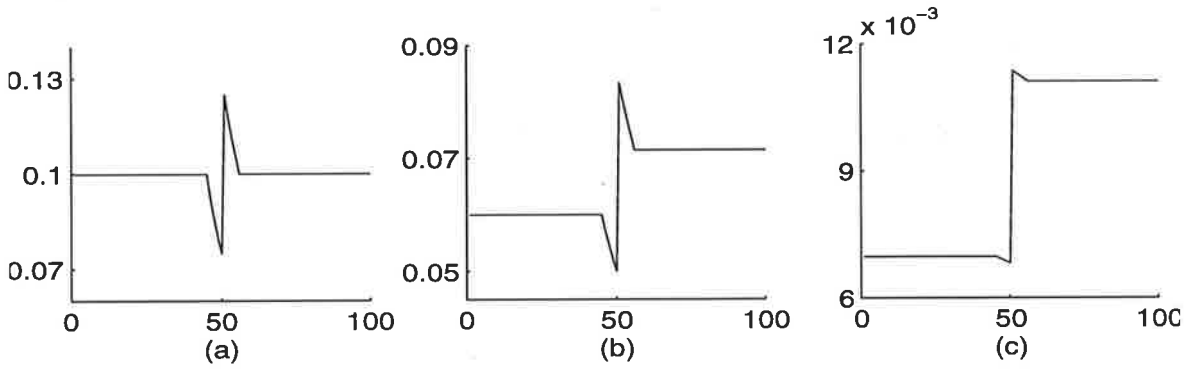


Figure 9.6 SICNN output when its decay factor is (a) $a \ll I_0$, (b) $a \approx I_0$ and (c) $a \gg I_0$, where I_0 is the mean input intensity.

9.4.2.1 Derivation of Optimal Decay Factor for 1 Iteration

We now derive an expression for the decay factor that maximises the EEP of the first iteration output. That is, we want to find the decay factor which maximises EQ (9.3) for a step edge input. Assuming a linear activation function and symmetrical weights, and from of the SICNN output as shown in Figure 5.7, the peak-to-peak edge intensity (Δy_p) and the difference in background intensities (Δy) are, respectively:

$$\Delta y_p = \frac{I_0(1+c)}{a + I_0 \sum_{j=-r}^{r} w_j} - \frac{I_0(1-c)}{a + I_0 \sum_{j=-r}^{r} w_j} = \frac{2cI_0}{a + I_0 \sum_{j=-r}^{r} w_j}$$

$$\Delta y = \frac{I_0(1+c)}{a + I_0(1+c) \sum_{j=-r}^{r} w_j} - \frac{I_0(1-c)}{a + I_0(1-c) \sum_{j=-r}^{r} w_j} = \frac{2acI_0}{\left(a + I_0(1+c) \sum_{j=-r}^{r} w_j\right)\left(a + I_0(1-c) \sum_{j=-r}^{r} w_j\right)}$$

The EEP is then given by

$$EEP = 4\left(1 - \frac{\Delta y}{\Delta y_p}\right)\frac{\Delta y}{\Delta y_p} = \frac{\left(4aI_0 \sum_{j=-r}^{r} w_j\right)\left(a + I_0 \sum_{j=-r}^{r} w_j\right)\left(a + I_0(1-c^2) \sum_{j=-r}^{r} w_j\right)}{\left(a + I_0(1+c) \sum_{j=-r}^{r} w_j\right)^2 \left(a + I_0(1-c) \sum_{j=-r}^{r} w_j\right)^2}.$$

In this equation we can maximise either w.r.t. the decay factor or the sum of weights. We choose to maximise w.r.t. the decay factor only since the sum of weights is normally fixed. Furthermore, if the maximum EEP is obtained by optimising w.r.t. the decay factor, then it is pointless to optimise w.r.t. the sum of weights, or indeed any other parameter. To find the value of the decay factor which maximises the EEP, we need to solve $dEEP/da = 0$. This solution is a quartic in a, and so has four possible solutions. Two of the solutions are complex conjugates, whilst the remaining two solutions are real and equal in magnitude, but they differ in polarity. The only non-negative, real root of *EEP*, and thus the optimal value of the decay factor, can be shown to be

$$a_{opt} = I_0\sqrt{1-c^2} \sum_{j=-r}^{r} w_j, \qquad \text{EQ (9.7)}$$

and the corresponding value of the EEP measure for this value of decay factor is

$$EEP = \frac{4\left(\sqrt{1-c^2}\right)\left(\sqrt{1-c^2}-1\right)\left(\sqrt{1-c^2}+1-c^2\right)}{\left(\sqrt{1-c^2}+1+c\right)^2\left(-\sqrt{1-c^2}-1+c\right)^2} = 1.$$

Thus, the maximum possible value of the *EEP* with the SICNN, for a 1-D step edge input, is equal to 1. In fact from Figure 9.4, this is the maximum EEP that we expect to get when $\Delta y/\Delta y_p = 0.5$, hence EQ (9.7) could have been obtained by solving $EEP = 1$ instead.

To check the validity of this derivation and equations, the optimal theoretical values of the decay factor and the EEP are compared to the values obtained experimentally. Figure 9.7(a) shows the EEP value for a step edge input as the decay factor of the SICNN is varied. The SICNN has a linear activation function, symmetrical weights that sum to unity, while the edge input has mean intensity $I_0 = 10$ and contrast $c = 0.25$. This gives the theoretically optimal values of $a = 9.68$ and $EEP = 1$. From Figure 9.7(a) in the $ENR = 10\,\text{dB}$ case, the optimal value of the decay factor is around 10, and the optimal value of the EEP is about 0.96. Clearly, these agree very well with the theoretical values. For an input with $ENR = 0$ dB, the optimal decay factor is again approximately 10, but the EEP is 0.80 which is less than the theoretical optimal value. This is because in our definition of the EEP we assumed that $\Delta y \le \Delta y_p$, but this may not be true when the input is very noisy. Hence the EEP for some edge may in fact be negative, which causes the overall EEP, when averaged over many edges, to be less than 1.

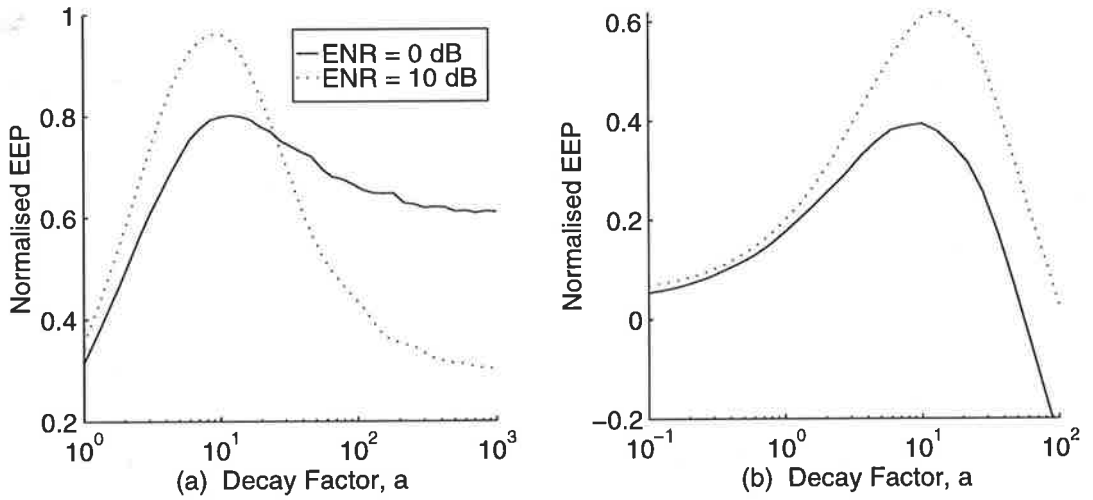


Figure 9.7 EEP as a function of the decay factor. (a) is for 1-D step edges with $I_o = 10$, $c = 0.25$, while the SICNN has symmetrical weights summing to 1, and $r = 5$. (b) is for the 2-D synthetic image. The SICNN has symmetrical 2-D weights summing to 1. In (a) and (b), the noise is multiplicative.

For the 2-D synthetic image shown in Figure 6.13(a), Figure 9.7(b) shows the EEP as the decay factor, which is the same for each pixel of the image, is varied. In 2-D, the EEP is calculated by computing Δy and Δy_p in small local regions about each edge pixel of the

image. The maximum EEP is about 0.63 for $ENR = 10$ dB. If, however, the decay factor is chosen according to EQ (9.7) in a small local neighbourhood, the overall EEP of the enhanced image for $ENR = 10$ dB is approximately 0.76. Thus, by using a locally optimal value of the decay factor, the edge enhancement is increased by approximately 20%.

9.4.3 Enhancement with Varying Iterations

We introduced in Section 9.4.1 the discrete sequence $x_i(k)$ given by EQ (9.6) and used for edge enhancement, as:

$$x_i(k) = \frac{I_i}{a_i + \sum_{j=-r}^{r} w_j f(x_{i-j}(k-1))} \qquad \text{EQ (9.8)}$$

where $k = 1, 2, \ldots, \infty$.

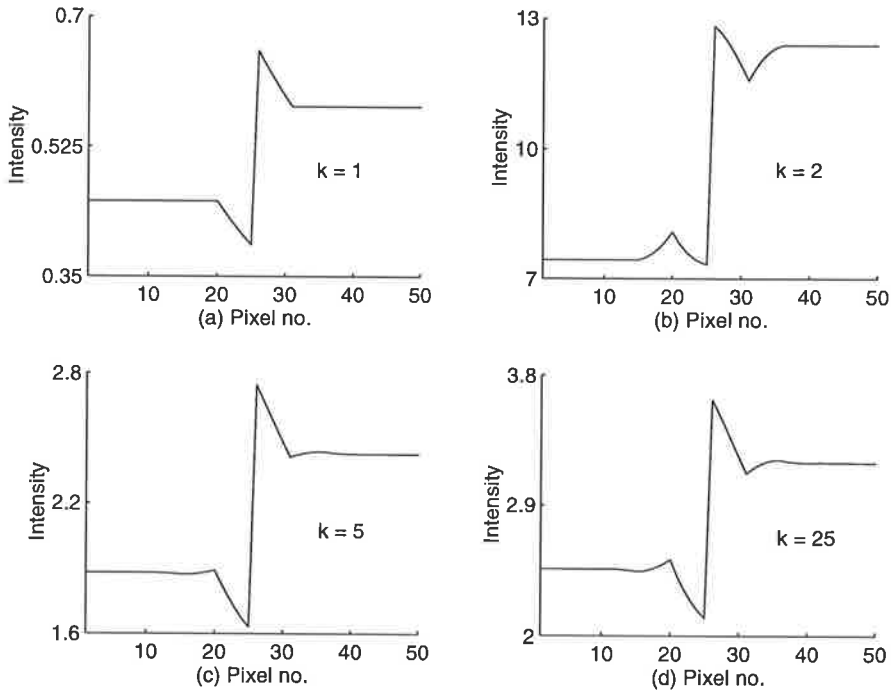


Figure 9.8 SICNN edge response for (a) $k = 1$ iteration, (b) $k = 2$ iterations, (c) $k = 5$ iterations, and (d) $k = 25$ iterations. The decay factor is chosen to maximise the EEP of the output edge. The input step edge has $I_o = 10$, and $c = 0.25$.

Figure 9.8 shows the SICNN output to a step edge input, as given by EQ (9.8), after 1, 2, 5, and 25 iterations. For each iteration, the decay factor is chosen to maximise the EEP of the enhanced output edge. From these values of the EEP, and from observing the output step edges, we can clearly see that the edge enhancement after 1 iteration of the SICNN is

almost the same as the enhancement after 25 iterations. The enhancement is only small for two iterations of the SICNN. The EEP for $k = 1, 5$ and 25 is 1, while for $k = 2$ it is 0.37.

Figure 9.9(a) shows the EEP of the SICNN output to a step edge input for different number of iterations, k. For the output of each iteration, the decay factor is again chosen to maximise the EEP. We showed in Section 9.4.2.1 how to analytically derive the optimal decay factor for 1 iteration ($k = 1$), however due to the nonlinear nature of the SICNN it is not possible to find an analytical expression for the EEP for $k > 1$. Thus, the optimal results shown in the Figure are obtained by simulations. Note that when $k = 0$, $x_i(k)$ is equal to the step edge input to the SICNN, thus the EEP is zero since there is no enhancement.

As observed from Figure 9.8, Figure 9.9(a) again indicates that the EEP for odd iterations is always equal to 1, while the EEP for even-numbered iterations increases asymptotically to 1. We note that the enhancement of the output, once it has reached steady-state, is exactly the same as the enhancement after 1 iteration. This fact, combined with the impossibility of deriving the optimal decay factor for $k > 1$ and the ease with which the optimal decay factor can be computed for 1 iteration, implies that the first iteration of the SICNN output can be used for edge enhancement without any loss in actual enhancement. Figure 9.9 (b) shows the simulated EEP values of the output as its decay factor is varied for 1, 2, and 25 iterations of EQ (9.8).

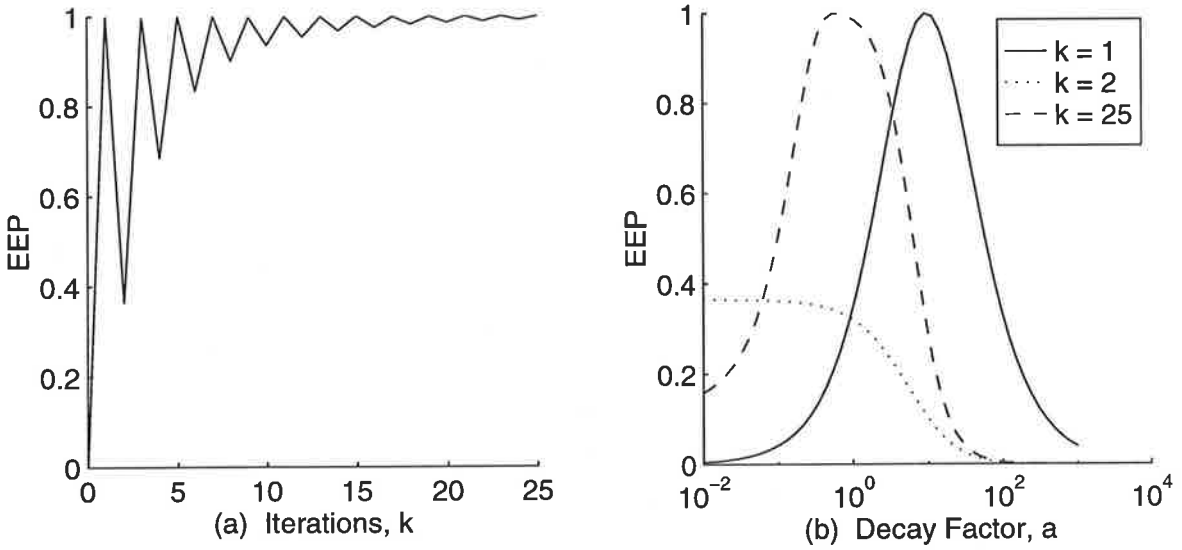


Figure 9.9 (a) The optimum EEP as a function of its decay factor, and (b) the EEP as a function of the decay factor for 1, 2, and 25 iterations. The input is a step edge with $I_o = 10$, and $c = 0.25$.

9.5 Comparison with Other Schemes

We compare the edge enhancement abilities of the SICNN with those of other edge enhancers for 1-D step edges with noise. The performance is measured in terms of the usual EEP of the output step edge. The four edge enhancers considered are

- a SICNN with a neighbourhood size of r, linear activation function, rectangular symmetrical weight distribution, and optimal decay factor as given in EQ (9.7),

$$a_{opt} = I_0\sqrt{1-c^2}\sum_{j=-r}^{r} w_j \qquad \text{EQ (9.9)}$$

- a SICNN with Gaussian pre-smoothing. The Gaussian has a support size of $2r+1$, with width $\sigma = r/3.5$. The other SICNN parameters are as for the SICNN with no smoothing.

- the gradient-like enhancer (GLE) as described in Section 9.2.4. In the same way that the optimal decay factor for the SICNNs was derived in Section 9.4.2.1, the value of α in EQ (9.2) which maximises the EEP of the GLE's output to a 1-D step edge is derived. This derivation of the optimal α is presented in Appendix E, and is given by EQ (E.1).

- hyperbolic tangent (*tanh*) contrast enhancement, $F(c_j) = \tanh(nc_j)$, which was described in Section 9.2.3. Using the same approach as for the SICNN optimal decay factor and the GLE's optimal α, the optimal value of n that optimises the output EEP is derived in Appendix E, and is given by EQ (E.2).

For the 1-D comparison, the GLE and *tanh* contrast enhancement schemes have support size $2r+1$, where r is the neighbourhood size of both SICNNs. The width of the Gaussian smoothing filter is also $2r+1$. For the 2-D comparison, the size of the SICNNs weight distribution is $(2r+1)\times(2r+1)$, where r is again the neighbourhood size. The support of the Gaussian smoothing filter, the GLE and *tanh* contrast enhancement schemes is also $(2r+1)\times(2r+1)$. The locally optimal decay factor, α and n for the SICNNs, the GLE and *tanh* contrast enhancement schemes, respectively, are computed as in the 1-D case by estimating the mean input intensity and contrast in a small local window of size $(2r+1)\times(2r+1)$ about each pixel.

One contribution of this Chapter is to provide means of estimating the optimal values of both α and n for the GLE and *tanh* enhancement schemes, respectively. The authors of both these edge enhancers did not give any sound basis for choosing the values of these parameters, even though these parameters greatly affect the edge enhancement properties; a means is needed of selecting them according to some criterion, rather than choosing them haphazardly. Our criterion of maximising the output EEP allows us to choose values of these important parameters for not only these schemes, but for a number of others which are not described; see Appendix E.

9.5.1 One Dimensional Results

Multiplicative Noise

Figure 9.10 shows the 1-D comparison for step edges with multiplicative noise. The step edge has overall length of 200 pixels. The neighbourhood size of both SICNNs is $r = 5$, hence the local window size of the GLE and *tanh* contrast enhancement operators is $2r+1 = 11$ pixels. All four edge enhancers have optimal parameters as explained above.

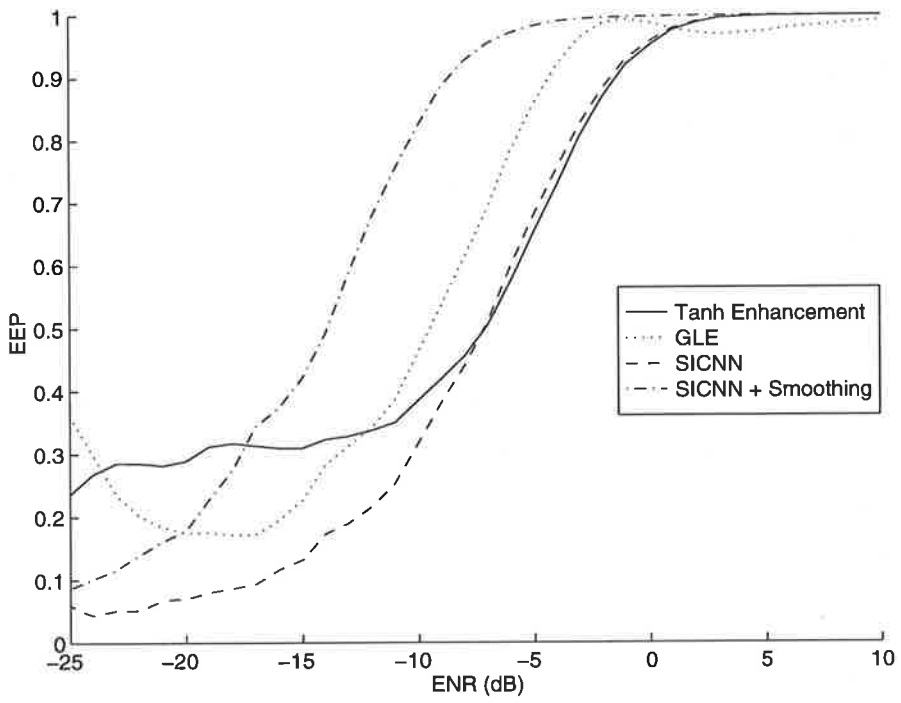


Figure 9.10 EEP comparison for the four edge enhancers as the ENR is varied. The input step edge has $I_o = 10$, $c = 0.25$, and multiplicative noise. See text for the parameters used for each edge enhancer.

For ENR greater than about -17 dB, the SICNN with smoothing gives the greatest EEP, followed by the GLE (except for very large ENR), the SICNN and finally the *tanh* contrast enhancer. For ENR less than about -15 dB, however, the input step edge is extremely noisy, so much so that it is hard to visually discern any edge at all in the input. Inputs with so much noise cause the enhancement algorithms to become somewhat numerically sensitive, hence the EEP of edge enhancers do not rapidly decrease to zero as the ENR is made very small. Simulations on a far greater number of edges would rectify the problem, but this is very time-consuming. Consequently, all results for very low ENR should be discarded as being insignificant, since they do not give us any insight into edge enhancement properties of the enhancers.

Thus, the SICNN with smoothing appears to be the best for edge enhancement, followed by the GLE algorithm. We note that both these edge enhancers have some form of smoothing which enables them to achieve superior enhancement to either the SICNN with no smoothing and the *tanh* enhancement operator.

Additive Gaussian Noise

Figure 9.11 shows the results for step edge with Gaussian noise. Once again, the SICNN with smoothing is by far the best enhancer, followed by the GLE, the *tanh* contrast enhancer and finally the normal SICNN. For ENRs between -20 and -10 dB, the SICNN outperforms the *tanh* contrast enhancer. As with the multiplicative noise results above, however, the EEP of the edge enhancers for very low ENR are not very meaningful.

Uniform Noise Result

Finally, Figure 9.12 shows the EEP comparison for step edge inputs with uniform noise. Again, the EEP of the SICNN with smoothing is vastly superior to all three remaining edge enhancers. The second best edge enhancer is the GLE, followed by the *tanh* contrast enhancer, and finally the normal SICNN.

In summary, we showed that for 1-D step edges with either multiplicative or additive noise, the SICNN with smoothing is always the superior edge enhancer, followed by the GLE method. For multiplicative noise, the SICNN outperforms the *tanh* contrast enhancer, whereas for additive noise, the *tanh* contrast enhancer outperforms the normal SICNN. Clearly, from the superior enhancement of the SICNN with smoothing and the

GLE method (which also has smoothing), filtering the input to remove some of the noise can improve the edge enhancement performance enormously.

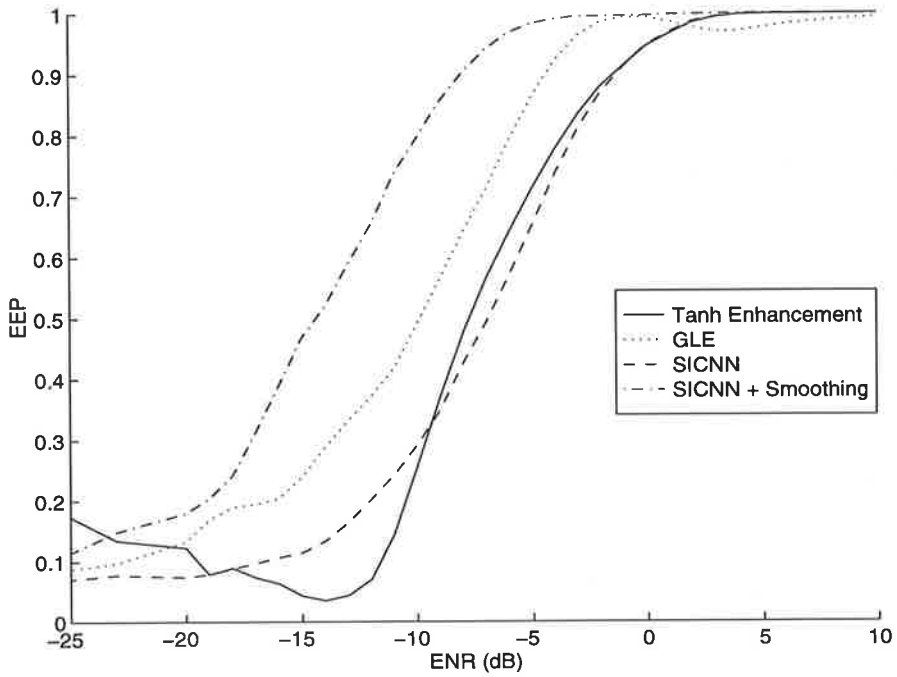


Figure 9.11 EEP comparison for the four edge enhancers as the ENR is varied. The input step edge has $I_o = 10$, $c = 0.25$, and additive Gaussian noise.

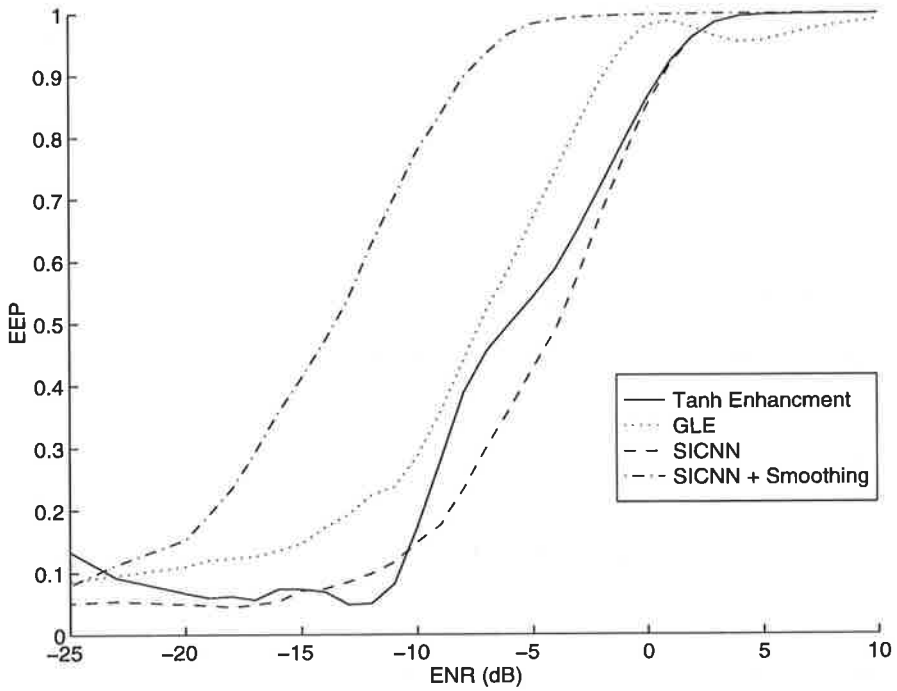


Figure 9.12 EEP comparison for the four edge enhancers as the ENR is varied. The input step edge has $I_o = 10$, $c = 0.25$, and additive uniform noise. See text for the parameters used for each edge enhancer.

9.5.2 Two-Dimensional Comparison

The above 1-D enhancement comparison is now extended to the 2-D synthetic image shown in Figure 6.13. The edge enhancers compared are identical to those used for the 1-D step edge case: the SICNN with and without smoothing, the GLE and *tanh* contrast enhancement schemes.

9.5.2.1 Synthetic Image Simulations

Multiplicative Noise

Our comparison begins with the synthetic image with multiplicative noise. In the results, all the EEP values below zero are truncated to zero; we now explain the reasoning behind this. We know that, theoretically, for a clean input the smallest EEP is zero, which occurs when there is no enhancement of the input image edges. For the 2-D synthetic image, the EEP is estimated around every edge pixel using a window size of approximately $(2r+1) \times (2r+1)$. Thus, when the input is very noisy, the output of the enhancer is also noisy, and this causes the locally-computed EEP to become numerically sensitive, or to vary considerably, so much so that it can be negative for low ENR. So, we do not display the negative EEP values since the actual enhanced output images are so noisy that the EEP values do not convey any significant information. This problem can be avoided if we simulated on thousands of images, but this is too time-consuming.

From Figure 9.13 for low ENRs, the performance of the SICNN is clearly the best, which is expected for Gaussian pre-smoothing. Following this is the SICNN, while both the *tanh* contrast enhancement and the GLE schemes perform badly for these low ENRs. As the ENR increases, the performance of the *tanh* contrast enhancement scheme rapidly increases to its maximum value, as eventually the GLE scheme does too. For very large ENR, both the *tanh* contrast enhancement and GLE schemes have larger EEP than either SICNNs.

The maximum EEP of each edge enhancer is different due to the difference in EEP values at the intersection points of the edges in the synthetic image. These corner pixels can have neighbouring regions with up to four different intensities. This in particular affects the SICNN because its output intensity is nonlinearly related to the intensity in its local neighbourhood. In fact, in the SICNN algorithm, the local intensities about the edge pixel are multiplied with the weights and then summed, hence the corner pixels are inhibited by

pixels with up to four different intensities. This causes the SICNN EEP to be less than 1 for large ENR. For the SICNN with smoothing we have the added effect of edge blurring which reduces the sharpness of the enhanced edges, thereby causing the EEP to be even smaller than the normal SICNN EEP. For the *tanh* contrast enhancement scheme, we only need to estimate the local contrast in the input, hence the output EEP almost reaches 1. For the GLE, the EEP slowly increases to 1 as there is some form of smoothing too in this case.

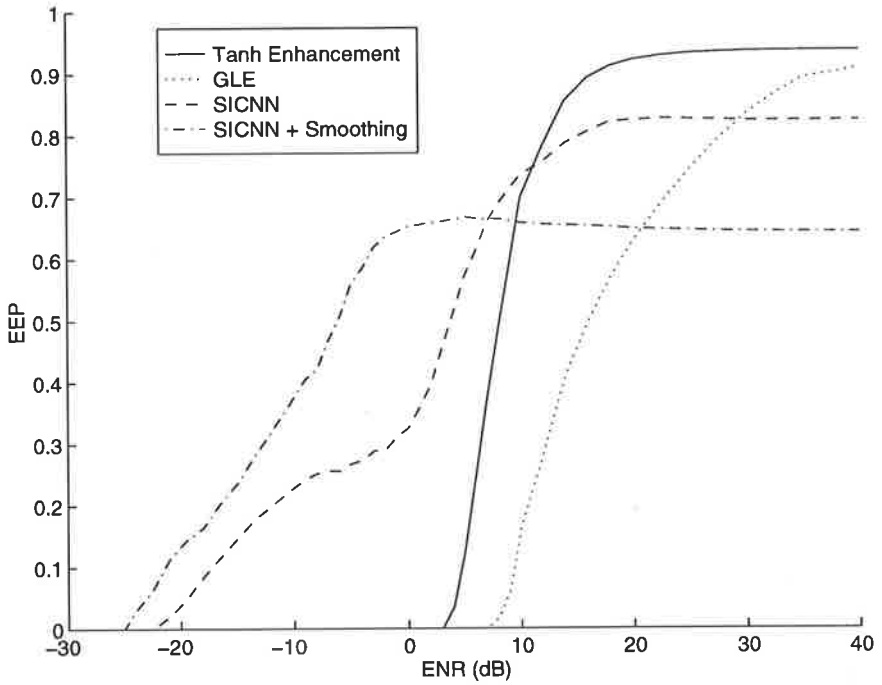


Figure 9.13 Comparison of the EEP for four different edge enhancers for the input synthetic image with multiplicative noise. See text for further details.

Thus, for low ENR the SICNN with smoothing performs best followed by the SICNN, and then the *tanh* contrast enhancement and GLE schemes, whereas for very large ENR, the *tanh* contrast enhancement scheme has the best performance followed by the GLE, SICNN and SICNN with smoothing.

Gaussian Noise Results

As for the results for multiplicative noise, all negative EEP values are truncated to zero. Again, for low ENR the SICNN with smoothing clearly has the best EEP, followed by the SICNN and the GLE scheme, and finally the *tanh* contrast enhancement scheme. As the ENR increases, the EEP of the *tanh* contrast enhancement increases rapidly, such that for

very large ENR the *tanh* contrast edge enhancer has the greatest EEP, followed by the GLE, SICNN and then the SICNN with smoothing. The difference in EEP values for large ENR was discussed above. The only major difference between the results here and those shown above for multiplicative noise is that for small ENR, the EEP of the GLE scheme is greater than zero and greater than the EEP of the *tanh* contrast enhancement scheme.

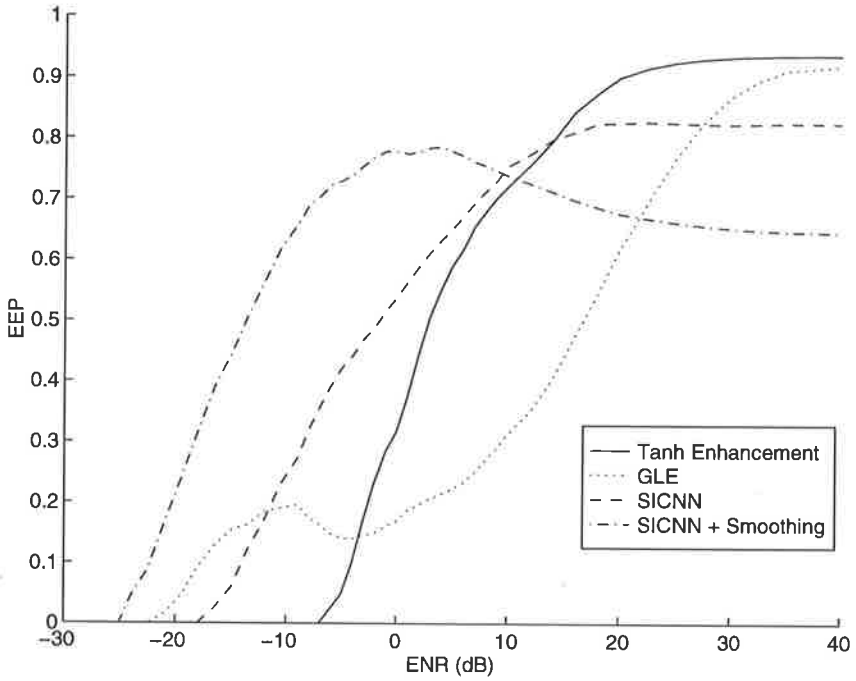


Figure 9.14 Comparison of the EEP for four different edge enhancers for the input synthetic image with Gaussian noise. See text for further details.

Uniform Noise Results

The enhancement results of the input synthetic image with uniform noise is very similar to that for Gaussian noise. Again all EEP values which lie below zero for very low ENRs are truncated; the enhanced images are so noisy at these ENRs that the numerical value for the EEP are essentially meaningless. Once again for inputs with very small ENR, the SICNN with smoothing performs best followed by the SICNN. The GLE scheme has a slightly non-zero EEP, whereas the contrast enhancement scheme has zero EEP. As the ENR increases, however, the EEP of the *tanh* contrast enhancement scheme rapidly increases to its maximum value. In fact, for very large ENR, the *tanh* enhancement scheme performs best followed by the GLE scheme, the SICNN and finally the SICNN with smoothing.

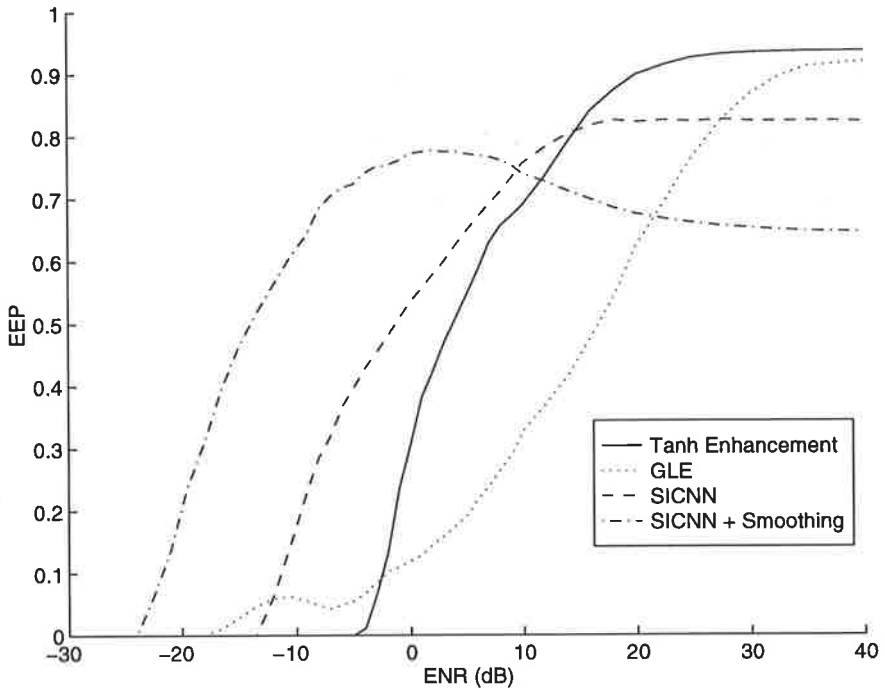


Figure 9.15 Comparison of the EEP for four different edge enhancers for the input synthetic image with uniform noise. See text for further details.

9.5.2.2 Real Image

The above comparison on 2-D synthetic images is now extended to that of an image of a real scene. We investigate the "Gatlin" image, as shown in Figure 9.16, consisting of three men in front of a quite detailed background. We note that very little detail of the jackets of two of the men is visible. The edge enhancers compared are exactly as above and the neighbourhood size of each SICNN is $r = 5$, hence the support size of the other two edge enhancers is $2r + 1 = 11$.

Figure 9.16 "Gatlin" image used to test the edge enhancers.

The procedure is to first determine the edges in the input image, which can be done with a simple Sobel operator. Then in small windows of size $(2r+1) \times (2r+1)$ at each edge pixel, the optimal parameters for each scheme are estimated. For the two SICNNs, the local mean intensity and contrast are estimated to obtain the optimal decay factor given by EQ (9.9), the local intensity and contrast are estimated to compute the GLE's optimal α using EQ (E.1), and the local contrast is estimated for the optimal n for the *tanh* enhancement scheme using EQ (E.2).

From the results of the edge enhancers on the Gatlin image shown in Figure 9.17, clearly the output of the SICNN with smoothing is of the poorest quality. The image is very blurry, which is expected since the input is smoothed with a Gaussian filter. The output of the *tanh* contrast enhancement scheme in (a) is better than that of the SICNN with smoothing, while the output of the GLE scheme and SICNN as shown in (b) and (c) respectively, are very similar in quality.

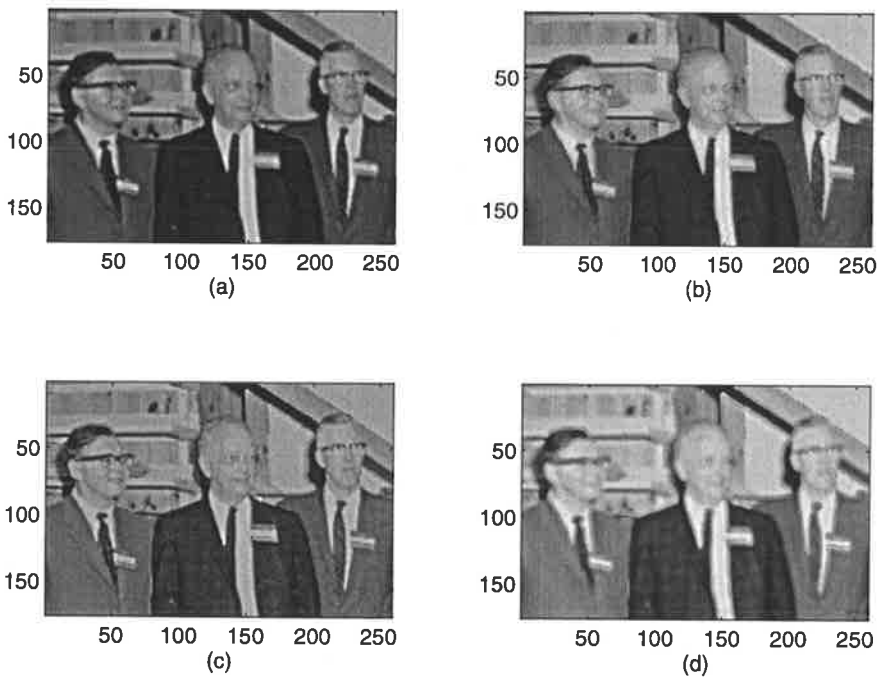


Figure 9.17 Output to the "Gatlin" image for (a) the *tanh* contrast enhancement scheme, (b) GLE scheme, (c) SICNN, and SICNN with smoothing. See text for more details.

The overall output of the GLE scheme is slightly brighter than that of the SICNN, which gives it a more pleasant visual appearance, however, the output of the SICNN clearly shows the patterns on the suit of the man in the centre and the man immediately to his

right. Furthermore, many of the features on these two suits, such as the collars and features defined by edges, are clearly more visible in the SICNN outputs than the output of the GLE scheme. Thus, overall the output of the SICNN is arguably better than the output of the GLE, though such a comparison is subjective. We point out that these schemes only enhance the edges of the input image, and not the entire image itself. So, we should not expect an improvement of the entire image's quality, rather just the edges or sharpness of the image.

9.6 Conclusions

We began this Chapter by reviewing a number of different image enhancement schemes and algorithms, with particular attention to the contrast transformation methods and nonlinear unsharp masking. The image enhancement measure used in the literature were reviewed and found to be unsuitable for our needs, so the Edge Enhancement Product (EEP) was defined, which measures both the enhancement of the edge pixels' intensities and the difference in the background intensities.

We then proceeded to derive the discrete-time sequence whose steady-state equals the recurrent SICNN steady-state output. For the output after 1 iteration, an expression for the SICNN decay factor which maximises its output EEP was derived. We then investigated the maximum EEP of the SICNN for different iterations and found that all odd-numbered iteration outputs have the maximum EEP, while the even-numbered iterations output's EEP gradually increases to the same maximum value. Thus, the output after 1 iteration has an EEP greater than or equal to the EEP of the output after 1 iteration, implying that there is no obvious advantage in using the SICNN output after more than 1 iteration.

Using the same approach as for the SICNN optimal decay factor, the expressions for the free parameters of the GLE and contrast enhancement schemes which maximised their output EEP were derived; these are presented in Appendix E.

Next we compared the 1-D EEP performance of the SICNN, the SICNN with smoothing, the GLE and contrast enhancement schemes. In general, the SICNN with smoothing performed best followed by the GLE, and finally the SICNN and *tanh* contrast enhancers. For the 2-D synthetic image with low ENR, the SICNN with smoothing performed best

followed by the SICNN, with both the GLE and contrast enhancement schemes having bad performance. For large ENR, the *tanh* contrast enhancement scheme had the greatest EEP followed by the GLE, then the SICNN and finally the SICNN with smoothing. For the real "Gatlin" image, the SICNN with smoothing and the *tanh* contrast enhancement scheme both did not enhance the edges in the image very well, in fact the quality of their output was quite poor. The outputs of the GLE scheme and the SICNN appeared roughly equal in quality; the GLE's output was slightly brighter and thus a little more clearer, whereas in the SICNN output many of the features of the jackets and overall image were enhanced, became much more clearer, and showed more detail.

Chapter 10 Conclusions

10.1 Summary

In this thesis we studied the design and application of shunting inhibitory cellular neural networks to the problem of edge detection and enhancement. The work presented in this thesis is summarised below.

- In Chapter 1, we briefly introduced vision, shunting inhibition, and the shunting inhibitory cellular neural network (SICNN) for edge detection and enhancement.
- Chapter 2 presented a description of the underlying physical processes involved in the generation of intensity or edge profiles, and then described the many applications of edge detections in industry, consumer electronics and other computer vision systems. A comprehensive review of edge detectors was also presented.
- Chapter 3 gave an overview of cellular neural networks, their different implementations and applications. The derivation and stability analysis of the SICNN was also reviewed and discussed.
- In Chapter 4, we investigated the response properties of both recurrent and feedforward SICNN systems. The convergence and step edge response of the recurrent SICNN were investigated, while the step edge response of the recurrent SICNN was discussed in detail. By linearising the feedforward SICNN, its impulse response was derived and using this, a detailed analysis of the SICNN response to noise was performed. The feedforward SICNN was then used throughout the remaining Chapters of this thesis.

- Chapter 5 introduced and discussed the application of the SICNN to edge detection. First, we defined three performance measures for edge detectors, namely the HR, ESD, and EB. These measures were used to described the effect of the SICNN output noise variance, shape of edge response and PNR on its performance. We then investigated the effect of the SICNN parameters, i.e., the decay factor and weights, on these factors and hence the edge detection performance.

- In Chapter 6, the SICNN decay factor and weight distribution that maximised the 1-D edge detection performance according to certain criteria were derived. The SICNN performance with various nonlinear activation functions was also investigated.

- Chapter 7 presented a number of postprocessing methods to improve the SICNN edge detection performance. The techniques were mainly concerned with combining the outputs of different SICNNs to improve the performance. Two edge tracking methods were introduced, where the edges in the SICNN output are tracked as the neighbourhood size or weight distribution width are decreased. We also investigated the combination of SICNN outputs with various permutations of weight distributions and thresholding schemes. Finally, we examined postprocessing the output edge map to eliminate spurious edges in small, local window. Most of these postprocessing techniques improved the performance of the SICNN.

- In Chapter 8, the SICNN edge detection performance was compared with the differentiator and Deriche operator on both 1-D and 2-D synthetic edges, as well as two real images.

- Finally, in Chapter 9 the edge enhancement properties of the SICNN was investigated. We reviewed a number of edge enhancement algorithms and measures, and then proposed a new edge enhancement measure, the EEP. The decay factor and the parameters of the other edge enhancers that maximise the EEP were derived. Finally, the edge enhancement performance of the SICNN (with and without smoothing) was compared with the GLE and *tanh* contrast enhancement schemes on both synthetic edges and a real 2-D image.

10.2 Results and Contributions

The results and contributions of knowledge of this dissertation are:

- the application of a feedforward cellular neural network to edge detection, and a description of how the inhibitory effects of the network on the input allow it to detect, or enhance, intensity discontinuities.

- The impulse response of the SICNN was derived by linearising the feedforward SICNN using perturbation analysis. This impulse response was shown to be a nonlinear function of the mean input intensity, among other parameters, hence indicating the ability of the SICNN to adapt itself to its input signal.

- Using the SICNN impulse response, the transfer function was derived. Also with this impulse response, we derived an expression for the SICNN output noise variance for a noisy DC input, and showed that this equation is valid for all but very low SNR, small neighbourhood sizes, small decay factor, large sum of weights, and large mean input intensities (for a given SNR). We also derived a theoretical expression for the SICNN noise figure (NF) for a linear and nonlinear activation function, and showed that the theoretical NF compared very closely to the experimental values for medium to high SNR inputs.

- By considering the intensity of each of the SICNN output pixels as a random variable, the output pdf for each pixel was derived, as were the expressions for the hit rate, probability of detection and false alarm rate. These theoretical values closely matched the experimental ones for different noise types, and even for low ENR inputs.

- The SICNN 1-D edge detection performance depends upon three factors: the output noise variance, the shape of the edge response, and the PNR. Increasing the output noise variance increases the ESD, while decreasing the width of the edge response increases the HR. Although we could not find an explicit relationship between the PNR and performance measures, the shape of the PNR does provide us with a good indication of the shape of the PNR curves, such as where the performance increases, peaks and levels off. We also showed that it is the ratio of the sum of weights and mean input intensity to the decay factor which determines the performance. For a given sum

of weights and mean input intensity, we experimentally showed that one particular decay factor gives zero EB and also optimises the HR and ESD. This optimal decay factor was shown to be close to the decay factor that maximises the PNR.

- We observed that reducing the sum of squared weights (SSW) reduces the SICNN output noise variance, which in turn decreases the ESD. Making the edge response or weight distribution as narrow as possible by increasing SSW, however, increases the HR. The shape and SSW of the weight distribution have conflicting effect on the performance. Using a constrained optimisation approach with Lagrange multiplier method, we numerically derived the weight distribution that attempts to simultaneously optimise the HR and ESD. The resultant optimal weight distribution has one free parameter that determines the shape of the weight distribution, and provides a trade-off between good HR and good ESD.

- We defined the optimal decay factor as the value that results in the SICNN having zero EB, i.e., the decay factor which ensures that, on average, the detected edge is not biased to the right or left of the true position. For multiplicative noise, the optimal value was shown to be zero. For additive noise, the optimal decay factor was proportional to the product of the noise standard deviation and sum of weights. The constant of proportionality, λ, was then empirically derived for different edge, noise, and SICNN parameters for both 1-D and 2-D synthetic edges.

- A number of postprocessing methods were presented which significantly improved the SICNN performance. In scale-space processing, the edges in the SICNN output are tracked as the neighbourhood size decreases, while in the variable weight combination scheme, the edges are tracked as the width of the weight distribution is slowly reduced. Both of these two methods start with low ESD and HR and then gradually increase the HR whilst maintaining the ESD at its low value. In 2-D, the area under the PD vs. FA curve was maintained or increased, with a distinct improvement in the appearance of the resultant output edge map. Combining the outputs of SICNNs with different weight distributions and thresholding schemes was also investigated, with the output of one SICNN limiting the search region for the edges of the second SICNN. For 1-D edges only, detecting edges in the SICNN with asymmetrical weights (and maximum thresholding) with the SICNN with symmetrical weights (and zero-crossing thresholding) significantly improved the performance. For inputs with low contrast,

combining the outputs of SICNNs with identical but reversed weights, improved the performance too. In neighbourhood processing, the visual appearance of the edge map was improved by eliminating spurious edges in small, local windows.

- Finally for edge detection we compared the performance of various edge detectors on the synthetic image. For multiplicative noise, the SICNN with smoothing performed better than the normal SICNN, discrete differentiator and Deriche operator. Both SICNNs performed better, as their impulse response is nonlinearly dependent upon the local mean input intensity, and hence they are well suited to overcome the effect of multiplicative noise where the noise strength is proportional to the input intensity. For additive noise, the SICNN with smoothing and the Deriche operator appeared to perform equally well. For the Lenna image, the SICNN detected the most edges although its output was quite noisy. For the SAR image, the output of the SICNN with smoothing was clearly better than any of the other edge detector outputs.

- A new edge enhancement measure, the EEP was proposed, which measures the difference in the edge pixel intensities and the difference in the background intensities for a step edge. Using this measure, the decay factor which maximises the SICNN EEP was derived. This criterion of maximising the EEP was also used to derive optimal parameters for a number of other standard edge enhancers, such as the GLE and *tanh* contrast enhancement schemes.

- The EEP of the SICNN output after 1 iteration was greater than or equal to the EEP of the output for more than 1 iteration. Furthermore, the equations for the optimal decay factor, for enhancement, could be derived, whereas this is not possible for more than 2 iterations. Thus, the output after 1 iteration is the best for edge enhancement.

- For 1-D edges with both multiplicative and additive noise, the SICNN with smoothing performed better than the normal SICNN, GLE and *tanh* contrast enhancement schemes. For the 2-D synthetic image, the SICNN with smoothing performed best for low ENR, whereas the *tanh* contrast enhancement scheme gave the best enhancement for large ENR. For the real image investigated, both the SICNN and the GLE scheme had good outputs but in different respects, though overall the SICNN produced greater enhancement of the edges.

10.3 Future Work

Although this thesis has presented a comprehensive review of the applications and design of the SICNN for edge detection and enhancement, there is still much work to do. We now outline a number of issues which need to be addressed in order to further our understanding of the SICNN and also to improve its edge detection and enhancement performance. Possible future work includes:

- **Deriving an analytical expression for the optimal** λ: we derived the optimal SICNN decay factor for additive noise, and showed that it is proportional to the input noise standard deviation and sum of weights. However, we could not derive an exact analytical expression for constant of proportionality, λ. Rather, for both 1-D and 2-D synthetic edges, its value was empirically computed for different edge and SICNN parameters. For real images the value of λ needs to be guessed, although it is not difficult to choose an adequate value by systematically trying a small number of different values. For real and complex synthetic images, only a finite number of values of λ can be tested for computational reasons, so we are never fully certain that the value of λ is in fact optimal. Thus, it would be very beneficial if an analytical expression could be derived for λ which related it to the input edge parameters, such as the mean intensity and the contrast.

- **Nonlinear activation function**: We investigated a small number of nonlinear activation functions and showed that they did not improve the SICNN edge detection performance. In fact, these activation functions made the performance worse than with the linear function. A more detailed investigation, however, could reveal why these activation functions were unsuitable, and if indeed there does exist an optimal activation function, apart from those tested here.

- **Optimising the other SICNN parameters**: We have observed throughout this thesis that the SICNN has many free parameters, each of which can affect the edge detection performance in a different way. Further work is needed to investigate the possibility of adaptively selecting these parameters to maximise the performance. Although we have presented some means of adaptively choosing the SICNN decay factor, we have not investigated at all how to adaptively choose the weight function and neighbourhood size according to the characteristics of the input image. This would enable the SICNN to be applied to a larger range of inputs with less intervention from the user.

- **Variable Gaussian smoothing**: For the SICNN performance with the synthetic 2-D image, we observed that Gaussian smoothing the input improved the performance, particularly for low ENR. Thus, the performance could be improved by varying the amount of smoothing according to the amount of noise present in the input. The input noise could be easily estimated using a number of techniques, such as Wiener filtering, and the noise strength could then be used to determine the width of the Gaussian filter. For example, the filter's width could be made proportional to the noise variance.

- **2-D implementation of the SICNN**: in the 2-D implementation, we applied the 1-D SICNN to each of the four orthogonal directions of the image, and then combined these outputs to obtain the overall output. There may, however, be advantages in applying only one SICNN with isotropic weight distribution, in which case the SICNN is equally likely to detect edges in all directions. This would, however, restrict us to using only a symmetrical weight distribution and zero-crossing thresholding scheme.

References

Abdou, I.E., & Pratt, W.K. (1979). "Quantitative Design and Evaluation of Enhancement/ Thresholding Edge Detectors." *Proc. of the IEEE*, vol. 67, no. 5, pp. 753–763, May.

Adamson, T.A., (1992). *Electronic Communications*. 2nd Edition, Delmar Publishers, Inc.

Ala, S.R., Chamberlain, D., & Ellis, T.J. (1992). "Real Time Inspection of Masonry Units." *IEE Int. Conference on Image Processing and its Applications*, no. 354, pp. 181-184, Maastricht, The Netherlands, April.

Appelhans, P. & Schroder, H. (1995). "Ghost Cancellation for Stationary and Mobile Television." *IEEE Trans. on Consumer Electronics*, vol. 41, no. 3, pp. 472-475, August.

Azevedo, S. & Longini, R.L. (1980) "Abdominal-Lead Fetal Electrocardiographic R-wave Enhancement of Heart Rate Determination." *IEEE Trans. on Biomedical Engineering*, vol. 27, no. 5, pp. 255-260, May.

Barlow, H.B. (1981). "The Ferrier Lecture. Critical Limiting Factors in the Design of the Eye and the Visual Cortex." *Proc. R. Soc. Lond.*, vol. B212, pp. 1–34.

Barnes, N. & Liu, Z.Q. (1995). "Vision Guided Circumnavigating Autonomous Robots." *Third Int. Computer Science Conference: Image Processing Applications and Computer Graphics*, HongKong.

Barone, A., Balsi, M., & Cimagalli, V. (1993). "Polynomial Cellular Neural Networks: A New Dynamical Circuit For Pattern Recognition." *Proc. of Int. Specialist Workshop on Nonlinear Dynamics of Electronic Systems*, July Dresden, Germany.

Batchelor, B.G., & Braggin, D.W. (1992). *Commercial Vision Systems*, in "Computer Vision: Theory and Applications.", Torras, C (ed.) pp. 405-452, Springer-Verlag.

Beghdadi, A. & Le Negrate, A. (1989) "Contrast Enhancement Technique based on Local Detection of Edges." *Computer Vision, Graphics and Image Processing*, vol. 46, pp.

162–174.

Bergholm, F. (1987) "Edge Focusing." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, no. 6, pp. 726–741, November.

Berzins, V. (1984). "Accuracy of Laplacian Edge Detectors." *Computer Vision, Graphics and Image Processing*, vol. 27, pp. 195–210.

Bouzerdoum, A. (1991). *Nonlinear Lateral Inhibitory Neural Networks: Analysis and Application to Motion Detection*. PhD thesis, University of Washington, July.

Bouzerdoum, A. (1994). "A Hierarchical Model for Early Visual Processing." *Proceedings of SPIE on Human Vision, Visual Processing, and Digital Display V*, vol. 2179, pp. 10–17, 8-10 February, San Jose, California, USA.

Bouzerdoum, A., & Pinter, R.B. (1993). "Shunting Inhibitory Neural Networks: Derivation and Stability Analysis." *IEEE Trans. on Circuits and Systems-I*, vol. 40, no. 3, pp. 215–221, March.

Boyer, K.L. & Sarkar, S. (1994). "On the Localization Performance Measure and Optimal Edge Detection." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 106–108, January.

Batchelor, B.G. & Braggin, D.W. (1992). *Commerical Vision Systems*, in "Computer Vision: Theory and Applications.", Torras, C (ed.), pp. 405-452, Springer-Verlag.

Brzackovic, D., Patton, R., & Wong, R.L. (1991). "Rule Based Multi-Template Edge Detection." *Computer Vision, Graphics and Image Processing*, vol. 53, no. 3, pp. 258–268.

Brodie, S.E., Knight, B.W., & Ratliff, F. (1978). "The Spatiotemporal Function of the Limulus lateral Eye." *J. General Physiology*, vol. 72, pp. 167–202.

Canny, J. (1986). "A Computational Approach to Edge Detection." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, November.

Chao, C-H. & Dhawan, A.P. (1994a). "Edge Detection Using Hopfield Neural Network." *Proc. of SPIE Conference on Applications of Artificial Neural Networks V*, vol. 2243, pp. 242–251, Orlando, Florida, April.

Chao, C-H. & Dhawan, A.P. (1994b). "Edge Detection using a Hopfield Network." *Optical Engineering*, vol. 33, no. 11, pp. 3739–3747, November.

Chen, J.S., Huertas, A., & Medioni, G. (1987). "Fast Convolution with Laplacian-of-Gaussian Masks." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 584–590, July.

Chua, L.O. & Wu, C.W. (1992). "On the Universe of Stable Cellular Neural Networks." *Int. J. of Circuit Theory and Applications*, vol. 20, pp. 497–517.

Chua, L.O. & Yang, L. (1988). "Cellular Neural Networks: Theory." *IEEE Trans. on Circuits and Systems*, vol. 35, no. 10, pp. 1257–1272, October.

Cimagalli, V., Bobbi, M., & Balsi, M. (1993). "MODA: Moving Object Detecting Architecture." *IEEE Trans. on Circuits and Systems-II*, vol. 40, no. 3, pp. 174–183, March.

Crounse, K.R., Roska, T., & Chua, L.O. (1993). "Image Halftoning with Cellular Neural Networks." *IEEE Trans. on Circuits and Systems-II*, vol. 40, no. 4, pp. 267–283, April.

Cruz, J.M. & Chua, L.O. (1991). "A CNN Chip for Connected Component Detection." *IEEE Trans. on Circuits and Systems*, vol. 38, no. 7, pp. 812–817, July.

Dash, L. & Chatterji, B.N. (1991). "Adaptive Contrast Enhancement and De-enhancement." *Pattern Recognition*, vol. 24, no. 4, pp. 289–302.

Davies, E.R. (1992). "A Skimming Technique For Fast Accurate Edge Detection." *Signal Processing*, vol. 26, no. 1, pp. 1–16, January.

Deriche, R. (1990). "Fast Algorithms for Low-Level Vision." *IEEE Trans. on Pattern Recognition and Machine Intelligence*, vol. 12, no. 1, pp. 78-87, January.

Deutsch, S. & Deutsch, A. (1993). *Understanding the Nervous System. An Engineering Perspective*. IEEE Press.

Dhawan, A.P., Buelloni, G., & Gordon, R. (1986). "Enhancement of Mammographic Features by Optimal Adaptive Neighbourhood Image Processing." *IEEE Trans. on Medical Imaging*, vol. 5, no. 1, pp. 8–15, March.

Ejiri, M. (1989). *Machine Vision. A Practical Technology for Advanced Image Processing*. Gordon and Breach Science Publishers, Japanese Technology Reviews, vol. 10.

Galias, Z. (1993). "Designing Cellular Neural Networks for the Evaluation of Local Boolean Functions." *IEEE Trans. on Circuits and Systems-II*, vol. 40, no. 3, pp. 219–222, March.

Gonzalez, R.C., & Woods, R.E., (1992). *Digital Image Processing*. Addison-Wesley.

Gordon, R. & Rangayyan, R.M. (1984). "Feature Enhancement of Film Mammograms Using Fixed and Adaptive Neighborhoods." *Applied Optics*, vol. 23, no. 4, pp. 560–564, February.

Goshtasby, A. (1994). "On Edge Focusing." *Image and Vision Computing*, vol. 12, no. 4, pp. 247–256.

Govindaraju, V., Srihari, S.N. & Sher, D.B. (1990). "A Computational Model for Face Location." *Third Int. Conference on Computer Vision*, pp. 718-721, Osaka, Japan, December 4-7.

Guillon, S., Baylou, P., & Najim, M. (1996). "Robust Nonlinear Contrast Enhancement Filters." *Proc. IEEE International Conference on Image Processing*, vol. I, pp. 757–760, Lausanne, Switzerland.

Haralick, R.M. & Watson, L. (1981). "A Facet Model for Image Data." *Computer Graphics and Image Processing*, vol. 15, pp. 113–129.

Harrer, H. (1993). "Multiple Layer Discrete-Time Cellular Neural Networks Using Time-Variant Templates." *IEEE Trans. on Circuits and Systems-II*, vol. 40, no. 3, pp. 191–199, March.

Harrer, H. & Nossek, J.A. (1992). "Discrete Time Cellular Neural Networks." *Int. J. Circuit Theory and Applications*, vol. 20, pp. 453-467.

Harris, J.L. (1977). "Constant Variance Enhancement: A Digital Processing Technique." *Applied Optics*, vol. 16, no. 5, pp. 1268-1271, May.

Hartline, H.K. & Ratliff, F. (1957). "Inhibitory Interactions of Receptor Units in the eye of Limulus." *J. General Physiology*, vol. 40, pp. 357–376.

Hartline, H.K. & Ratliff, F. (1958). "Spatial Summation of Inhibitory Influences in the Eye of Limulus, and the Mutual Interaction of Receptor Units." *J. General Physiology*, vol. 41, pp. 1049–1066.

Haykin, S. (1994). *Neural Networks, A Comprehensive Foundation*. Macmilllan.

Herskovitz, A. & Binford, T.O. (1970). "On Boundary Detection." Technical Report AI Memo no. 183, MIT.

Hertz, J., Krogh, A., & Palmer, R.G. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley.

Hildreth, E.C. (1983). "The Detection Of Intensity Changes by Computer and Biological Vision Systems." *Computer Vision, Graphics and Image Processing*, vol. 22, pp. 1–27.

Hopfield, J.J. & Tank, D.W. (1986). "Computing with Neural Circuits: A Model." *Science*, vol. 233, pp. 625–633.

Horn, B.K.P. (1977). "Understanding Image Intensities." *Artificial Intelligence*, vol. 8, no. 2, pp. 201–231, April.

Huertas, A. & Medioni, G. (1986). "Detection of Intensity Changes with Subpixel Accuracy using Laplacian-Gaussian Masks." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, no. 5, pp. 651–664.

Iannella, N. & Bouzerdoum, A. (1996). "Time Evolution of Receptive Fields." *Proceedings of the 1996 Australian New Zealand Conference on Intelligent Information Systems*, pp. 105-108, 18-20 November, Adelaide, South Australian, Australian.

Ifeachor, E.C., & Jervis, B.W., (1993). *Digital Signal Processing: A Practical Approach.* Addison-Wesley.

Inamori, S., Yamauchi, S. & Fukuhara, K. (1993). "A Method of Noise Reduction on Image Processing." *IEEE Trans. on Consumer Electronics*, vol. 39, no. 4, pp. 801-805, November.

Jain, A.K. (1989). *Fundamentals of Digital Image Processing*. Prentice-Hall.

Jernigan, M.E., & McLean, G.F. (1992). *Lateral Inhibition and Image Processing*, chapter 17, pp. 451–463. CRC Press, Boca Raton, USA.

Kadono, S. & Yamamitsu, C. (1993). "A Study on High Efficiency Coding of GDTV at 50 Mbps." *IEEE Trans. on Comsumer Electronics*, vol. 39, no. 1, pp. 49-56, February.

Kakarala, R. & Hero, A.O. (1992). "On Achievable Accuracy in Edge Localization." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 14, no. 7, pp. 777–781, July.

Kaufmann, P., Medioni, G. & Nevatia, R. (1984). "Visual Inspection Using Linear Features." *Pattern Recognition*, vol. 17, no. 5, pp. 485-491.

Kelley, R.B. (1992). *Bin-Picking Techniques*, in "Computer Vision: Theory and Applications.", Torras, C (ed.) pp. 337-375, Springer-Verlag.

Kitchen, L.J. & Malik, J.A. (1989). "The Effect of Spatial Discretization on the Magnitude and Direction Response of Simple Differential Edge Operators on a Step Edge." *Computer Vision, Graphics and Image Processing*, vol. 47, pp. 243–258.

Laine, A.F, Schuler, S., Fan, J., & Huda, W. (1994). "Mammographic Feature Enhancement by Multiscale Analysis." *IEEE Trans. on Medical Imaging*, vol. 13, no. 4, pp. 725-740, December.

Lee, C.L., Lee, D.H., Park, J.S., & Kim, Y.G. (1992). "A New Two-Layered Video Compression Scheme for Multiple Applications." *IEEE Trans. on Consumer Electronics*, vol. 38, no. 3, pp. 424-427, August.

Lindeberg, T. (1994). "Scale-Space Theory: A Basic Tool for Analysing Structures at Different Scales." *Journal of Applied Statistics*, vol. 21, no. 2, pp. 225-270. Supplement Advances in Applied Statistics: Statistics and Images: 2.

Liu, D. & Michel, A.N. (1993). "Cellular Neural Networks for Associative Memories." *IEEE Trans. on Circuits and Systems-II*, vol. 40, no. 2, pp. 119–121, February.

Lunscher, W.H.H.J. & Beddoes, M.P. (1986). "Optimal Edge Detector Design I: Parameter Selection and Noise Effects." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 164–186, March.

Lyvers, E.P. & Mitchell, R. (1988). "Precision Edge Contrast and Orientation Estimation."

IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 10, no. 6, pp. 927–937, November.

Mach, E. (1886a). "Uber den Physiologischen Effect Raumlich Verteiler Lichtrieze." *Sitzungsberichte der mathematisch-naturwissenschaftlichen Classe der kaiserlichen Akademie der Wissenschaften*, vol. 54 II, no. 134, pp. 131–144.

Mach, E. (1886b). "Uber die physiologische Wirkung raumlich verteilter Lichtreize III." *Sitzungsberichte der mathematisch-naturwissenschaftlichen Classe der kaiserlichen Akademie der Wissenschaften*, vol. 54 II, no. 134, pp. 393–408.

Marr, D. & Hildreth, E.C. (1980). "Theory of Edge Detection." *Proc. R. Soc. Lond. B*, vol. 207, pp. 187–217.

Matsumoto, T., Chua, L.O., & Suzuki, H. (1990a). "CNN Cloning Template: Hole-Filler." *IEEE Trans. on Circuits and Systems*, vol. 37, no. 5, pp. 635–638, May.

Matsumoto, T., Chua, L.O., & Suzuki, H. (1990b). "Image Thining with a Cellular Neural Network." *IEEE Trans. on Circuits and Systems*, vol. 37, no. 5, pp. 638–640, May.

Matsumoto, T., Chua, L.O., Suzuki, H. (1990c). "CNN Cloning Template: Connected Component Detector." *IEEE Trans. on Circuits and Systems*, vol. 37, no. 5, pp. 633–635, May.

Matsumoto, T., Chua, L.O., & Suzuki, H. (1990d). "CNN Cloning Template: Shadow Detector." *IEEE Trans. on Circuits and Systems*, vol. 37, no. 8, pp. 1070–1073, August.

McLean, G.F. & Jernigan, M.E. (1988). "Hierarchical Edge Detection." *Computer Vision, Graphics and Image Processing*, vol. 44, pp. 350–366.

Meer, P., Wang, S., & Wechsler, H. (1989). "Edge Detection by Associative Mapping." *Pattern Recognition*, vol. 22, no. 5, pp. 491–503.

Moganti, M., Ercal, F., Dagli, C.H., & Tsunekawa, S. (1996). "Automatic PCB Inspection Algorithms: A Survey." *Computer Vision and Image Understanding*, vol. 63, no. 2, pp. 287-313, March.

Nabet, B. & Pinter, R.B. (1991). *Sensory Neural Networks: Lateral Inhibition*. CRC Press.

Nalwa, V.S., & Binford, T.O. (1986). "On Detecting Edges." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 699–714, November.

Nevatia, R. & Babu, K.R. (1980). "Linear Feature Extraction and Description." *Computer Graphics and Image Processing*, vol. 13, pp. 257–269.

Novini, A.R. (1995). "Vision Technology in Food & Beverage Container Manufacturing." *Vision*, vol. 11, no. 4.

Nowak, A., Florek, A., & Piascik, T.A. (1992). "Computer Vision System Applied to Meat Classification." *IEE Int. Conference on Image Processing and it Applications*, no. 354, pp. 579-585, Maastricht, The Netherlands, April.

Ogata, K. (1987). *Discrete-time Control Systems*. Prentice-Hall, New Jersey.

Oppenheim, A.V., Schafer, R.W., & Stockham Jr, T.G. (1968). "Nonlinear Filtering of Multiplied and Convolved Signals." *Proc. of the IEEE*, vol. 56, no. 8, pp. 1264–1291, August.

Paik, J.K., Park, Y.C., & Kim, D.W. (1992a). "An Adaptive Motion Decision System for Digital Image Stabilizer Based on Edge Pattern Matching." *IEEE Trans. on Consumer Electronics*, vol. 38, no. 3, pp. 607-615, August.

Paik, J.K., Brailean, J.C., & Katsaggelos, A.K. (1992b). "An Edge Detection Algorithm Using Multi-State Adalines." *Pattern Recognition*, vol. 25, no. 12, pp. 1495–1504.

Pal, S.K. (1982). "A Note on the Quantitative Measure of Image Enhancement Through Fuzziness." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 4, no. 2, pp. 204–208, March.

Papoulis, A. (1991). *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill.

Paradis, M.A.K., & Jernigan, M.E. (1994). "Homomorphic vs Multiplicative Lateral Inhibition Models for Image Enhancement." pp. 286–291. IEEE Int. Conf. Syst., Man, and Cybernetics.

Paranjape, R.B., Morrow, W.M., & Rangayyan, R.M. (1992)."Adaptive-Neighbourhood Histogram Equalisation for Image Enhancement." *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 3, pp. 259–267, May.

Pau, L.F. (1990). *Computer Vision for Electronics Manufacturing*. Plenum Press, New York.

Peli, T., & Malah, D. (1982). "A Study on Edge Detection Algorithms." *Computer Graphics and Image Processing*, vol. 20, pp. 1–21.

Peli, E. (1990). "Contrast in Complex Images." *Journal of the Optical Society of America*, vol. 7, no. 10, pp. 2032-2040, October.

Perez-Munuzuri, V., Perez-Villar, V., & Chua, L.O. (1993). "Autowaves for Image Processing on a Two-Dimensional CNN Array of Excitable Nonlinear Circuits: Flat and Wrinkled Labyrinth." *IEEE Trans. on Circuits and Systems-I*, vol. 40, no. 3, pp. 174–181, March.

Perkins, W.A. (1983). "INSPECTOR: A Computer Vision System that Learns to Inspect Parts." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 5, no. 6, pp. 584-592, November.

Pinoli, J-C. (1997). A General Comparative Study of the Multiplicative Homomorphic, Log-Ratio and Logarithmic Image Processing Approaches“.” Signal Processing, vol. 58, pp. 11-45.

Pinter, R.B. (1983a). “Product Term Nonlinear Lateral Inhibition Enhances Visual Sensitivity for Small Objects or Edges.” *J. Theoretical Biology*, vol. 100, pp. 525–531.

Pinter, R.B. (1983b). “Thc Elcctrophysiological Bases for Linear and Nonlinear Product Term Lateral Inhibition and the Consequences for Wide Field Textured Stimuli.” *J. Theoretical Biology*, vol. 105, pp. 233–243.

Pinter, R.B. (1984). “Adaptation of Receptive Field Spatial Organisation via Multiplicative Lateral Inhibition.” *J. Theoretical Biology*, vol. 110, pp. 435–444.

Pinter, R.B. (1985). “Adaptation of Spatial Modulation Transfer Functions via Nonlinear Lateral Inhibition.” *Biological Cybernetics*, vol. 51, pp. 285–291.

Pizer, S.M., Amburn, E.P., Austin, J.D., Cromartie, R., Geselowitz, A., Greer, T., Romeny, B.T.H., Zimmerman, J.B., & Zuiderveld, K. (1987). “Adaptive Histogram Equalisation and its Variations.” *Computer Vision, Graphics and Image Processing*, vol. 39, pp. 355–368.

Poor, H.V. (1988). *An Introduction to Signal Detection And Estimation*. Springer-Verlag.

Price, K. (1986). “Anything you can do, I can do better (no you can’t).” *Computer Vision, Graphics and Image Processing*, vol. 36, pp. 387–391.

Proakis, J.G., & Manolakis, D.G. (1992). *Digital Signal Processing: Principles, Algorithms and Applications*, 2nd edition. Macmillan Publishing Company, New York.

Ratliff, F., Hartline, H.K., & Miller, W.H. (1963). “Spatial and Temporal Aspects of Retinal Inhibitory Interactions.” *J. Opt. Soc. Am.*, vol. 53, pp. 110–120.

Regazzoni, C.S., Tesei, A., & Vernazza, G. (1996). “*A Bayesian Network for Automatic Visual Crowding Estimation in Underground Stations*.” by Sanz, J.L.C. (ed.) in Image Technology, pp. 203-230.

Reklaitis, G.V., Ravindran, A., & Ragsdell, K.M. (1990). *Engineering Optimization: Methods and Applications*. John Wiley and Sons.

Robinson, G.S. (1977). “Edge Detection By Compass Gradient Methods.” *Computer Graphics and Image Processing*, vol. 6, pp. 492–501.

Rodieck, R.W. (1965). “Quantitative Analysis of Cat Retinal Ganglion Cell Response to Visual Stimuli,Laminar Origins and Termination of Cortical Connections of the Movement Detectors.” *Vision Research*, vol. 5, pp. 583–601.

Roska, T., Boros, T., Radvanyi, A., Thiran, P., & Chua, L.O. (1992a). "Detecting Moving and Standing Objects Using Cellular Neural Networks." *Int. J. Circuit Theory and Applications*, vol. 20, pp. 613–628.

Roska, T. & Chua, L.O. (1992b). "Cellular Neural Networks with Non-linear and Delay-Type Elements and Non-Uniform Grids." *Int. J. Circuit Theory and Applications*, vol. 20, pp. 469–481.

Sandefur, J.T. (1990). *Discrete Dynamical Systems: Theory and Applications*. Clarendon Press.

Schneiderman, H. & Nashman, M. (1992). "Visual Processing for Autonomous Driving." *IEEE Workshop on Applications of Computer Vision*, pp. 164-171, Palm Springs, USA, Nov. 30 - Dec. 2.

Shi, B.E., Roska, T., & Chua, L.O. (1993). "Design of Linear Cellular Neural Networks for Motion Sequence Filtering." *IEEE Trans. on Circuits and Systems-II*, vol. 40, no. 5, pp. 320–321, May.

Slot, K. (1992). "Cellular Neural Network Design for Solving Specific Image-Processing Problems." *Int. j. Circ. Th. Appl.*, vol. 20, pp. 629–637.

Sonka, M., Hlavac, V., & Boyle, R. (1993). *Image Processing, Analysis and Machine Vision*. Chapman & Hall.

Sotak Jr, G.E. & Boyer, K.L. (1989). "The Laplacian-of-Gaussian Kernel: A Formal Analysis and Design Procedure for Fast Accurate Convolution and Full-Frame Output." *Computer Vision, Graphics and Image Processing*, vol. 48, pp. 147–189.

Spillman, L. & Werner, J.S. editors. (1990). *Visual Perception: the Neurophysiological Foundations*. Academic Press.

Srinivasan, M.V., Laughlin, S.B., & Dubs, A. (1982). "Predictive Coding: a Fresh View of Inhibition in the Retina." *Proc. R. Soc. Lond.*, vol. B216, pp. 427–459.

Suresh, B.R., Fundakowski, R.A., Levitt, T.S., & Overland, J.E. (1983). "A Real-Time Automated Visual Inpection System for Hot Steel Slabs." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 5, no. 6, pp. 563-572.

Suzuki, H., Matsumoto, T., & Chua, L.O. (1992). "A CNN Handwritten Character Recognizer." *Int. J. Circ. Th. Appl.*, vol. 20, pp. 601–612.

Sziranya, T. & Csicsvari, J. (1993). "High-Speed Character Recognition Using a Dual Cellular Neural Network Architecture (CNND)." *IEEE Trans. on Circuits and Systems-II*, vol. 40, no. 3, pp. 223–231, March.

Tagare, H.D. & deFigueridо, R.J.P. (1990). "On the Localization Performance Measure and Optimal Edge Detection." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 12, pp. 1186–1190, December.

Tagare, H.D. & deFigueiredo, R.J.P. (1994). "Reply to "On the localization Performance Measure and Optimal Edge Detection"." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 108–110, January.

Torre, V. & Poggio, T. (1986). "On Edge Detection." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, no. 5, pp. 651-664.

Uchiyama, M., Itoh, F., Niwa, H., Kawagoe, Y., Takizawa, S., Tsuge, S., Ohki, M. & Hasiguchi, S. (1992). "A Digital Still Camera." *IEEE Trans. on Comsumer Electronics*, vol. 38, no. 3, pp. 698-701, August.

van Vliet, L.J., Young, I.T., & Beckers, G.L. (1989). "A Nonlinear Laplace Operators as Edge Detector in Noisy Images." *Computer Vision, Graphics and Image Processing*, vol. 45, pp. 165–195.

West, G.A.W. (1984). "A System for the Automatic Inspection of Bare-Printed Circuit Boards." *IEEE Trans. on System, Man and Cybernetics*, vol. 14, no. 5, pp. 767-773, September/November.

White, R.G. & Schowengerdt, R.A. (1992). "Effect of Point-Spread Functions on the Edge Orientation Precision of First-Derivative Operators." *Optical Engineering*, vol. 31, no. 10, pp. 2239–2245, October.

Widrow, B., Winter, R.G., & Baxter, R.A. (1988). "Layered Neural Nets for Pattern Recognition." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1109–1118, July.

Zimmerman, J.B., Pizer, S.M., Staab, E.V., Perry, J.R., McCartney, W., & Brenton, B.C. (1988). "An Evaluation of the Effectiveness of Adaptive Histogram Equalization for Contrast Enhancement." *IEEE Trans. on Medical Imaging*, vol. 7, no. 4, pp. 304–312, December.

Appendix A Derivation of the SICNN pdf, HR, PD and FA.

A.1 Theoretical SICNN Output pdf

A.1.1 Gaussian Noise Case

First, consider a constant input to the feedforward SICNN with additive Gaussian noise, denoted by the *random variable* (RV) $\boldsymbol{I}$. Also let the RV z be $z = a + \sum_j w_j \boldsymbol{I}_j$ such that the SICNN output is given by the RV

$$y = \frac{\boldsymbol{I}}{z}.$$

From pg. 138 of Papoulis (1991), the *probability density function* (pdf) of y is

$$f_y(y) = \int_{-\infty}^{\infty} |z| f_y(I, z)\, dz = \int_{-\infty}^{\infty} |z| f_y(zy, z)\, dz. \qquad \text{EQ (A.1)}$$

Let $\boldsymbol{I} \sim N(\mu_I, \sigma_I)$ and $z \sim N(\mu_z, \sigma_z)$, that is

$$f_y(I) = \frac{1}{\sqrt{2\pi}\sigma_I} \exp\left(\frac{-(I-\mu_I)^2}{2\sigma_I^2}\right)$$

$$f_y(z) = \frac{1}{\sqrt{2\pi}\sigma_z} \exp\left(-\frac{-(z-\mu_z)^2}{2\sigma_z^2}\right).$$

If both $\boldsymbol{I}$ and z are independent, then

$$f_y(I, z) = f_y(I) f_y(z) = \frac{1}{2\pi\sigma_I\sigma_z} \exp\left(-\frac{(I-\mu_I)^2}{2\sigma_I^2} - \frac{(z-\mu_z)^2}{2\sigma_z^2}\right)$$

$$\Rightarrow f_y(zy, z) = \frac{1}{2\pi\sigma_I\sigma_z}\exp\left(-\frac{(zy-\mu_I)^2}{2\sigma_I^2}-\frac{(z-\mu_z)^2}{2\sigma_z^2}\right)$$

$$= \frac{1}{2\pi\sigma_I\sigma_z}\exp\left(-\frac{(z^2y^2-2zy\mu_I+\mu_i^2)\,\sigma_z^2+(z^2-2z\mu_2+\mu_2^2)\,\sigma_I^2}{2\sigma_I^2\sigma_z^2}\right)$$

$$= \frac{1}{2\pi\sigma_I\sigma_z}\exp\left(-\frac{z^2(y^2\sigma_z^2+\sigma_I^2)-2z(y\mu_1\sigma_z^2+\mu_2\sigma_I^2)+(\mu_I^2\sigma_z^2+\mu_z^2\sigma_I^2)}{2\sigma_I^2\sigma_z^2}\right)$$

Letting $\alpha = \frac{y^2\sigma_z^2+\sigma_I^2}{2\sigma_I^2\sigma_z^2}$, $\gamma = \frac{y\mu_I\sigma_z^2+\mu_z\sigma_I^2}{2\sigma_I^2\sigma_z^2}$ and $\rho = \frac{1}{2\pi\sigma_I\sigma_z}\exp\left(-\frac{\mu_I^2\sigma_z^2+\mu_z^2\sigma_I^2}{2\sigma_I^2\sigma_z^2}\right)$, then

$$f_y(zy, z) = \rho\exp(-\alpha(z^2-2z\gamma))$$

$$= \rho\exp(-\alpha(z^2-2z\gamma+\gamma^2-\gamma^2))$$

$$= A\exp(-\alpha(z-\gamma)^2)$$

where $A = \rho\exp(-\gamma^2)$. Now, using this expression,

$$f_y(y) = \int_{-\infty}^{\infty}|z|f_y(zy, z)\,dz = A\int_{-\infty}^{\infty}|z|\exp(-\alpha(z-\gamma)^2)\,dz$$

$$= A\int_{-\infty}^{0}(-z)\exp(-\alpha(z-\gamma)^2)\,dz + A\int_{0}^{\infty}z\exp(-\alpha(z-\gamma)^2)\,dz$$

Changing variables with $s = z-\gamma \quad \Rightarrow dz = ds$, we obtain

$$f_y(y) = -A\int_{-\infty}^{-\gamma}(s+\gamma)\exp(-\alpha s^2)\,ds + A\int_{-\gamma}^{\infty}(s+\gamma)\exp(-\alpha s^2)\,ds$$

$$= -A\gamma\int_{-\infty}^{-\gamma}\exp(-\alpha s^2)\,ds - A\int_{-\infty}^{-\gamma}s\exp(-\alpha s^2)\,ds$$

$$+ A\gamma\int_{-\gamma}^{\infty}\exp(-\alpha s^2)\,ds + A\int_{-\gamma}^{\infty}s\exp(-\alpha s^2)\,ds$$

Evaluating each term of this expression:

$$-A\int_{-\infty}^{-\gamma}s\exp(-\alpha s^2)\,ds = \frac{-A\exp(-\alpha s^2)}{2\alpha}\Bigg|_{-\infty}^{-\gamma} = \frac{A\exp(-\alpha\gamma^2)}{2\alpha},$$

$$A\int_{-\gamma}^{\infty}s\exp(-\alpha s^2)\,ds = \frac{A\exp(-\alpha s^2)}{2\alpha}\Bigg|_{-\gamma}^{\infty} = \frac{A\exp(-\alpha\gamma^2)}{2\alpha}, \text{ and}$$

$$-A\gamma\int_{-\infty}^{-\gamma}\exp(-\alpha s^2)\,ds + A\gamma\int_{-\gamma}^{\infty}\exp(-\alpha s^2)\,ds = A\gamma\int_{-\gamma}^{\gamma}\exp(-\alpha s^2)\,ds. \qquad \text{EQ (A.2)}$$

If we change variables $\xi = \sqrt{\alpha}s \Rightarrow ds = d\xi/\sqrt{\alpha}$, then EQ (A.2) becomes

$$A\gamma\int_{-\gamma}^{\gamma}\exp(-\alpha s^2)\,ds = \frac{A\gamma}{\sqrt{\alpha}}\int_{-\gamma\sqrt{\alpha}}^{\gamma\sqrt{\alpha}}\exp(-\xi^2)\,d\xi = \frac{2A\gamma}{\sqrt{\alpha}}\int_{0}^{\gamma\sqrt{\alpha}}\exp(-\xi^2)\,d\xi = A\gamma\sqrt{\frac{\pi}{\alpha}}erf(\gamma\sqrt{\alpha}).$$

where $erf(x) = \int_0^x \exp(-x^2)\,dx$. Combining these we obtain the pdf of the SICNN output as

$$f_y(y) = A\gamma\sqrt{\frac{\pi}{\alpha}}erf(\gamma\sqrt{\alpha}) + \frac{A\exp(-\alpha\gamma^2)}{\alpha}. \qquad \text{EQ (A.3)}$$

A.1.2 Uniform Noise

Suppose that x is a uniformly distributed RV with mean μ and variance σ, i.e. $x \sim U(\mu, \sigma)$. The pdf of x is

$$\begin{aligned} f_x(x) &= \frac{1}{2\sqrt{3}\sigma}, \quad -\sqrt{3}\sigma + \mu \le x \le \sqrt{3}\sigma + \mu \\ &= \frac{1}{2\sqrt{3}\sigma}u(x - \mu + \sqrt{3}\sigma)\,[1 - u(x + -\mu - \sqrt{3}\sigma)] \end{aligned} \qquad \text{EQ (A.4)}$$

where $u(x)$ is the unit step function.

As we did for the Gaussian noise case, we assume a constant input to the SICNN with additive uniformly distributed noise. Let I be the RV for this input signal, and z be the RV where $z = a + \sum_j w_j I_j$, such that the RV of the SICNN output is $y = I/z$. We recall from Papoulis (1991) that the pdf of y is given by

$$f_y(y) = \int_{-\infty}^{\infty}|z|f_y(I, z)\,dz = \int_{-\infty}^{\infty}|z|f_y(zy, z)\,dz. \qquad \text{EQ (A.5)}$$

If the RVs I and z have distributions $I \sim U(\mu_I, \sigma_I)$ and $z \sim U(\mu_z, \sigma_z)$, then using EQ (A.4) we can write their pdfs as

$$f_y(I) = \frac{1}{2\sqrt{3}\sigma_I}u\left(I - \mu_I + \sqrt{3}\sigma_I\right)\left[1 - uI + -\mu_I - \sqrt{3}\sigma_I\right], \qquad \text{EQ (A.6)}$$

$$f_z(z) = \frac{1}{2\sqrt{3}\sigma_z} u\left(z-\mu_z+\sqrt{3}\sigma_z\right)\left[1-u\left(z+-\mu_z-\sqrt{3}\sigma_z\right)\right]. \qquad \text{EQ (A.7)}$$

If we assume that I and z are independent, then

$$\begin{aligned} f_y(I,z) &= f_y(I) f_z(z) \\ &= \frac{1}{2\sqrt{3}\sigma_I} u\left(I-\mu_I+\sqrt{3}\sigma_I\right)\left[1-u\left(I-\mu_I-\sqrt{3}\sigma_I\right)\right] \\ &\quad \times \frac{1}{2\sqrt{3}\sigma_z} u\left(z-\mu_z+\sqrt{3}\sigma_z\right)\left[1-\left(uz-\mu_z-\sqrt{3}\sigma_z\right)\right], \end{aligned} \qquad \text{EQ (A.8)}$$

$$\begin{aligned} f_y(zy,z) &= \frac{1}{12\sigma_I\sigma_z} u\left(zy-\mu_I+\sqrt{3}\sigma_I\right)\left[1-u\left(zy-\mu_I-\sqrt{3}\sigma_I\right)\right] \\ &\quad \times u\left(z-\mu_z+\sqrt{3}\sigma_z\right)\left[1-u\left(z-\mu_z-\sqrt{3}\sigma_z\right)\right] \end{aligned} \qquad \text{EQ (A.9)}$$

The pdfs $f_y(I)$, $f_y(z)$ and $f_y(I,z)$ are shown in Figure A.1.

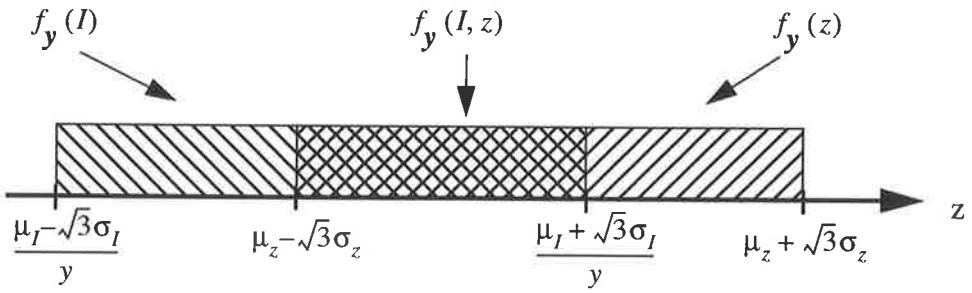


Figure A.1 The pdfs $f_y(I)$, $f_y(z)$ and $f_y(I,z)$ which are given by EQ (A.6), EQ (A.7) and EQ (A.8) respectively. The cross-hatched area is where $f_y(I)$ and $f_y(z)$ overlap, corresponding to $f_y(I,z)$. We assume that $y>0$.

To obtain the pdf $f_y(y)$, the integral of $|z|f_y(zy,z)$ given in EQ (A.1) must be performed in 3 parts: for $y>0$, $y<0$ and $y=0$. We evaluate the integral for each of these y in turn.

For y>0

We first assume that, given any value of $y>0$, the two pdfs $f_y(I)$ and $f_y(z)$ must overlap, i.e., from Figure A.1 we have

$$\left[\left(\frac{\mu_I+\sqrt{3}\sigma_I}{y}\right)>\left(\mu_z-\sqrt{3}\sigma_z\right)\right] \text{ and } \left[\left(\frac{\mu_I-\sqrt{3}\sigma_I}{y}\right)<\left(\mu_z+\sqrt{3}\sigma_z\right)\right].$$

If this condition does not hold, then $f_y(y)$ is zero for that particular value of y. If it does hold, then we can evaluate the integral given by EQ (A.5). In fact, the integral only exists in the overlapping region shown in Figure A.1, given by

$$max\left\{\frac{\mu_I-\sqrt{3}\sigma_I}{y}, \mu_z-\sqrt{3}\sigma_z\right\} \le z \le min\left\{\frac{\mu_I+\sqrt{3}\sigma_I}{y}, \mu_z+\sqrt{3}\sigma_z\right\}.$$

Let $MAX = max\left\{\frac{\mu_I-\sqrt{3}\sigma_I}{y}, \mu_z-\sqrt{3}\sigma_z\right\}$, and $MIN = min\left\{\frac{\mu_I+\sqrt{3}\sigma_I}{y}, \mu_z+\sqrt{3}\sigma_z\right\}$.

If $MAX < 0$, then

$$\begin{aligned} f_y(y) &= \int_{MAX}^{0} (-z)\frac{1}{12\sigma_I\sigma_z}dz + \int_{0}^{MIN} z\frac{1}{12\sigma_I\sigma_z}dz \\ &= \frac{1}{12\sigma_I\sigma_z}\cdot\frac{1}{2}\cdot MAX^2 + \frac{1}{12\sigma_I\sigma_z}\cdot\frac{1}{2}\cdot MIN^2 \\ &= \frac{1}{24\sigma_I\sigma_z}(MAX^2+MIN^2) \end{aligned}$$

If $MAX \ge 0$,

$$f_y(y) = \int_{MAX}^{MIN} z\frac{1}{12\sigma_I\sigma_z}dz = \frac{1}{24\sigma_I\sigma_z}(MIN^2-MAX^2)$$

For y< 0

For $y < 0$, $f_y(y)$ is defined in the domain

$$-\left(\frac{\mu_I+\sqrt{3}\sigma_I}{|y|}\right) \le z \le -\left(\frac{\mu_I-\sqrt{3}\sigma_I}{|y|}\right)$$

We again assume that the pdfs $f_y(I)$ and $f_y(z)$ overlap, i.e.,

$$\left[-\left(\frac{\mu_I-\sqrt{3}\sigma_I}{|y|}\right) > \left(\mu_z-\sqrt{3}\sigma_z\right)\right] \text{ and } \left[-\left(\frac{\mu_I+\sqrt{3}\sigma_I}{|y|}\right) < \left(\mu_z+\sqrt{3}\sigma_z\right)\right].$$

If the pdfs do not overlap, then $f_y(y) = 0$ for this value of y. If this does hold, then the integral exists and can be evaluated. Let $MAX = max\left\{\frac{-\left(\mu_I+\sqrt{3}\sigma_I\right)}{|y|}, \mu_z-\sqrt{3}\sigma_z\right\}$ and $MIN = min\left\{\frac{-\left(\mu_I-\sqrt{3}\sigma_I\right)}{|y|}, \mu_z+\sqrt{3}\sigma_z\right\}$.

If $MAX < 0$,

$$f_y(y) = \int_{MAX}^{0} (-z)\frac{1}{12\sigma_I\sigma_z}dz + \int_{0}^{MIN} z\frac{1}{12\sigma_I\sigma_z}dz$$
$$= \frac{1}{12\sigma_I\sigma_z}\cdot\frac{MAX^2}{2} + \frac{1}{12\sigma_I\sigma_z}\cdot\frac{MIN^2}{2}$$
$$= \frac{1}{24\sigma_I\sigma_z}(MAX^2+MIN^2)$$

If $MAX \geq 0$,

$$f_y(y) = \int_{MAX}^{MIN} z\frac{1}{12\sigma_I\sigma_z}dz = \frac{1}{24\sigma_I\sigma_z}(MIN^2-MAX^2).$$

If y = 0,

From EQ (A.9), the joint pdf of I and z becomes,

$$f_y(0,z) = \frac{1}{12\sigma_I\sigma_z}u\left(0-\mu_I+\sqrt{3}\sigma_I\right)\left[1-u\left(0-\mu_I-\sqrt{3}\sigma_I\right)\right]$$
$$\times u\left(z-\mu_z+\sqrt{3}\sigma_z\right)\left[1-u\left(z+-\mu_z-\sqrt{3}\sigma_z\right)\right]$$

Thus, $f_y(0,z) = 0$ if either $\mu_I-\sqrt{3}\sigma_I > 0$ or $\mu_I+\sqrt{3}\sigma_I < 0$. The second condition can never be true (assuming $\mu_I \geq 0$), so the necessary condition for $f_y(y) = 0$ is $\mu_I > \sqrt{3}\sigma_I$.

If $\mu_I \leq \sqrt{3}\sigma_I$, then $f_y(0,z)$ is non-zero over the range $\mu_z-\sqrt{3}\sigma_z \leq z \leq \mu_z+\sqrt{3}\sigma_z$. If $\mu_z-\sqrt{3}\sigma_z < 0$, then

$$f_y(0) = \int_{\mu_z-\sqrt{3}\sigma_z}^{0} (-z)\frac{1}{12\sigma_I\sigma_z}dz + \int_{0}^{\mu_z+\sqrt{3}\sigma_z} z\frac{1}{12\sigma_I\sigma_z}dz$$
$$= \frac{1}{24\sigma_I\sigma_z}\left[\left(\mu_z-\sqrt{3}\sigma_z\right)^2+\left(\mu_z+\sqrt{3}\sigma_z\right)^2\right]$$
$$= \frac{1}{12\sigma_I\sigma_z}[\mu_z^2+3\sigma_z^2]$$

If $\mu_z-\sqrt{3}\sigma_z \geq 0$,

$$f_y(0) = \int_{\mu_z-\sqrt{3}\sigma_z}^{\mu_z+\sqrt{3}\sigma_z} z\frac{1}{12\sigma_I\sigma_z}dz = \frac{1}{24\sigma_I\sigma_z}\left[\left(\mu_z+\sqrt{3}\sigma_z\right)^2+\left(\mu_z-\sqrt{3}\sigma_z\right)^2\right] = \frac{\sqrt{3}\mu_z}{6\sigma_I}$$

A.1.3 Multiplicative Noise

The pdf of the SICNN output when the input has multiplicative noise is derived in exactly the same way as for the Gaussian noise case, using EQ (A.3). In this case, however, the input noise variance is made dependent upon the underlying signal intensity of that pixel, i.e.

$$\sigma = QI$$

where $Q = \sqrt{\frac{c^2 10^{-ENR/10}}{(1+c^2)}}$, and c is the contrast of the input step edge to the SICNN.

A.1.4 Experimental Comparison

We now compare the theoretical and experimental pdfs of the edge pixel of the SICNN output for an input step edge. Figure A.2 shows the comparison for Gaussian noise as the input ENR is varied. We can see that for all the values of the ENR, the theoretical curves compare extremely well to those obtained experimentally. Even for very small input ENR signals, the difference between the two curves is hardly observable. As expected, for low ENR, the pdf is very broad (as the noise variance is large), but as the ENR increases the pdf becomes narrower and larger in height, which maintains the unit area of the pdf.

Figure A.3 shows the comparison between the experimental and theoretical curves for input step edges has additive uniform noise. Once again, the difference between the two curves are negligible, indicating the accuracy of the derived equations. The pdfs have a square-like shape, which is expected since the input pdf is square. Like for the Gaussian noise case, the pdf is broad for low ENR values (indicating large variance due to noise), while for large ENR the pdf becomes narrower and larger in height.

Figure A.4 shows the comparison when the input has multiplicative Gaussian noise. In this case, we do in fact see some differences between the theoretical and experimental curves, although they are still very close to each other. As for the other noise distributions, the pdf is very broad for low ENR, but becomes narrower and larger for larger ENR.

The overall accuracy of the theoretical pdfs compared to the experimental ones confirms the accuracy of the derived analytical expressions for the SICNN output pdf, under a wide range of noisy input signals.

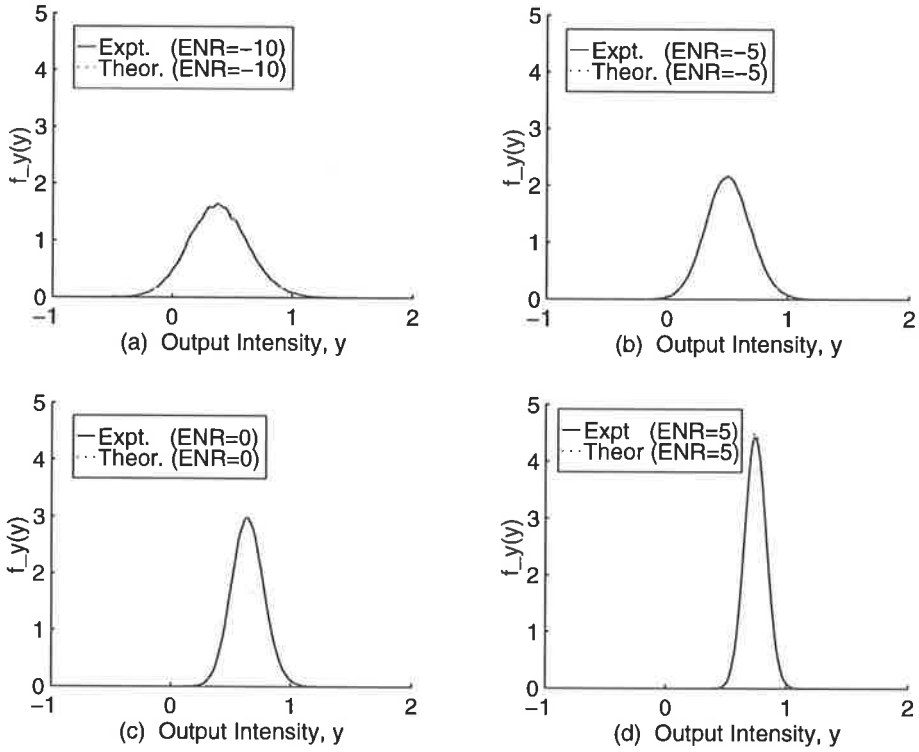


Figure A.2 The output pdf of the SICNN edge pixel when the input step edge has **Gaussian noise** with different ENR. The SICNN has asymmetrical rectangular weights, with $W = 1$, $r = 5$, optimal decay factor, $I_o = 10$ and $c = 0.25$.

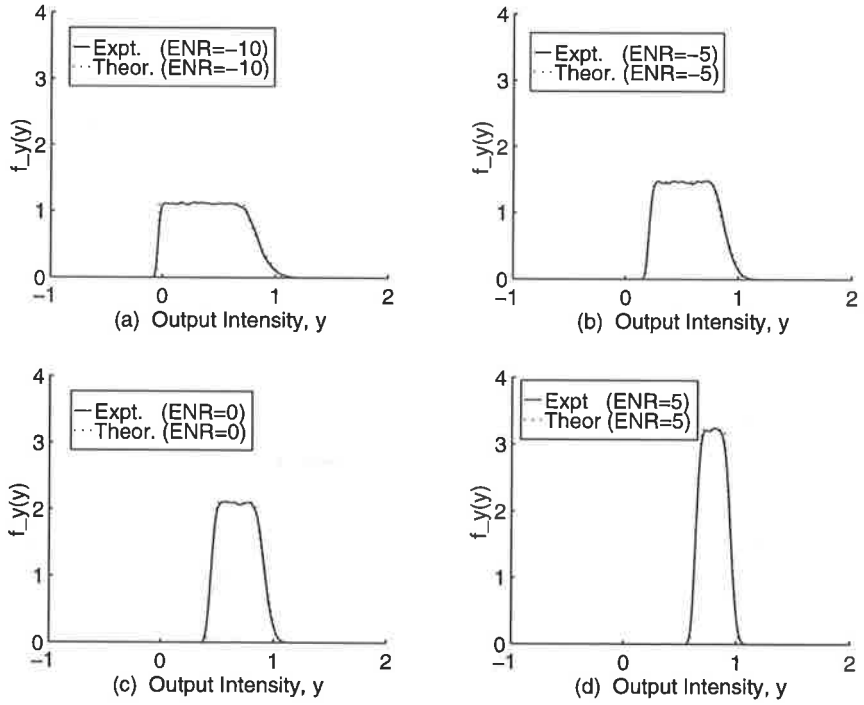


Figure A.3 As per Figure A.2 but for **uniform noise**.

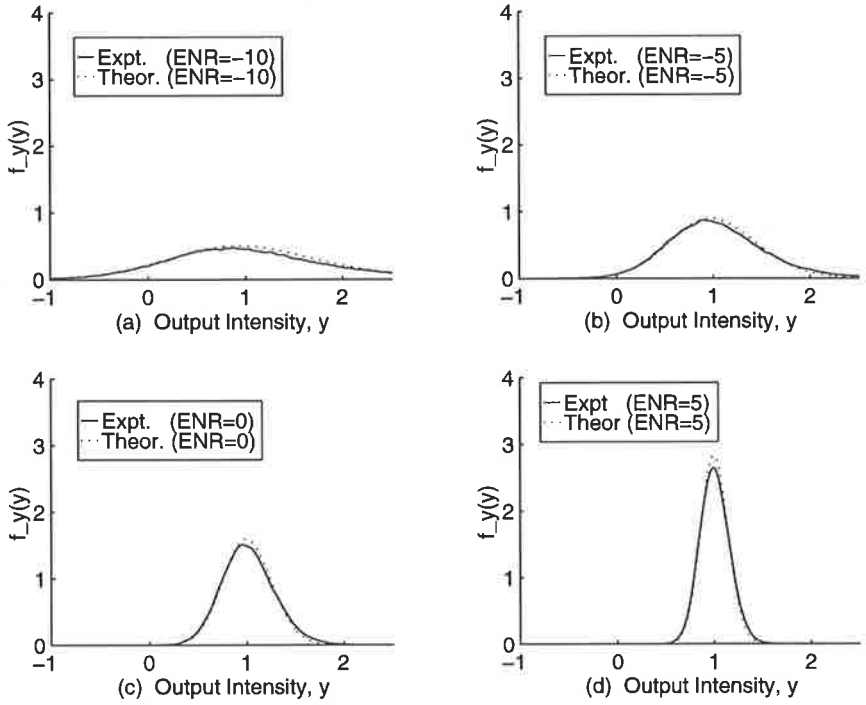


Figure A.4 As per Figure A.2 but for **multiplicative** noise.

A.2 Theoretical Hit Rate (HR)

The HR is defined as the probability that the intensity of the edge pixel is greater than the intensity of any other pixel (or greater than the maximum intensity of any non-edge pixel). From Papoulis (1991, pg. 141), the pdf of the maximum of N independent variables, $y = max(y_1, y_2, \ldots, y_N)$ is given by

$$\begin{aligned} f_{\mathbf{y}}(y) &= \left[f_{\mathbf{y}_1}(y) F_{\mathbf{y}_2}(y) \ldots F_{\mathbf{y}_N}(y)\right] + \left[f_{\mathbf{y}_2}(y) F_{\mathbf{y}_1}(y) \ldots F_{\mathbf{y}_N}(y)\right] + \ldots \\ &\quad + \left[f_{\mathbf{y}_N}(y) F_{\mathbf{y}_1}(y) \ldots F_{\mathbf{y}_{N-1}}(y)\right] \\ &= \sum_{i=1}^{N} \left[\prod_{\substack{j=1 \\ j \neq i}}^{N} F_{\mathbf{y}_j}(y)\right] f_{\mathbf{y}_i}(y) \end{aligned}$$

where $F_{\mathbf{y}_j}(y)$ is the cumulative distribution of $f_{\mathbf{y}_i}(y)$. We saw in the previous subsections how to compute $f_{\mathbf{y}_i}(y)$ for various types of input noise, while $F_{\mathbf{y}_i}(y)$ is the integral of $f_{\mathbf{y}_i}(y)$.

Consider a step edge input to the SICNN of mean intensity I_0, contrast c and overall length $M = 2N$, i.e., the first N pixels have a mean intensity of $I_0(1-c)$ - we nominally refer to these collectively as the "lower" part of the edge, while the next N pixels have a mean intensity of $I_0(1+c)$, which we refer to as the "upper" part of the edge.

It should be noted that only $2r$ SICNN output pixels are affected by the input step edge discontinuity: r pixels to the left of the discontinuity and r pixels on the right (or $(r-1)$ pixels if we exclude the edge pixel itself). Using the above equation we can find the pdf of the maximum value of the output intensity of the upper and lower parts of the SICNN output, denoted by f_{y_L} and f_{y_U}, where

$$y_L = max\{\frac{I_1}{z_1}, \frac{I_2}{z_2}, \ldots, \frac{I_N}{z_N}\}$$

$$y_U = max\{\frac{I_{N+2}}{z_{N+2}}, \frac{I_{N+3}}{z_{N+3}}, \ldots, \frac{I_{2N}}{z_{2N}}\}$$

y_U does not include I_{N+1}/z_{N+1} as this is the output edge pixel itself, and its distribution will eventually be compared to the distribution of the non-edge pixel.

The RV of the overall maximum output of all non-edge pixels is $y = max\{y_L, y_U\}$. Assuming that y_L and y_U are independent, then y has pdf

$$f_y(y) = f_{y_L}(y)F_{y_U}(y) + f_{y_U}(y)\ F_{y_L}(y).$$

where $F_{y_L}(y)$ and $F_{y_U}(y)$ are the cumulative distribution functions of $f_{y_L}(y)$ and $f_{y_U}(y)$, respectively. The HR is then the probability that the intensity of the edge pixel's RV y_e is greater than $y = max\{y_L, y_U\}$. So,

$$\begin{aligned} HR &= P(y_e > y) \\ &= P(y - y_e < 0) \end{aligned}$$

To find this probability, we first need to determine the pdf of the RV $s = y - y_e$. Thus, $y = s + y_e$ and $ds = dy$. From Papoulis(1991, pg. 136), the pdf of s is given by

$$f_s(s) = \int_{-\infty}^{\infty} f(y, y_e)\, dy_e = \int_{-\infty}^{\infty} f(s + y_e, y_e)\, dy_e.$$

If the RVs y and y_e are independent, then $f(y, y_e) = f_y(y)f_{y_e}(y_e)$ and

$$f_{\mathbf{s}}(s) = \int_{-\infty}^{\infty} f_{\mathbf{y}}(s+y_e) f_{\mathbf{y}_e}(y_e)\, dy_e = R_{f_y, f_{y_e}}(s)$$

where $R_{f_y, f_{y_e}}$ is the correlation function of the pdfs of the RVs y and y_e. Thus, the HR is

$$HR = P(y - y_e < 0) = \int_{-\infty}^{0} f_{\mathbf{s}}(s)\, ds = \int_{-\infty}^{0}\int_{-\infty}^{\infty} f_{\mathbf{y}}(s+y_e) f_{\mathbf{y}_e}(y_e)\, dy_e ds = \int_{-\infty}^{0} R_{f_y, f_{y_e}}(s)\, ds,$$

i.e., the HR for the SICNN can be computed as the area from $-\infty$ to 0 of the cross-correlation of the pdfs of the RVs y and y_e.

A.2.1 Experimental Comparison

We compare the theoretical and experimental HR for the SICNN for a step edge input as the neighbourhood size of the SICNN is varied. Figure A.5 shows the comparison for additive Gaussian noise; the agreement between the theoretical and experimental HR curves is very good, even for large neighbourhood sizes where we would expect the dependence between neighbouring output pixels to be stronger, and hence the assumptions to break down.

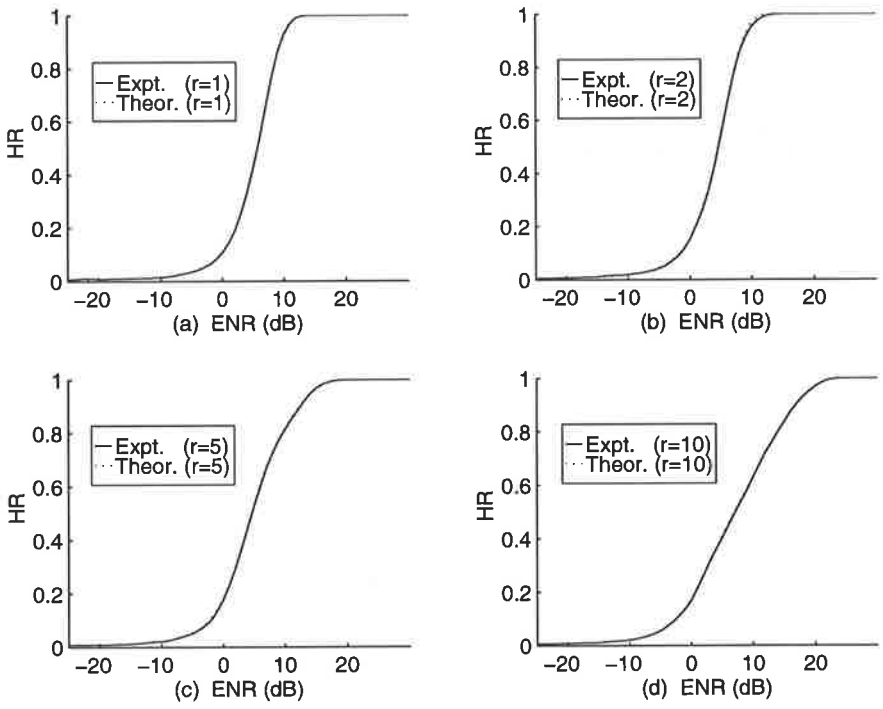


Figure A.5 Comparison of the SICNN theoretical and experimental HR. The SICNN has asymmetrical rectangular weights, with $W = 1$, $r = 5$, optimal decay factor, $I_O = 10$, $c = 0.25$ and additive **Gaussian** noise.

Figure A.6 shows the same comparison, but for the step edge with uniform noise. In this case, there are minor differences between the experimental curves and theoretical ones, particularly for $r = 2$ and $r = 5$, but the difference is still not great. Small differences between the experimental and theoretical HR curves are also observed when the input has multiplicative noise, as shown in Figure A.7, although these differences are not large. Thus, the overall agreement of the theoretical plots to the experimental ones, as is evident in all three figures, demonstrates the accuracy of the derived analytical expression for the HR.

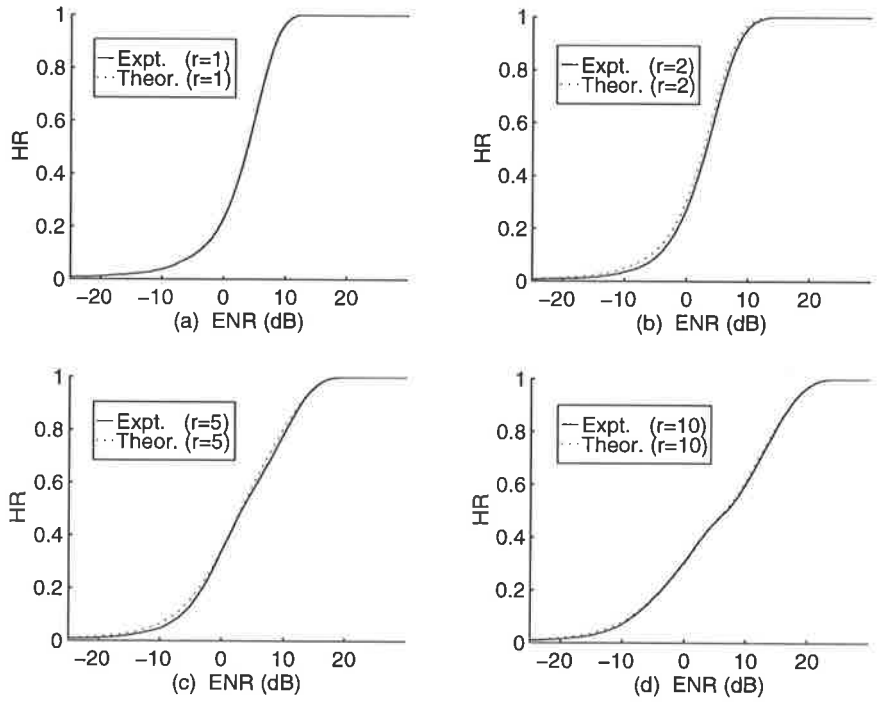


Figure A.6 As per Figure A.5 but for **uniform noise**.

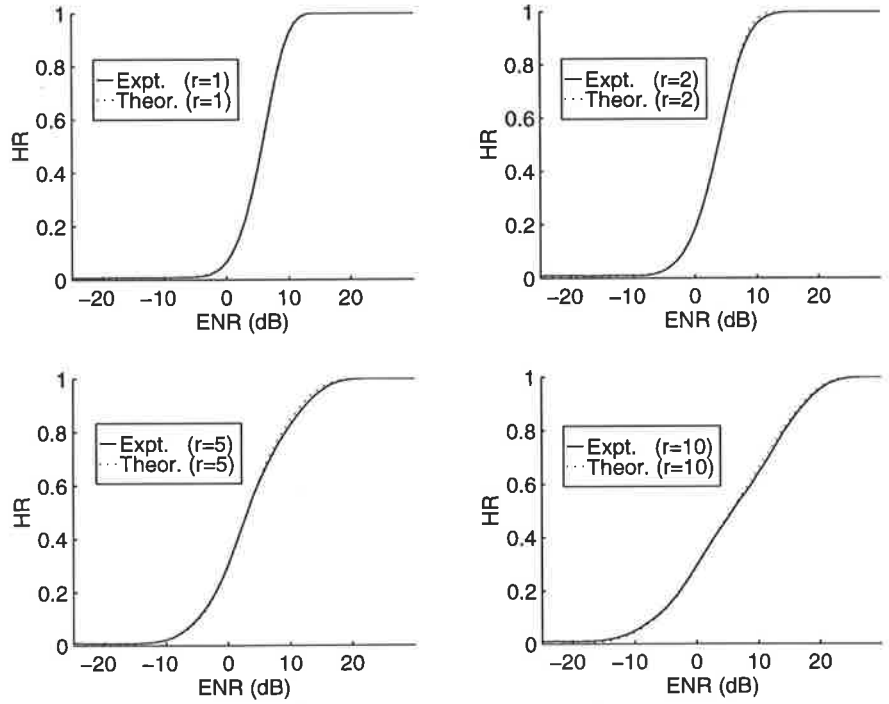


Figure A.7 As per Figure A.5 but for **multiplicative noise.**

A.3 Theoretical PD and FA

Let H_1 be the hypothesis that the SICNN output pixel is an edge pixel, and H_0 be the hypothesis that a SICNN output pixel is not an edge pixel. We can then define (Poor, 1988, pg. 31-33):

False alarm (FA) rate is the probability that we accept H_1 given that H_0 is true, and the ***Probability of Detection*** is the probability that we accept H_1 given that H_1 is true.

Thus, given that we threshold the output of the SICNN, the FA is the probability that the pixel whose intensity exceeds the threshold is actually a non-edge pixel. The PD is the probability that the intensity of the output edge pixel will exceed the threshold.

The probability that any given output pixel intensity y_i exceeds the threshold τ is

$$P(y_i > \tau) = \int_{\tau}^{\infty} f_{\mathbf{y}}(y)\, dy$$

where the pdf $f_{\mathbf{y}}(y)$ of the SICNN output was previously derived for an input with Gaussian, uniform and multiplicative noise. The PD is then given by

$$P(y_e > \tau) = \int_{\tau}^{\infty} f_{y_e}(y)\, dy.$$

Let y_{ne} denote the non-edge pixels, then the FA is

$$P(y_{ne} > \tau) = P(y_{ne} > \tau | y_1) P(y_1) + \ldots + P(y_{ne} > \tau | y_i) P(y_i) + \ldots + P(y_{ne} > \tau | y_{2N}) P(y_{2N})$$

where $i \neq N+1$, i.e., we exclude the output edge pixel, and $P(y_i)$ is the *a priori* probability of y_i occurring in the total input. Since all non-edge pixels are equally probable to occur, then

$$P(y_i) = \frac{1}{2N-1}$$

for all valid i (we use *2N-1* rather than *2N* as we exclude the edge pixel). Furthermore,

$$P(y_{ne} > \tau | y_i) = \int_{\tau}^{\infty} f_{y_i}(y)\, dy$$

where $f_{y_i}(y)$ was derived previously. Thus, the FA is

$$FA = \frac{1}{2N-1} \sum_{\substack{i=1 \\ i \neq N+1}}^{2N} \int_{\tau}^{\infty} f_{y_i}(y)\, dy.$$

Again, the value of $f_{y_i}(y)$ will be different for the upper and lower parts of the edge, and also for the non-edge pixels affected by the discontinuity.

A.3.1 Experimental Comparison

We compare the experimental and theoretical PD vs. FA curves for the SICNN as the ENR of the step edge input is varied. Figure A.8 shows the results when the input edge has Gaussian noise. The theoretical curves matches the experimental ones even for very small ENR. Any differences between the two curves are negligible. Figure A.9 shows the same comparison, but for a input with uniform noise. There is a small difference between the theoretical and experimental curves, particularly for low ENR. Overall, however, the match is very close. Figure A.10 is for multiplicative noise. Again, the two curves match extremely well, with only very small differences at low ENR inputs. The accuracy shown in all these three figures attests to the validity of the derived analytical expression for the PD and FA.

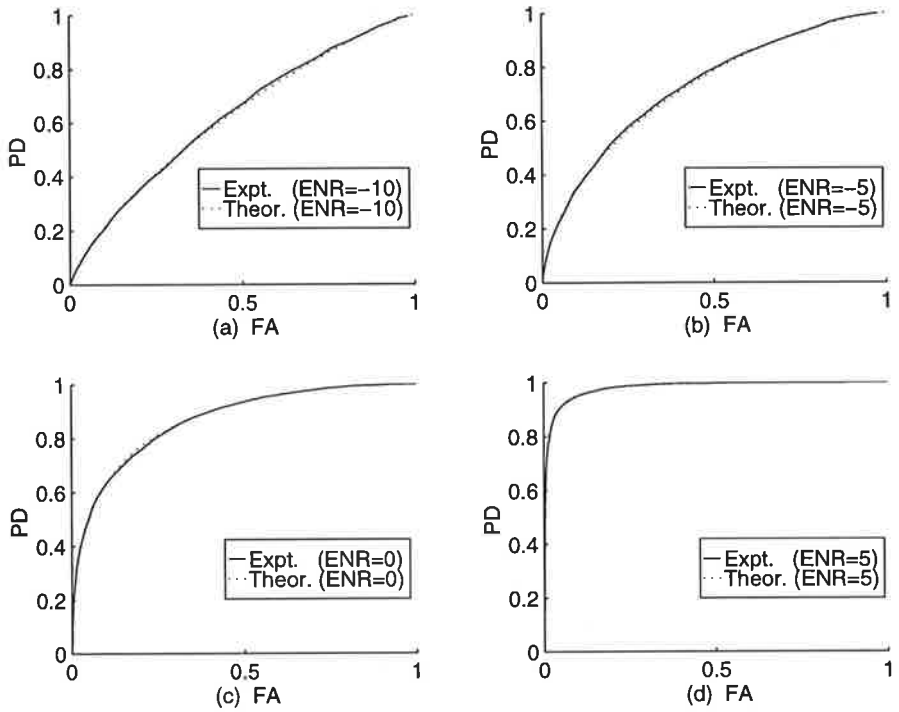


Figure A.8 Comparison of the theoretical and experimental SICNN PD vs. FA. The SICNN has asymmetrical rectangular weights, with $W = 1$, $r = 5$, optimal decay factor, $I_o = 10$, $c = 0.25$ and additive Gaussian noise.

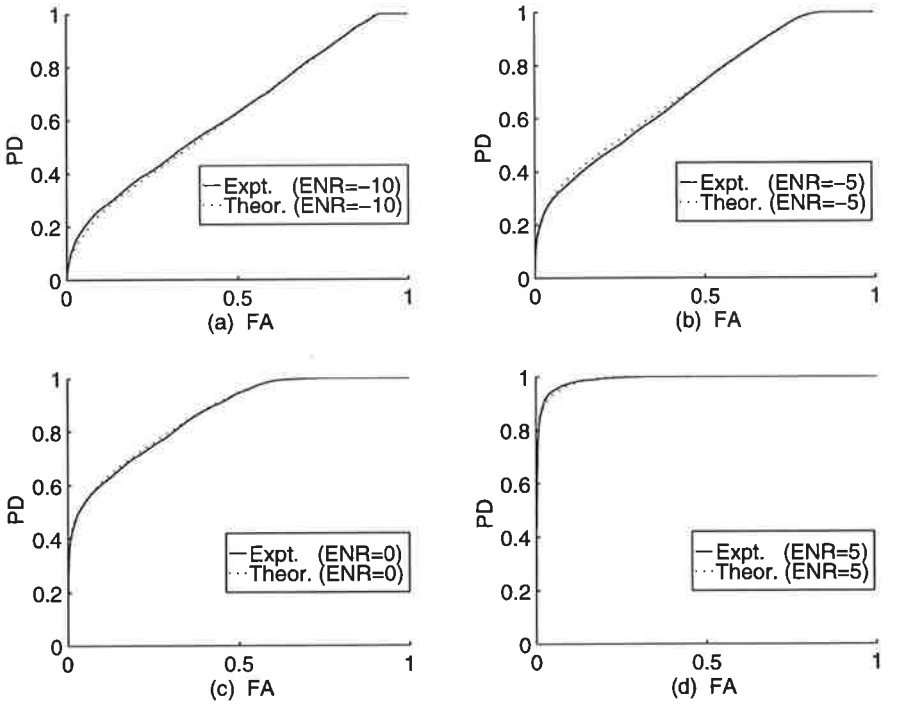


Figure A.9 As per Figure A.8 but for **uniform** noise.

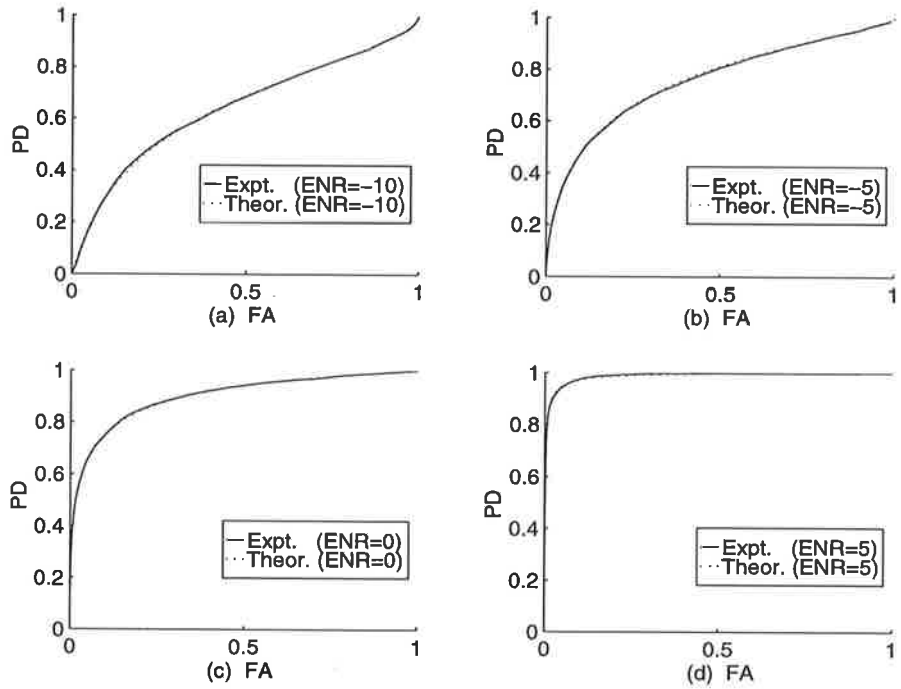


Figure A.10 As per Figure A.8 but for **multiplicative** noise.

Appendix B Estimation of Optimal Lambda

B.1 Estimation for the Rect., Gauss., and Triang. Weights

We now give the value of λ that optimises the SICNN performance with the rectangular, triangular and Gaussian weights. In 1-D, λ is chosen to minimises the EB over a range of ENR, while in 2-D, λ which maximises the area under the PD vs. FA curve.

B.1.1 One Dimensional Results

Without Gaussian Smoothing

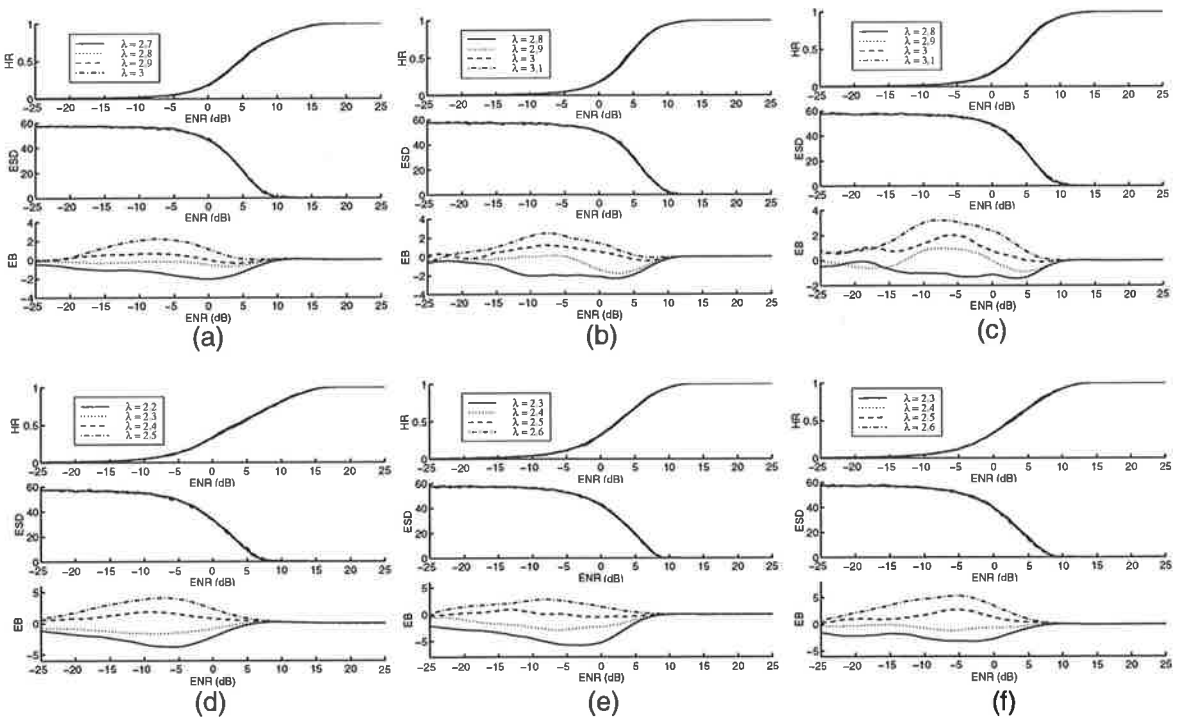


Figure B.1 1-D performance using four different values of λ, for asymmetrical rectangular, Gaussian, and triangular weights from left right. The SICNN has $r = 5$, $I_o = 10$, and $c = 0.25$. Top row is for Gaussian noise and the bottom row is for uniform noise.

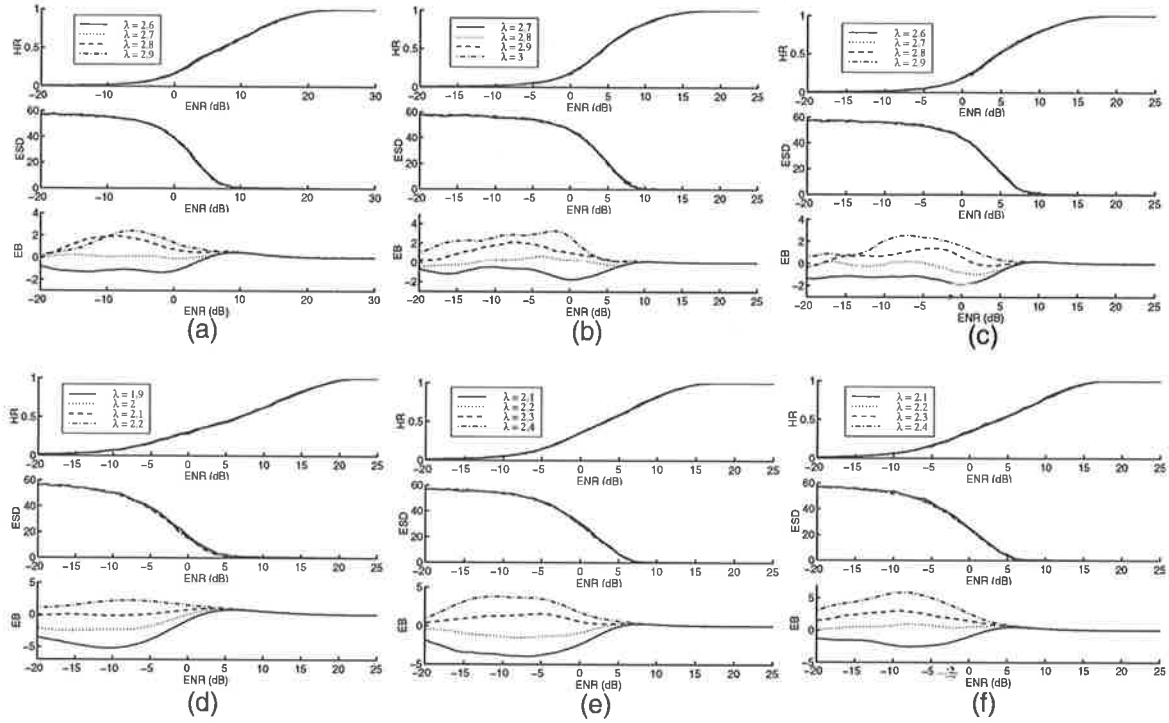


Figure B.2 As per Figure B.1 but for a SICNN with *r = 10.*

With Gaussian Smoothing

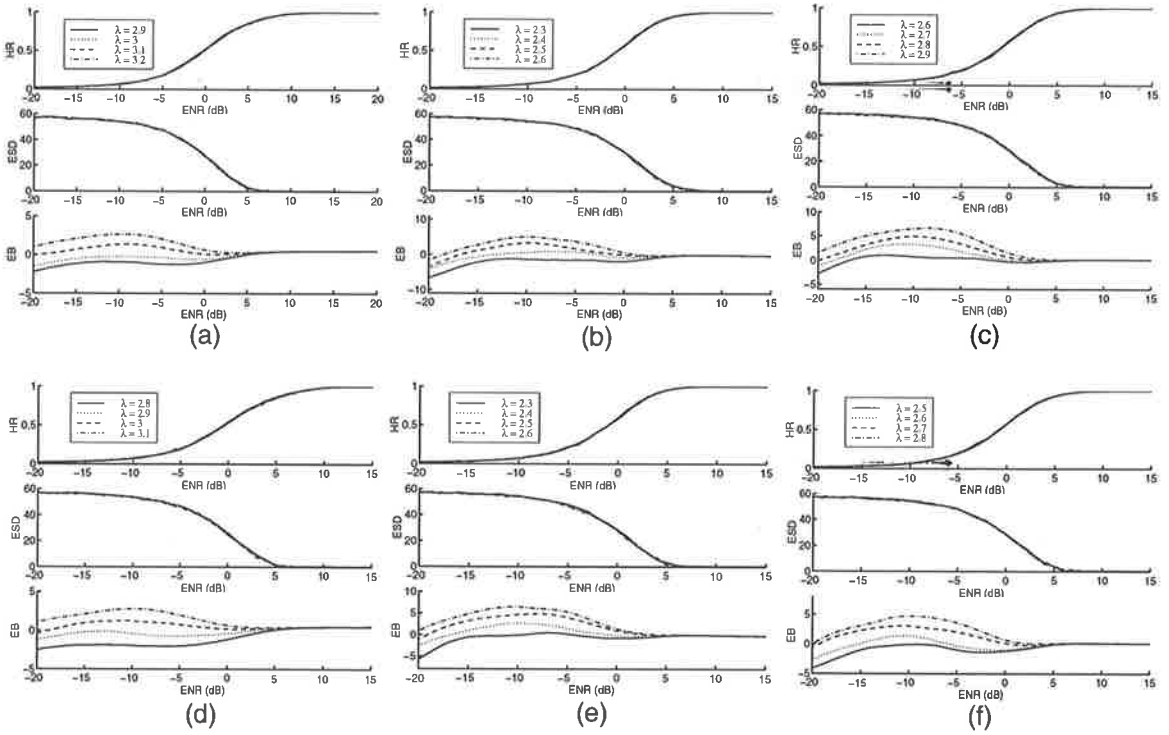


Figure B.3 As per Figure B.1 but with Gaussian smoothing of length *2r+1 = 11.*

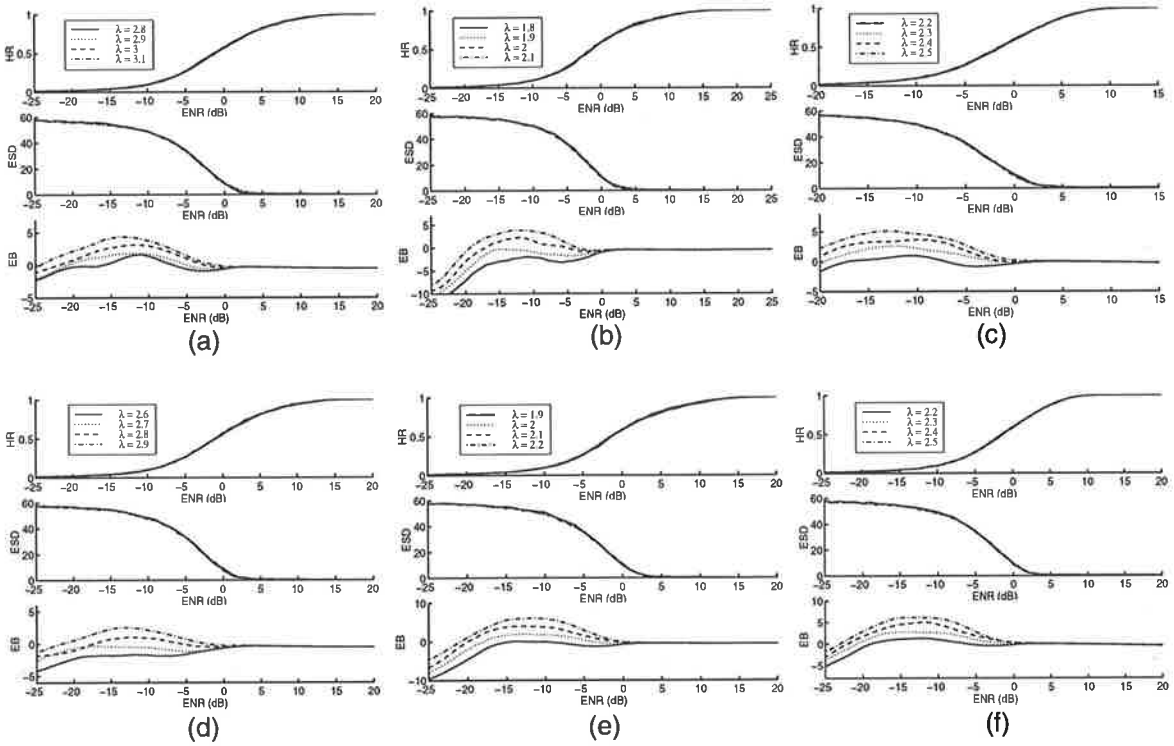


Figure B.4 As per Figure B.1 but with Gaussian smoothing of length *2r+1 = 21*.

B.1.2 Two Dimensional Results

Without Gaussian Smoothing

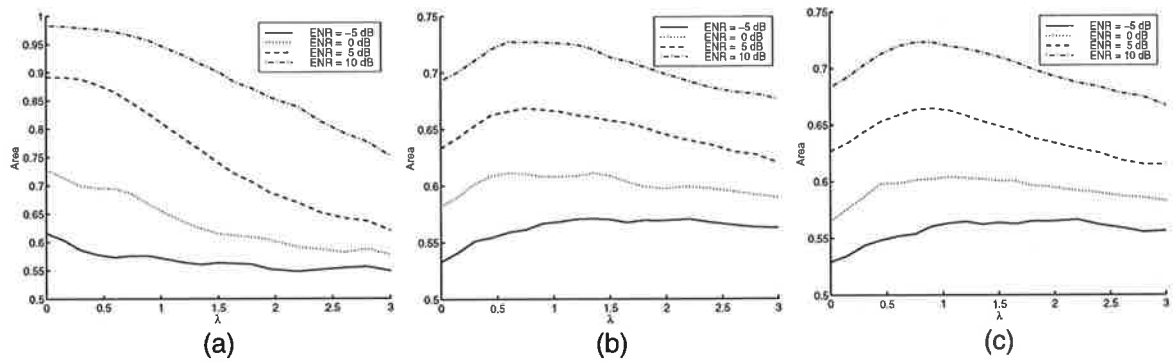


Figure B.5 2-D performance for *r = 1* as a function of λ for the synthetic input image with (a) multiplicative, (b) Gaussian and (c) uniform noise.

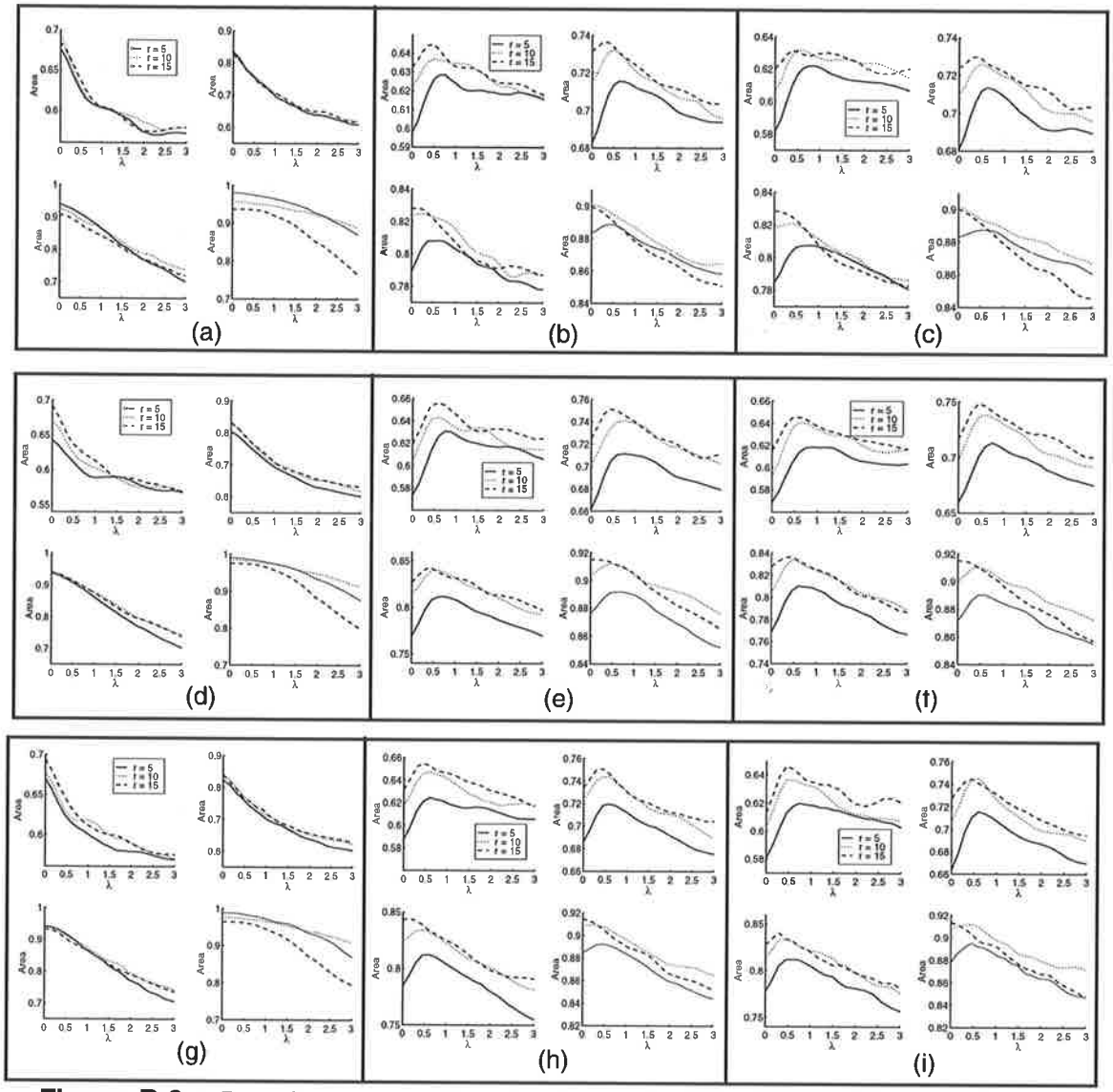


Figure B.6 2-D performance as λ varies for inputs with multiplicative, Gaussian and uniform noise from the left column to the right one, respectively. The SICNN has asymmetrical, rectangular, Gaussian, and triangular weights from top row to bottom, respectively. The ENR for each subplot is, from L-R and T-B, -5, 0, 5 and 10 dB.

With Gaussian Smoothing

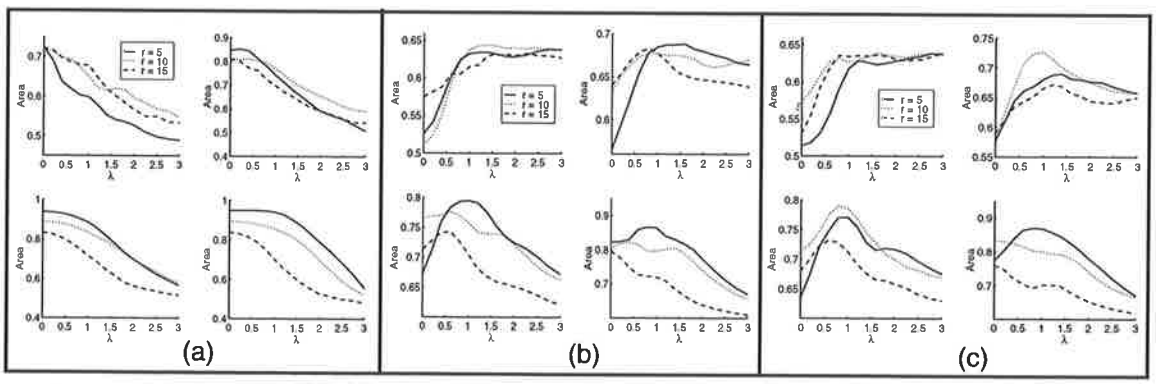


Figure B.7 As per Figure B.6 but with Gaussian smoothing of length *2r+1 = 11*.

B.2 Estimation for the Optimal Weight Distribution

In this section we present the value of λ that optimises the SICNN with the optimal weight distribution's performance. In 1-D, for the optimal weight distribution with attenuation factor of 0.5, 1 and 2, we give the values of λ which minimises the EB over the range of ENR. In 2-D, we give both the value of the attenuation factor and λ which maximises the area under its PD vs. FA curve. Results are also given for Gaussian smoothing.

B.2.1 One Dimension Results

No Smoothing

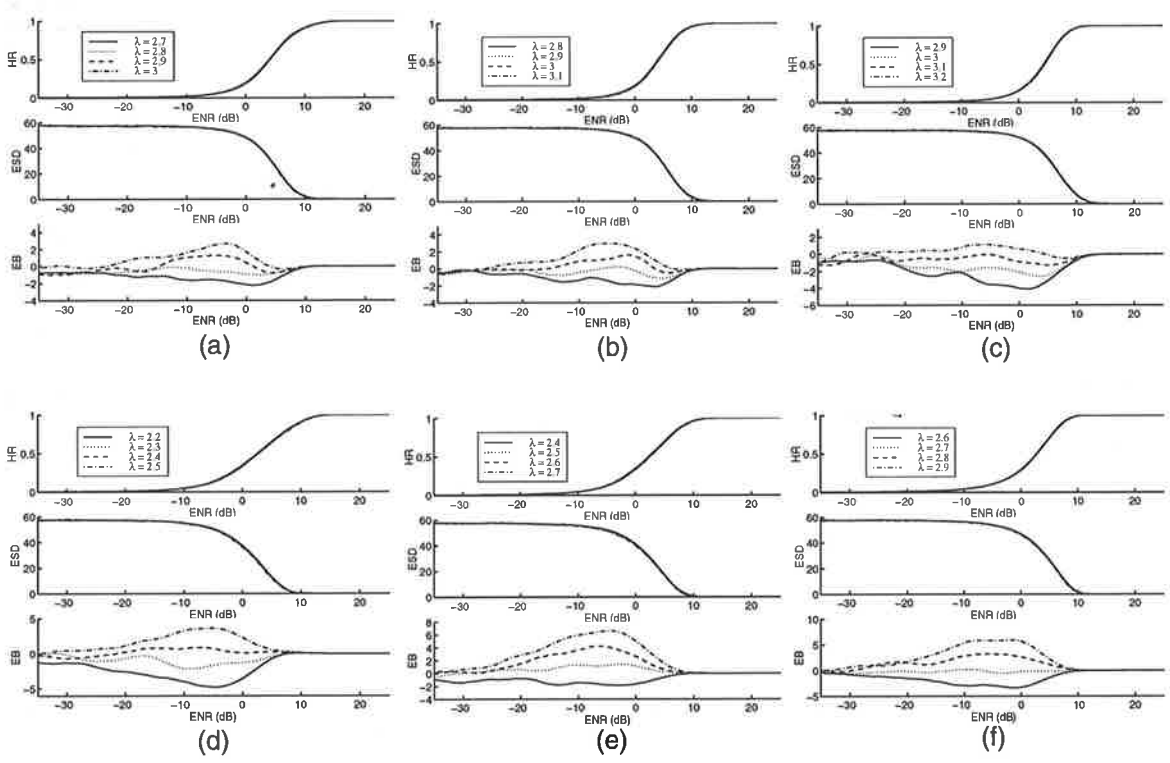


Figure B.8 1-D performance for four different values of λ, with asymmetrical, optimal weights with attenuation factor 0.5, 1, and 2 from left column to right one, respectively. The SICNN has $r = 5$, and $I_o = 10$, $c = 0.25$. The top row is for Gaussian noise, and the bottom row is for uniform noise.

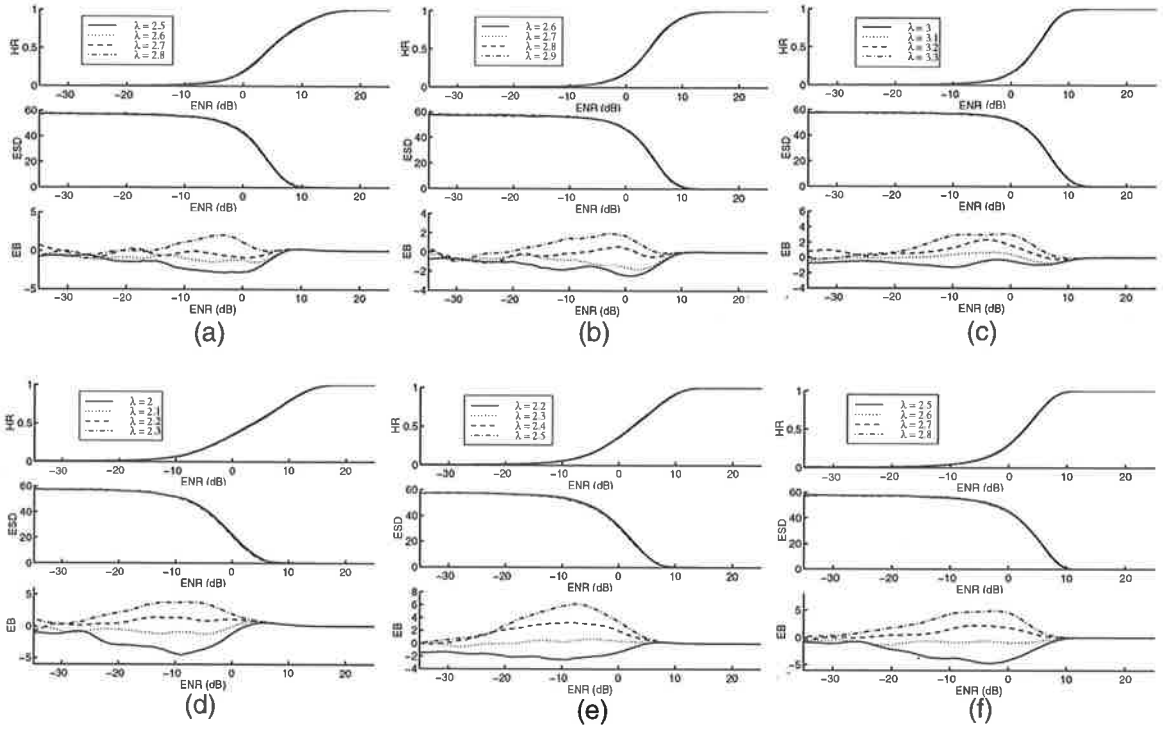


Figure B.9 As per Figure B.8 but for a neighbourhood size of $r = 10$.

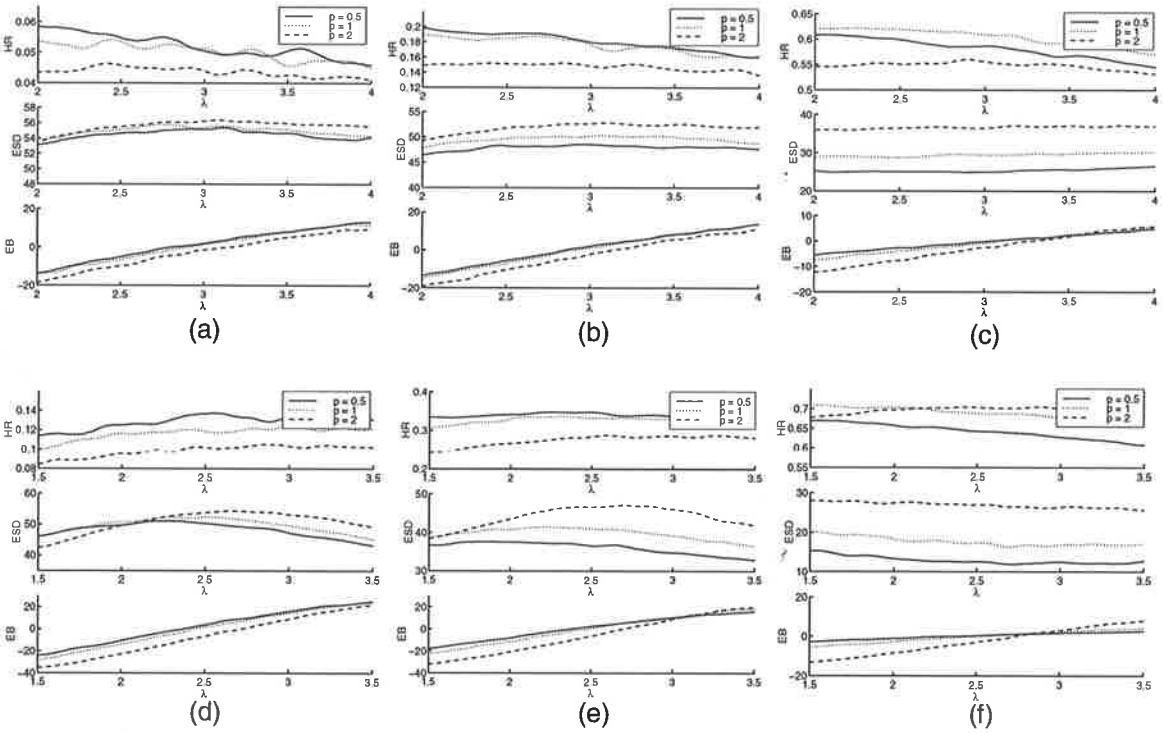


Figure B.10 1-D performance as a function of λ for step edge inputs with ENR of -5 dB, 0 dB, and 5 dB from the left column to the right one, respectively. The SICNN has asymmetrical, optimal weights with $p = 0.5, 1, 2$, $r = 5$ and $I_o = 10$, $c = 0.25$. The top row is for Gaussian noise and the bottom row is for uniform noise.

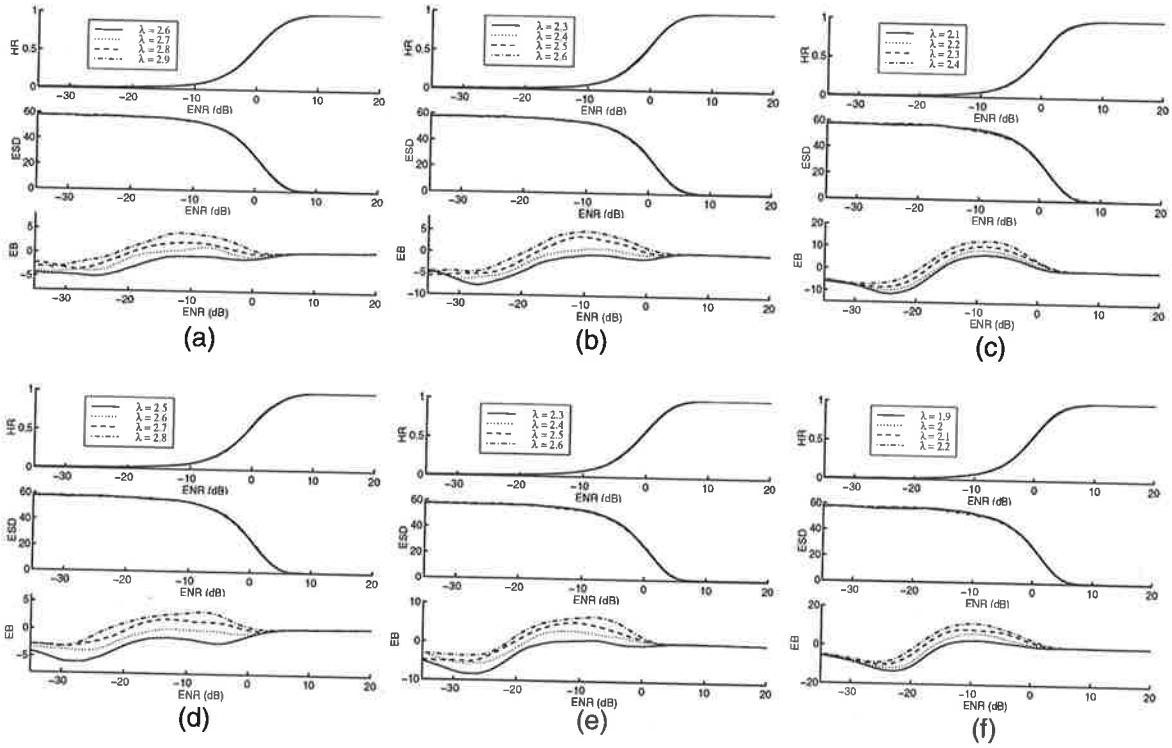


Figure B.11 1-D SICNN performance with smoothing for different values of λ and asymmetrical, optimal weights with p of 0.5, 1, and 2 from the left column to right one, respectively. The SICNN has $r = 5$, with $I_o = 10$, $c = 0.25$ and the Gaussian filter has length $2r + 1 = 11$. The top row is for Gaussian noise and the bottom row is for uniform noise.

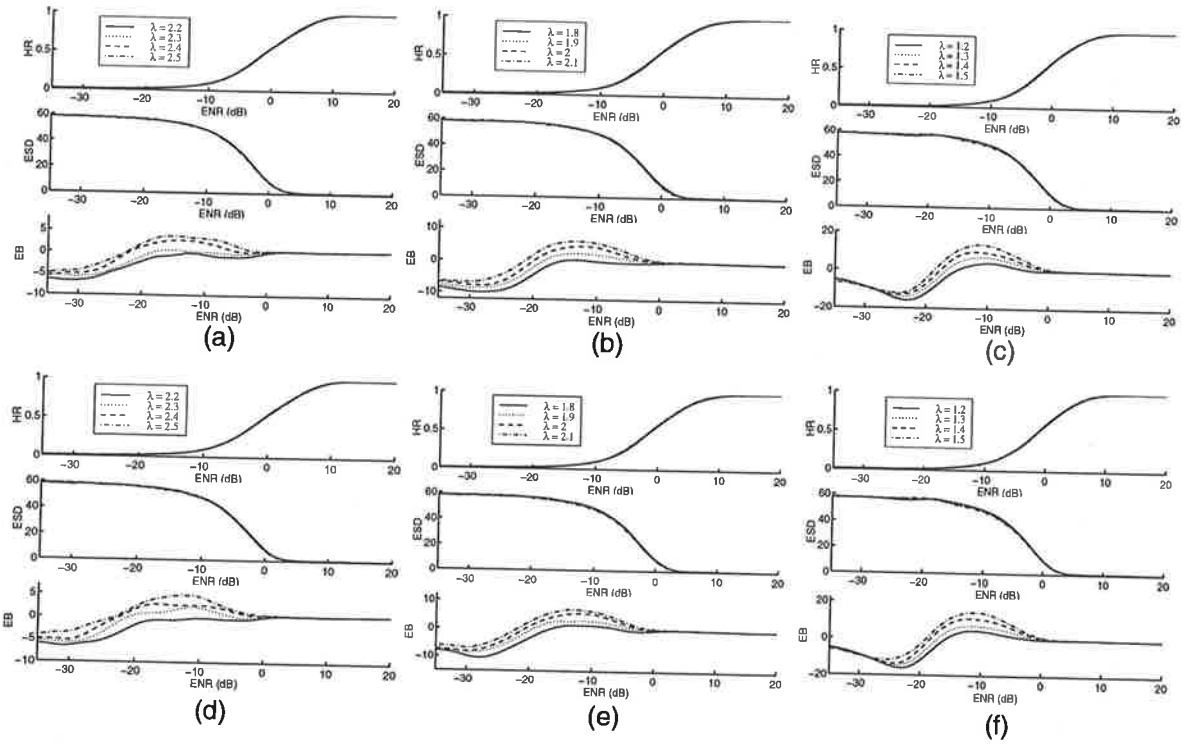


Figure B.12 As per Figure B.11 but for $r = 10$.

B.2.2 Two Dimensional Results

No Smoothing

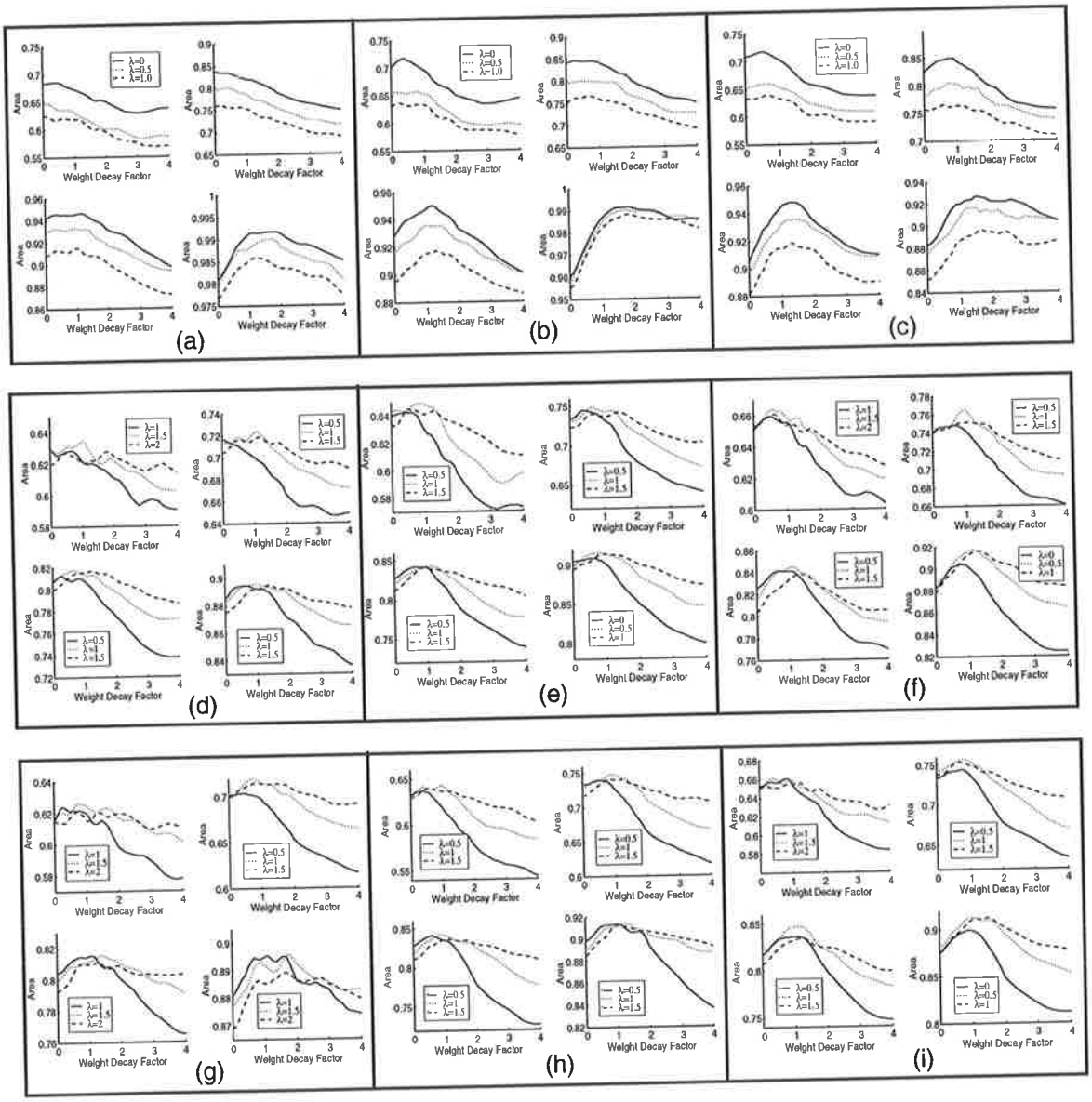


Figure B.13 2-D area under the PD vs. FA curve for the synthetic image for r of 5, 10 and 15 (from left to right). The results are for 3 values of λ that give the best performance as the attenuation factor is varied. Each subplot shows the results, from L-to-R and T-to-B, for ENR of -5, 0, 5 and 10 dB, respectively. The top, middle, and bottom rows are for multiplicative, Gaussian, and uniform noise, respectively.

With Gaussian Smoothing

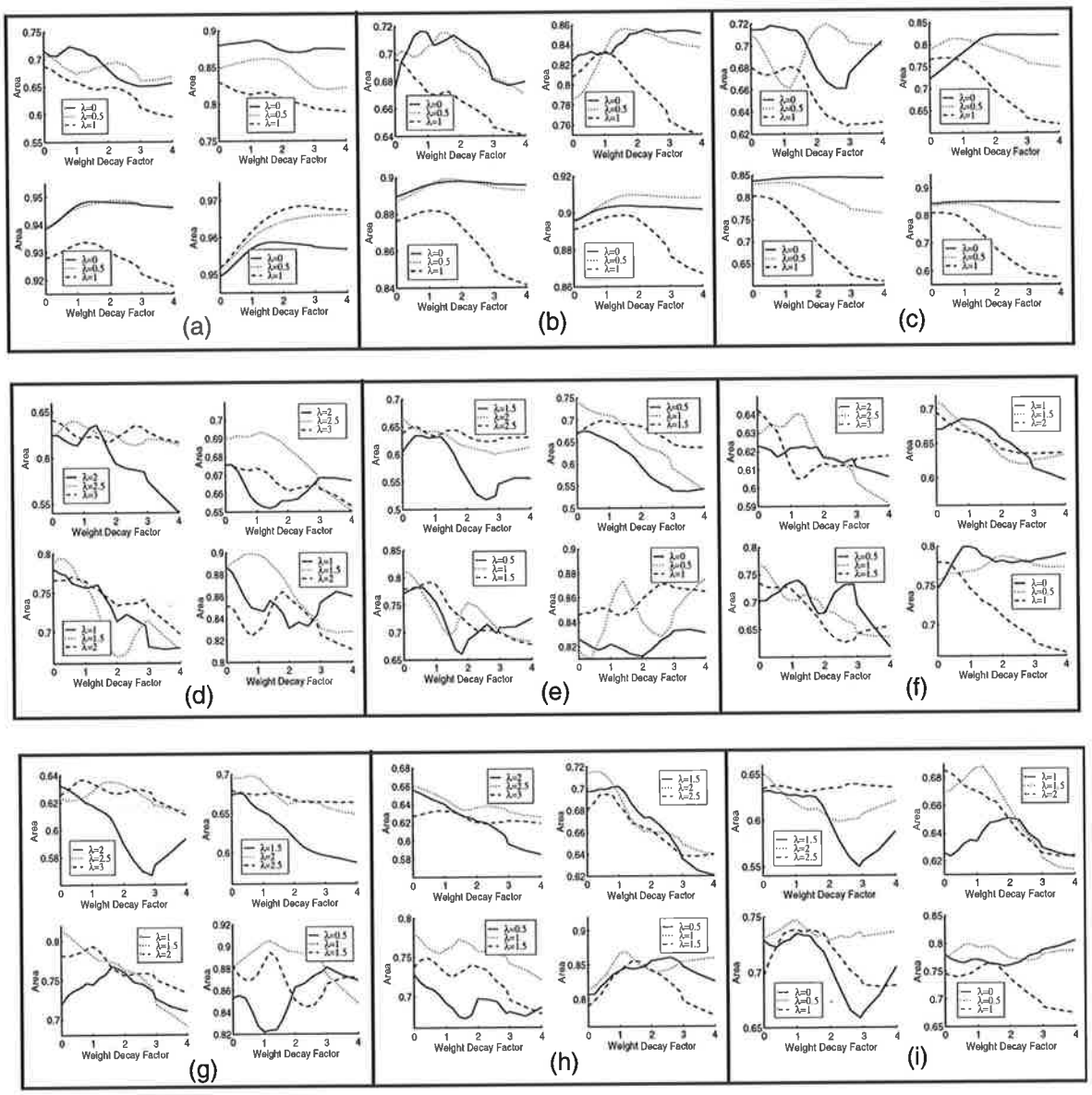


Figure B.14 As per Figure B.13 but Gaussian smoothing of length *2r+1*.

Appendix C Postprocessing Results

C.1 Scale Combination

C.1.1 One Dimension

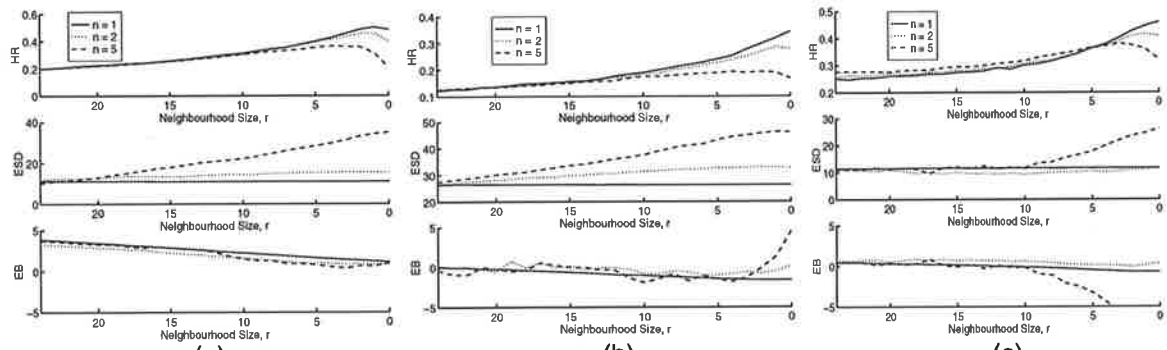


Figure C.1 1-D scale combination for 1, 2, and 5 largest maxima to determine the valid region. The SICNN has asymmetrical, rectangular weights, with $I_o = 10$, $c = 0.25$, $ENR = 0$ dB, and (a) multiplicative, (b) Gaussian, and (c) uniform noises.

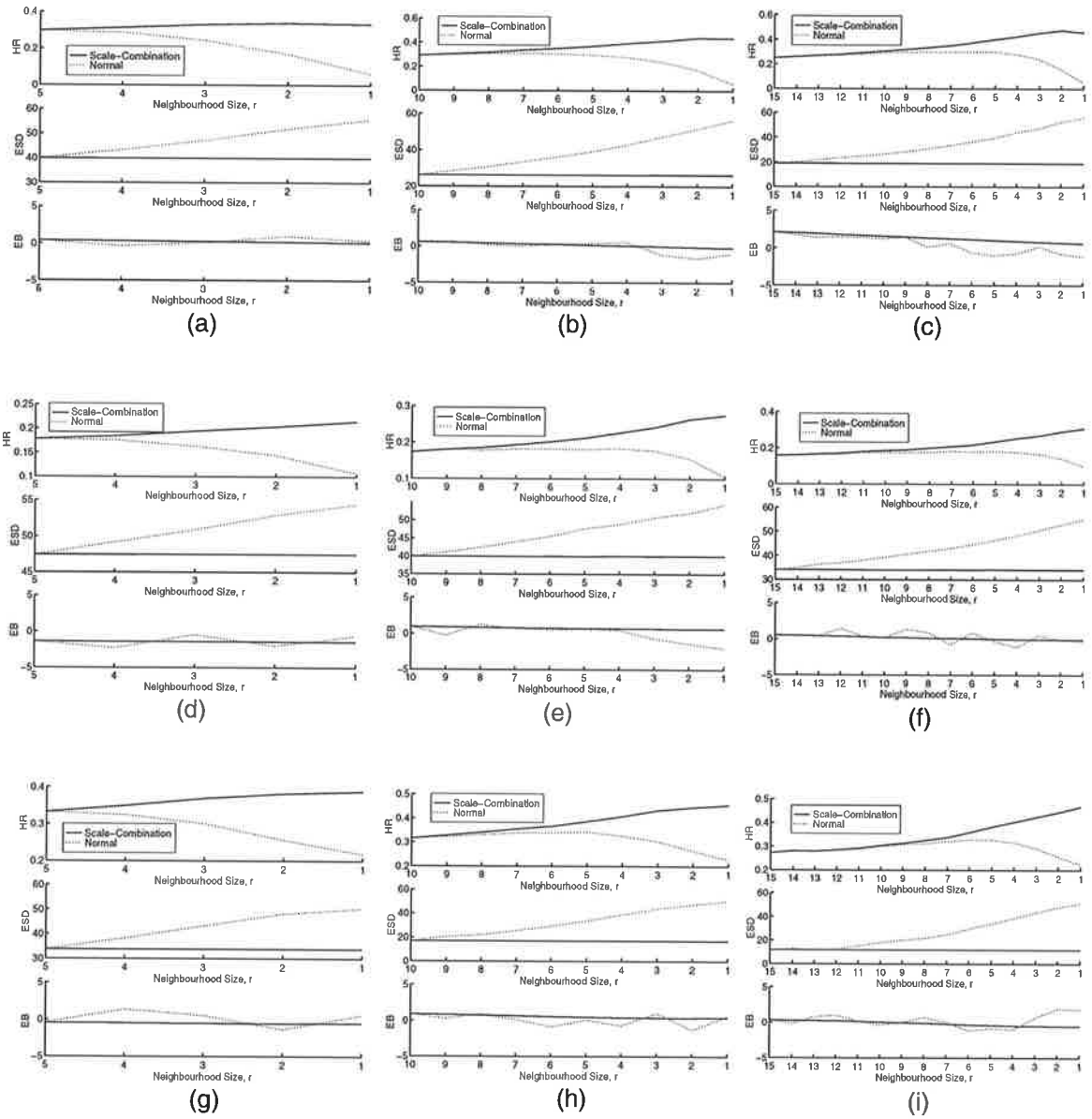


Figure C.2 1-D performance with and without scale combination for r decreasing from 5 down to 1, 10 down to 1, and 15 down to 1 for the leftmost column to the rightmost one, respectively. The weights are asymmetrical, rectangular, with $I_o = 10$, $c = 0.25$, $ENR = 0$ dB. The top, middle and bottom rows are for multiplicative, Gaussian and uniform noise, respectively.

C.1.2 Two Dimensions

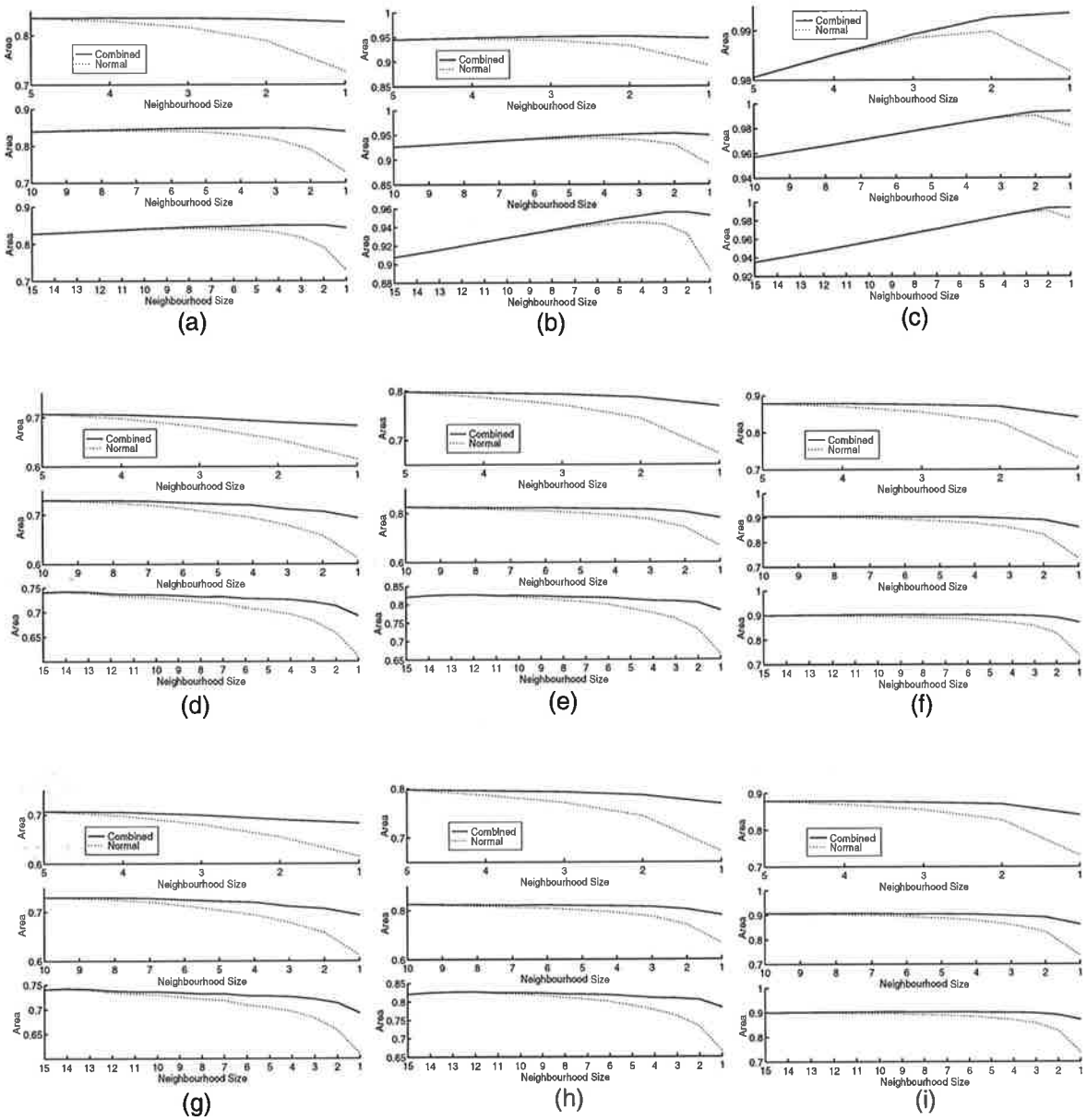


Figure C.3 2-D performance with and without scale-combination for ENR of 0 dB, 5 dB, and 10 dB from the leftmost column to the rightmost one, respectively. The SICNN has asymmetrical, **rectangular** weights, and optimal decay factor. The top, middle, and bottom rows are for multiplicative, Gaussian and uniform noise, respectively.

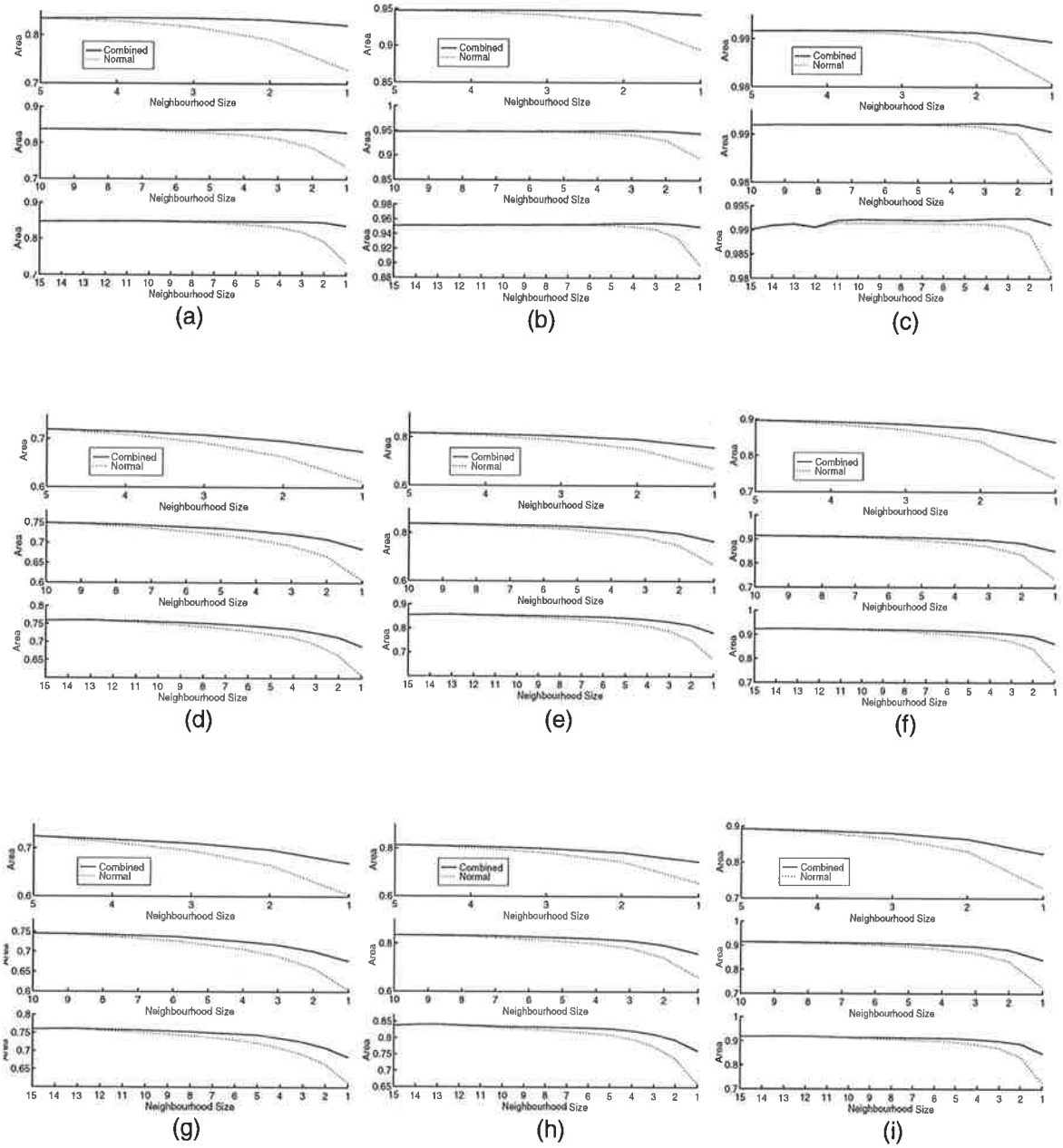


Figure C.4 As per Figure C.3 but for the asymmetrical, optimal weight distribution.

C.2 Weight Variation Combination

C.2.1 One Dimension

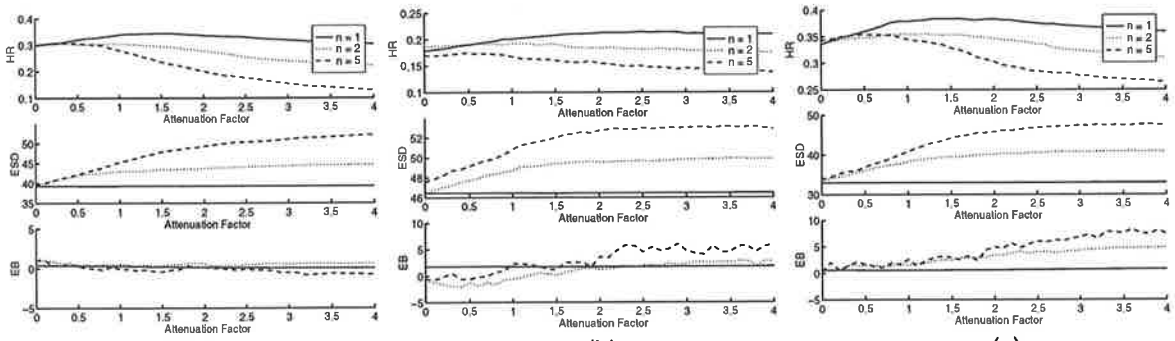


Figure C.5 1-D performance for weight variation combination of different number of maxima for (a) multiplicative, (b) Gaussian, and (c) uniform noises. The SICNN has asymmetrical, rectangular weights, $r = 5$, $I_o = 10$, $c = 0.25$ and $ENR = 0$dB.

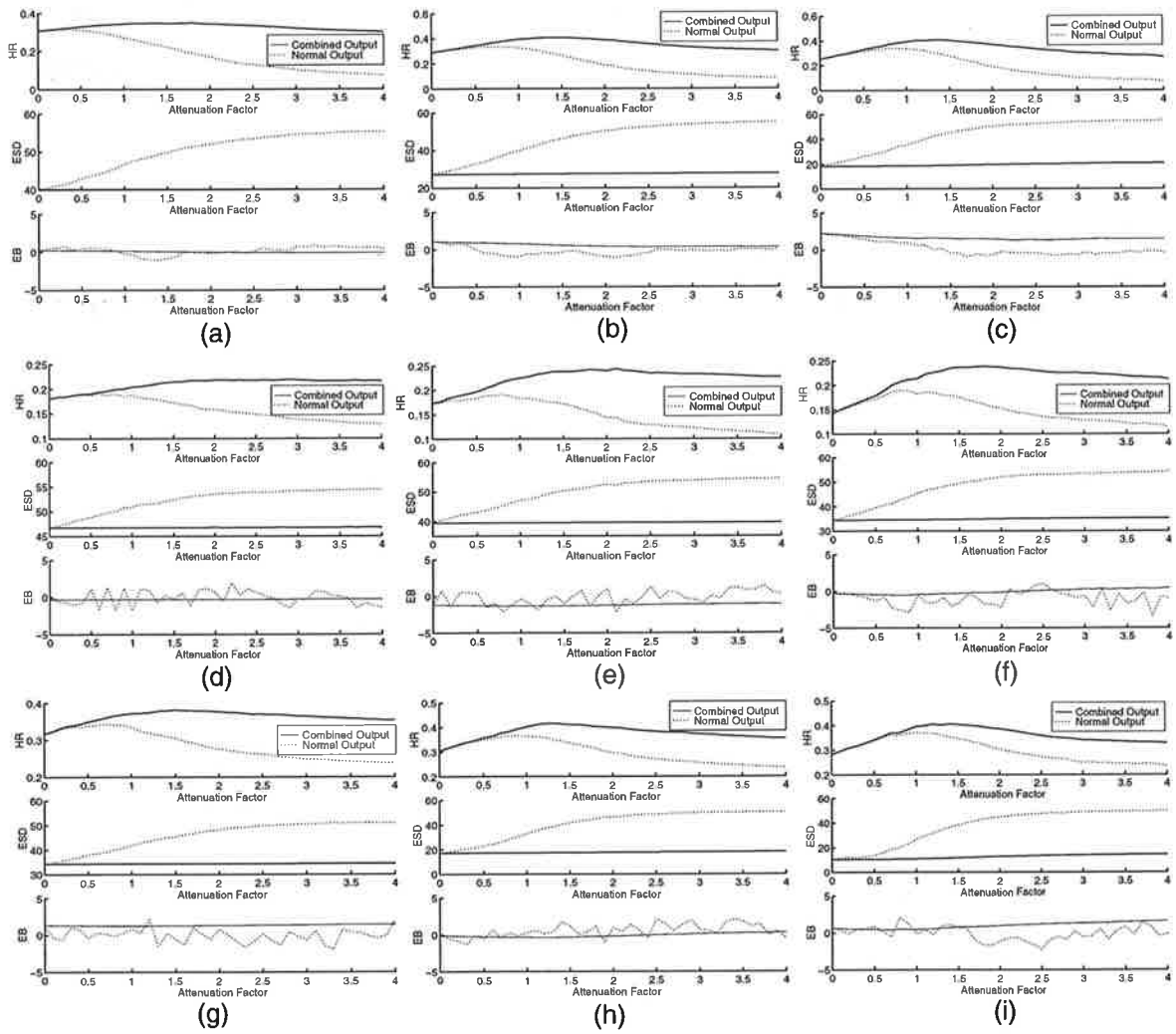


Figure C.6 2-D Performance with and without weight variation combination for the optimal weights with r of 5, 10 and 15 from the leftmost column to the rightmost one, respectively, and $I_o = 10$, $c = 0.25$, $ENR = 0$dB. The top, middle and bottom row are for multiplicative, Gaussian, and uniform noises.

C.2.2 Two Dimensions

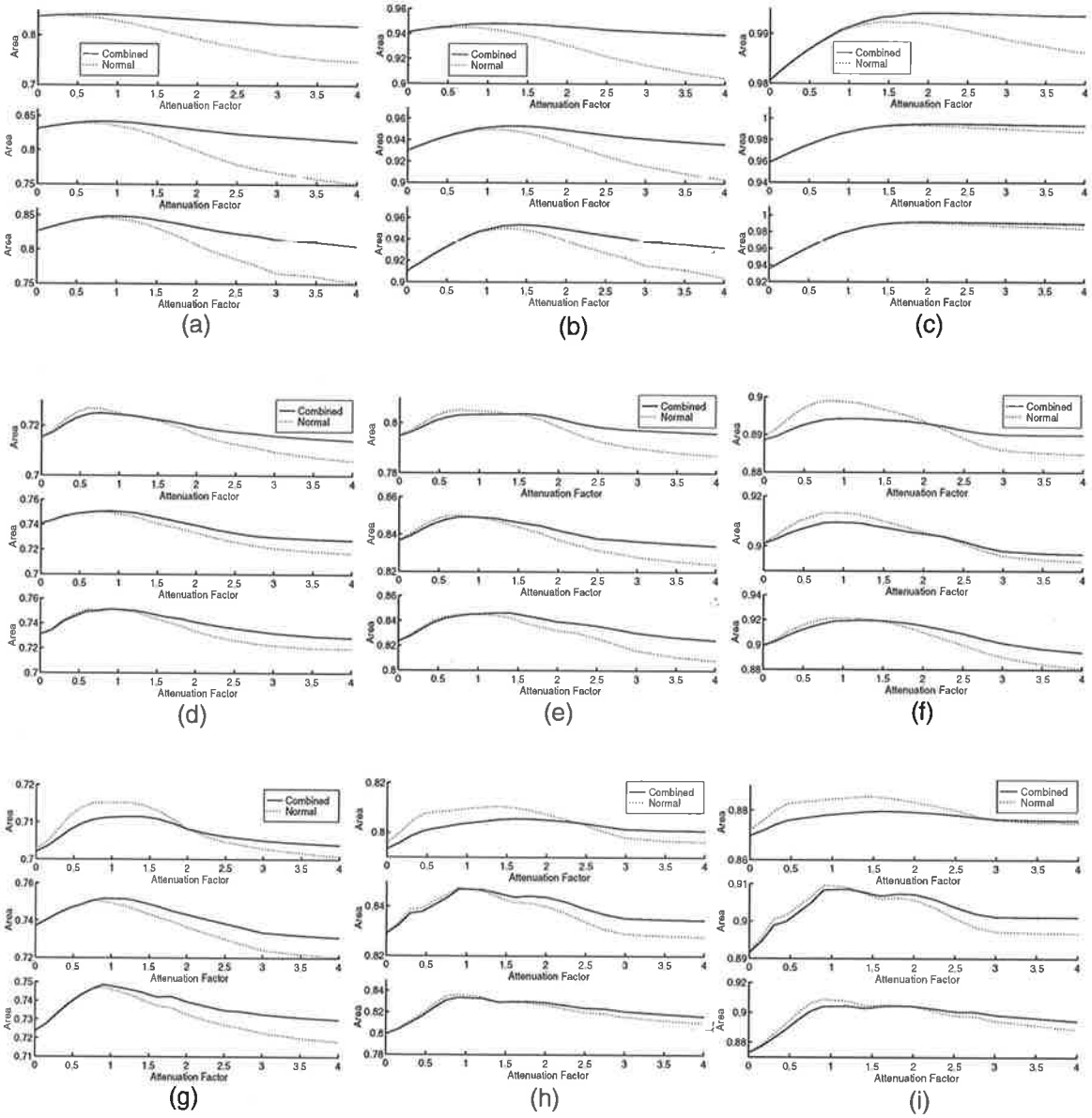


Figure C.7 2-D performance with and without weight variation combination with the optimal weights for ENR of 0 dB, 5 dB, and 10 dB from the left most column to the rightmost one, respectively. Each subplot shows the results for $r = 5$, $r = 10$ and $r = 15$, from top to bottom. The optimal decay factor is used in each case. The top, middle, and bottom rows are for multiplicative, Gaussian and uniform noise, respectively.

C.3 Combination of Different SICNN Outputs

C.3.1 One Dimension

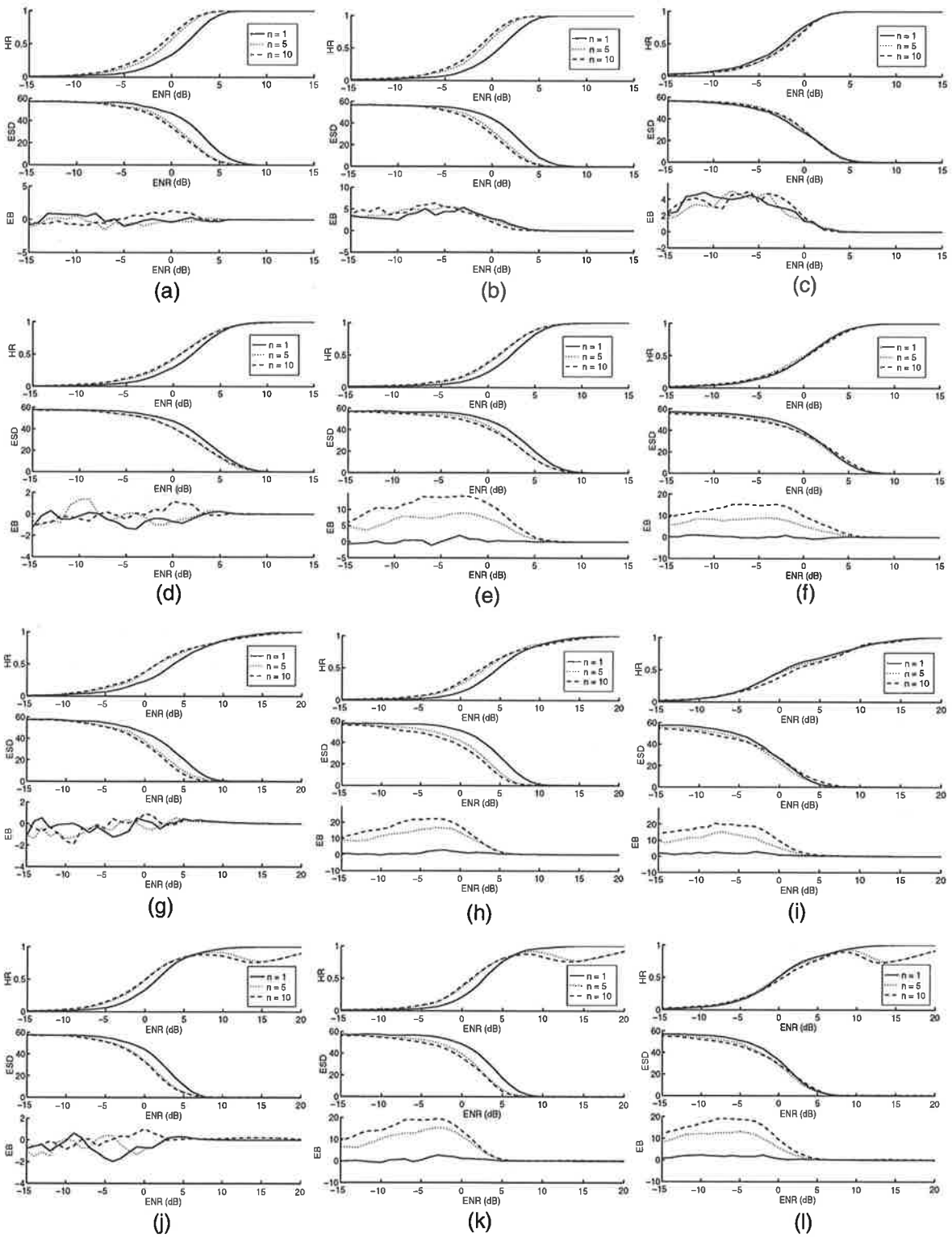


Figure C.8 1-D performance for the MAW-nZCSW, ZCSW-nMAW, MAW-nMSW, and MSW-nMAW (from top row to bottom, respectively) combination scheme for multiplicative, Gaussian and uniform noise (from left column to right one, respectively). The SICNN has a rectangular, asymmetrical weights, with $r = 5$, $I_o = 10$, $c = 0.25$ and $ENR = 0$dB.

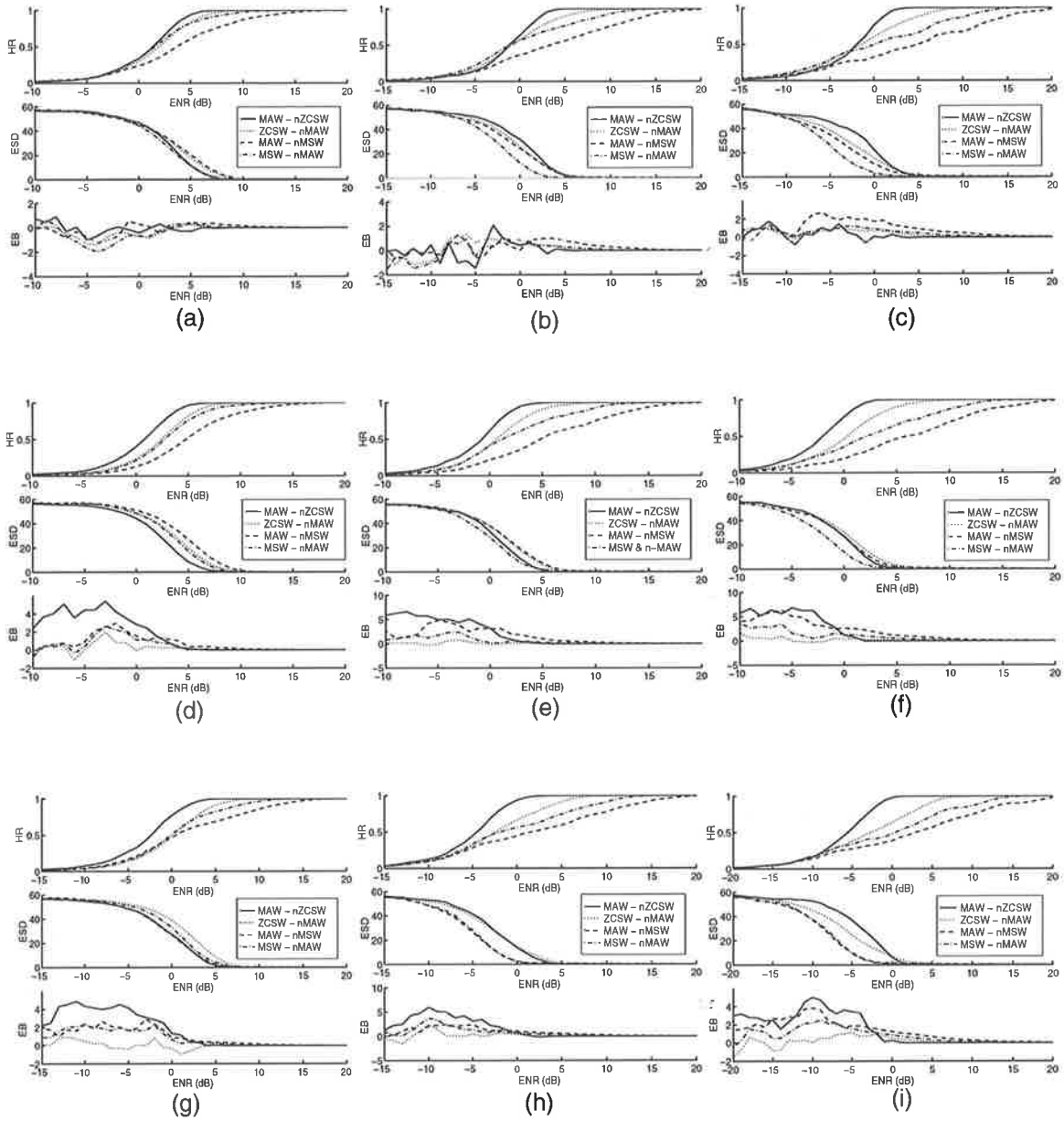


Figure C.9 1-D comparison of the different combination schemes for r of 5, 10, and 15 from the leftmost column to the rightmost one, respectively. The SICNN has asymmetrical, rectangular weights with $I_o = 10$, $c = 0.25$, and $ENR = 0$ dB. The top, middle, and bottom row are for multiplicative, Gaussian, and uniform noise respectively.

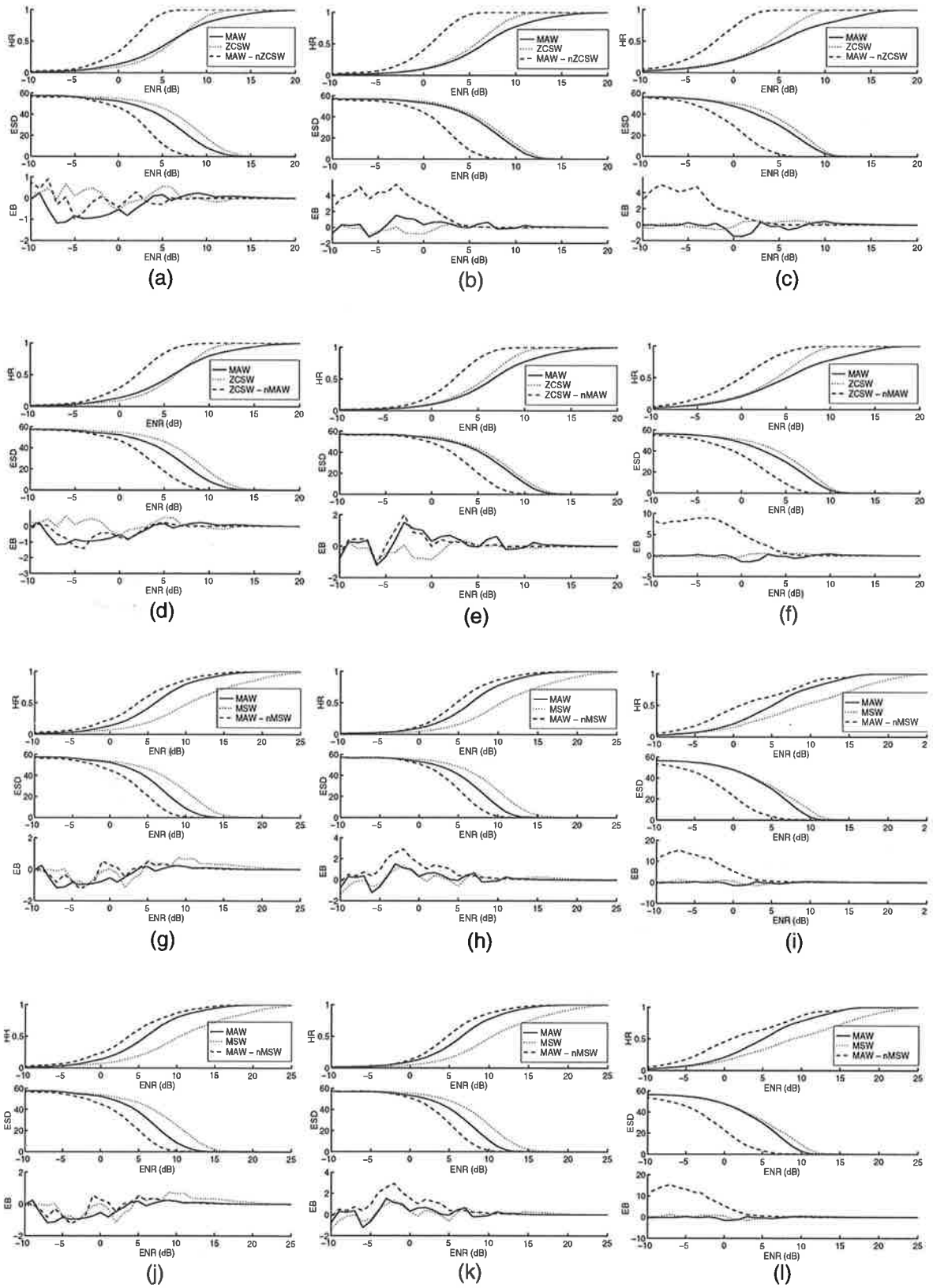


Figure C.10 1-D comparison for the MAW & n-ZCSW, ZCSW & n-MAW, MAW & n-MSW, and MSW & n-MAW from the top row to bottom one, respectively. The noise is multiplicative, Gaussian and uniform noise from the left column to the right one, respectively. The SICNNs has rectangular weights, with $r = 5$, $I_o = 10$, and $c = 0.25$.

C.3.2 Two Dimensions

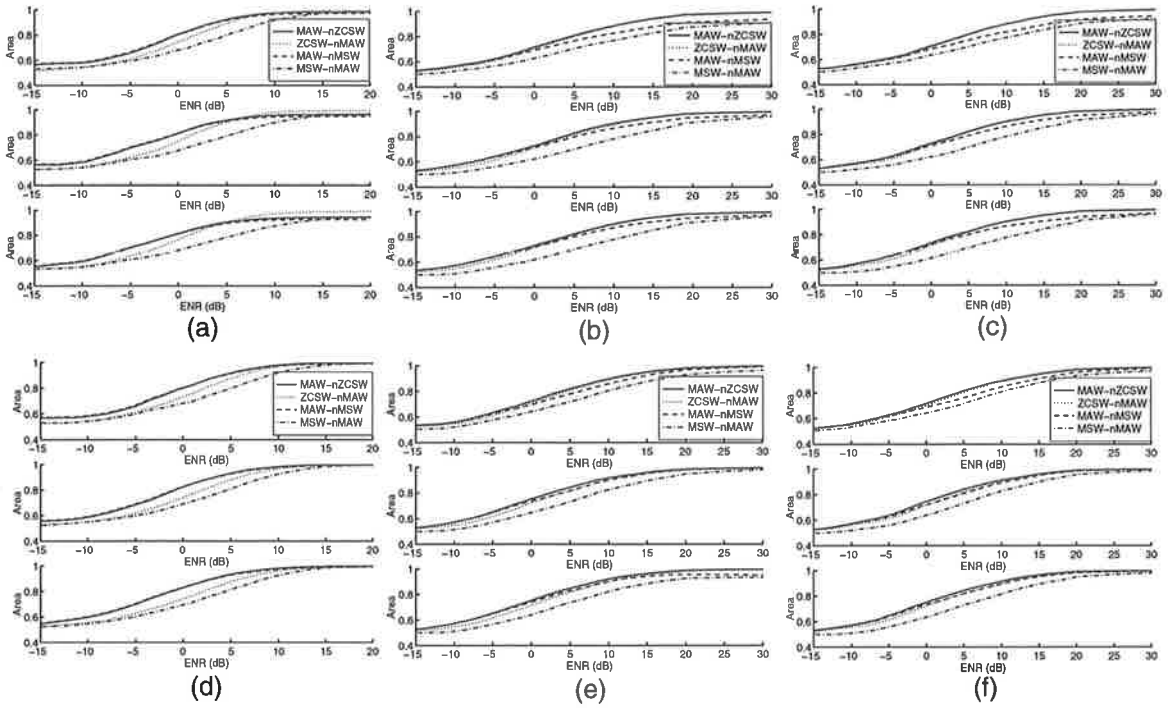


Figure C.11 2-D performance of different combination schemes for multiplicative, Gaussian, and uniform noise (from left column to right one). Each subplot shows the results for r of 5, 10 and 15 from top to bottom. The SICNN has optimal decay factor and asymm. rect. weights (top row) and optimal weights (bottom row).

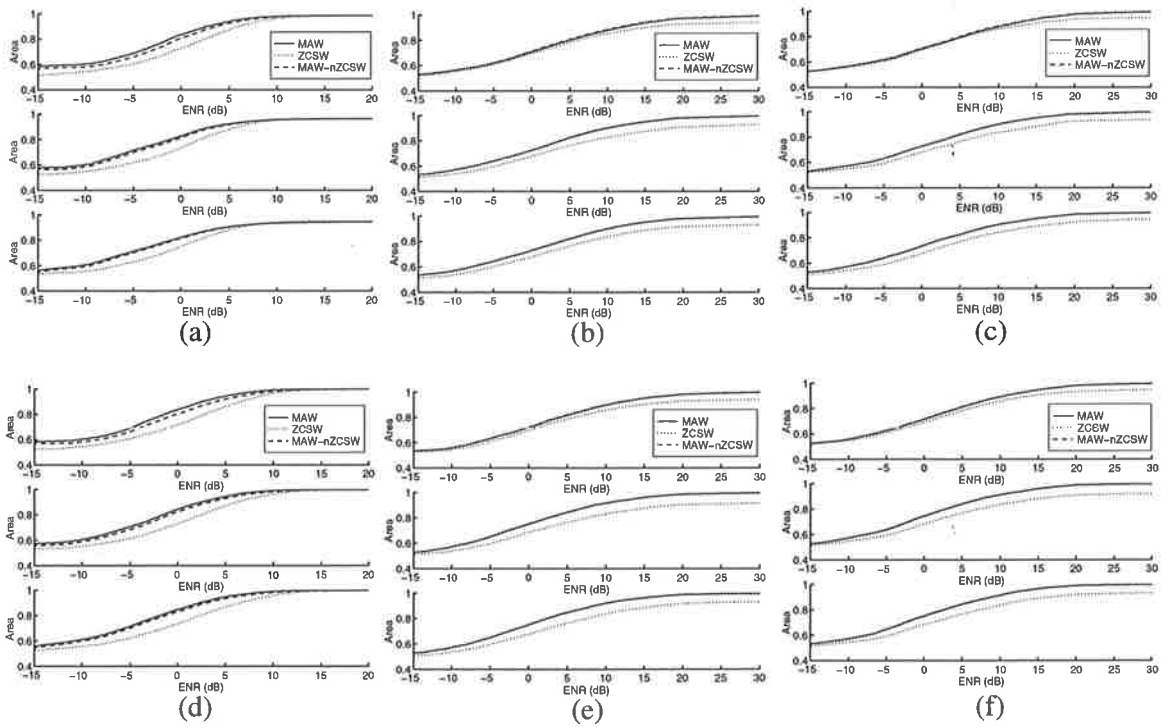


Figure C.12 2-D comparison of **MAW, ZCSW and MAW-nZCSW** methods for multiplicative, Gaussian, and uniform noise (left to right, respectively). Each subplot shows the results for r of 5, 10 and 15, from top to bottom. Top row is for rect. weights and bottom row is for optimal weights.

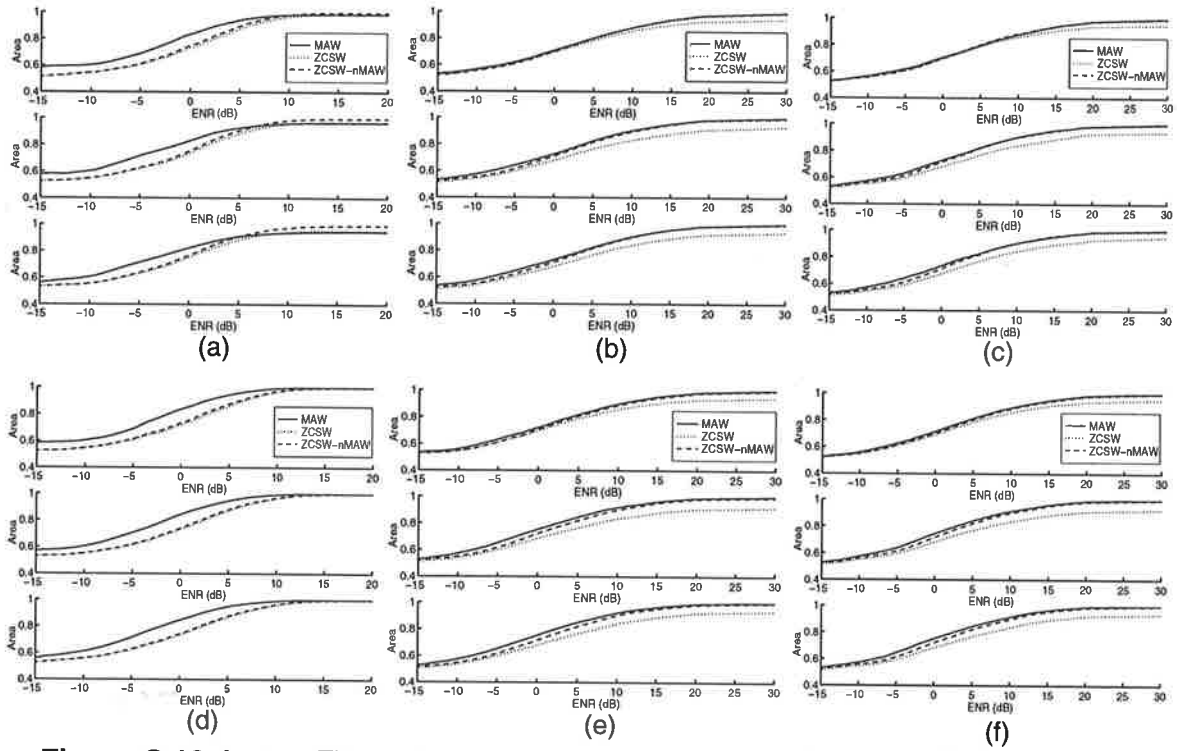


Figure C.13 As per Figure C.12 but for a comparison between **MAW, ZCSW** and **ZCSW-nMAW**.

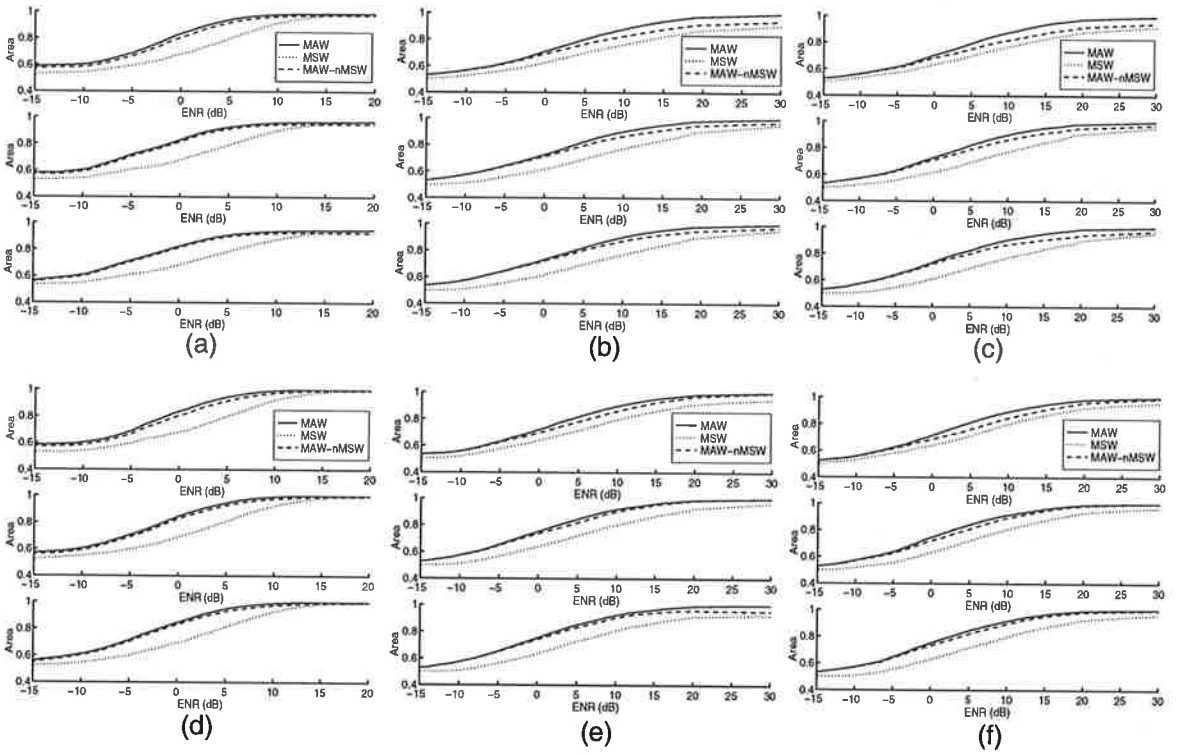


Figure C.14 As per Figure C.12 but for a comparison between **MAW, MSW** and **MAW-nMSW**.

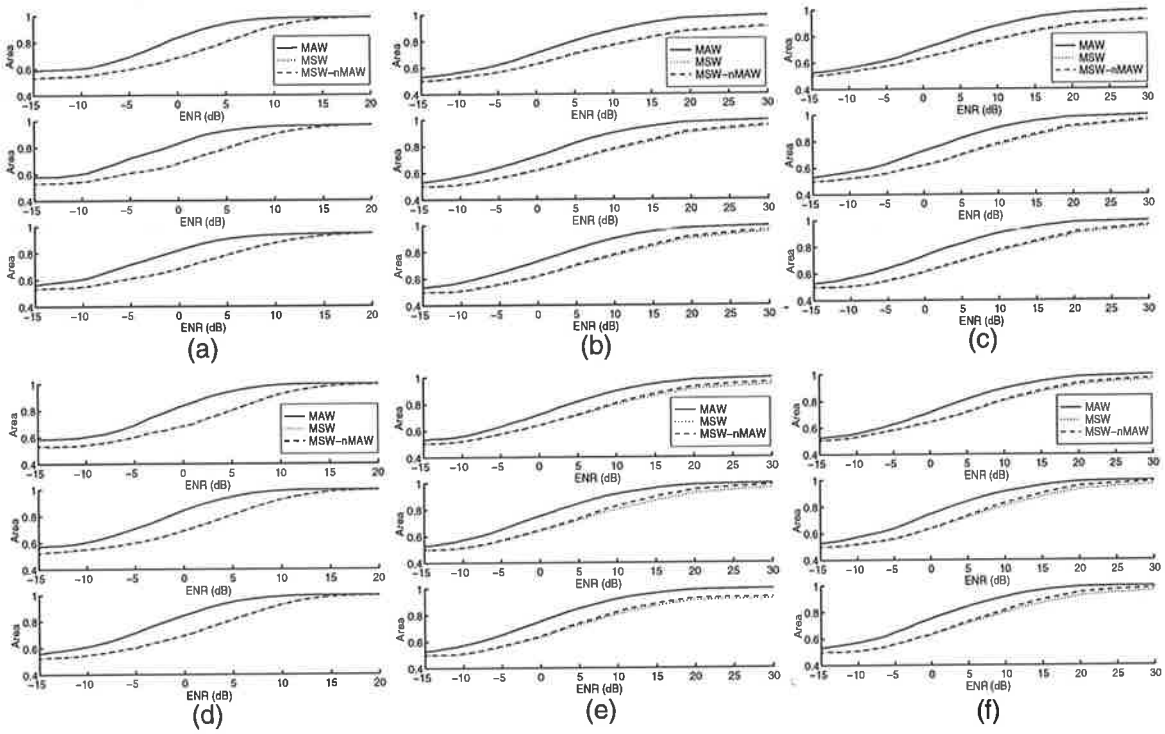


Figure C.15 As per Figure C.12 but for a comparison between **MAW, MSW** and **MSW-nMAW**.

C.4 Complementary Output Processing

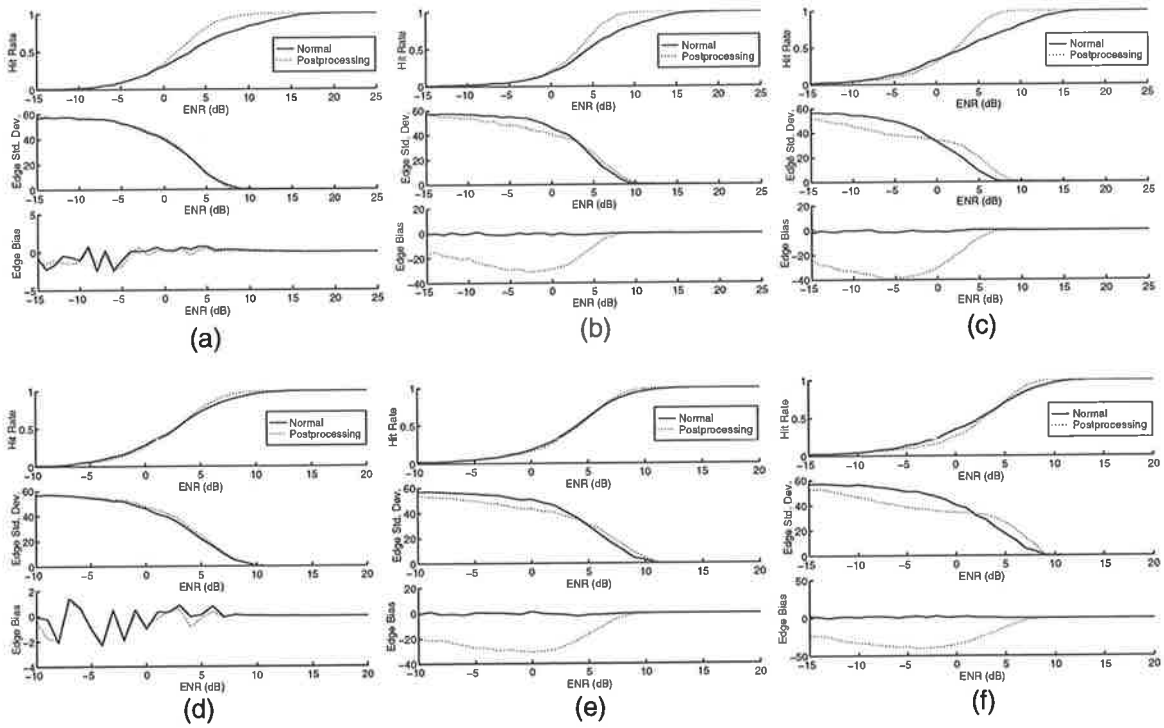


Figure C.16 1-D performance with complementary output postprocessing for multiplicative, Gaussian, and uniform noise (from the left column to right one). The SICNN has $r = 5$, and optimal decay factor, and $I_o = 10$, $c = 0.25$. Top row is for asymmetrical rectangular weights and bottom row is for optimal weights.

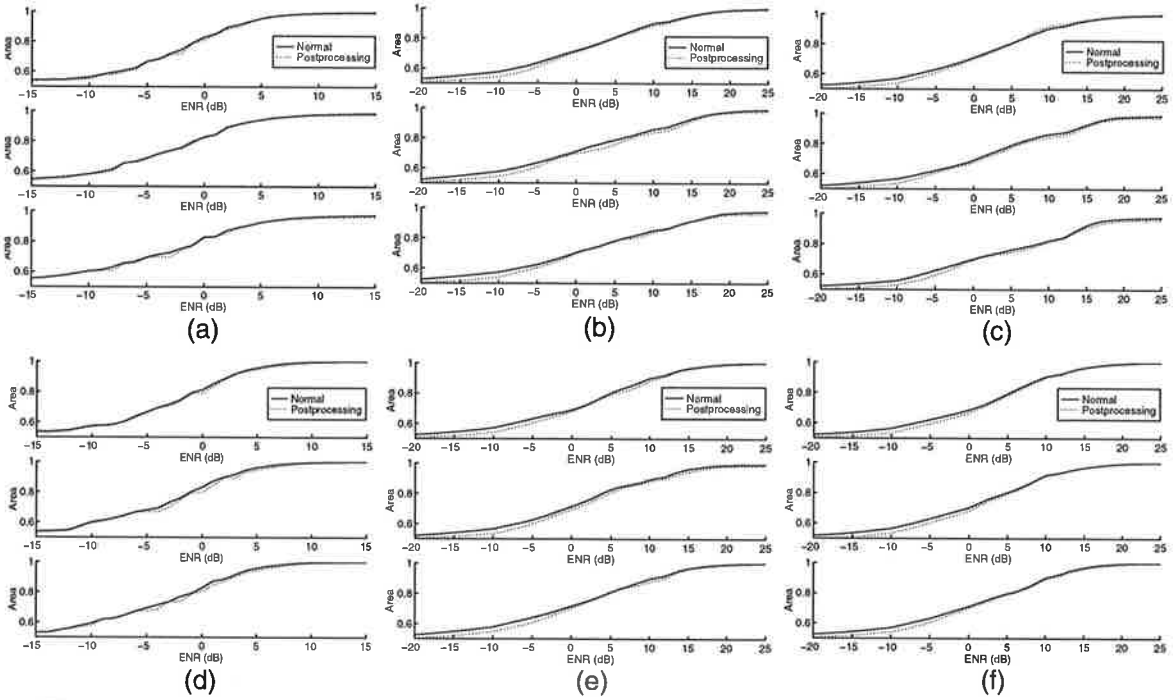


Figure C.17 Area with complementary output postprocessing for multiplicative, Gaussian, and uniform noise (from left column to right, respectively). The SICNN has, from top to bottom in each subplot, r = 5, 10, 15, and optimal decay factor. The weights are asymm. rectangular for top row and optimal for the bottom row.

C.5 Neighbourhood Processing

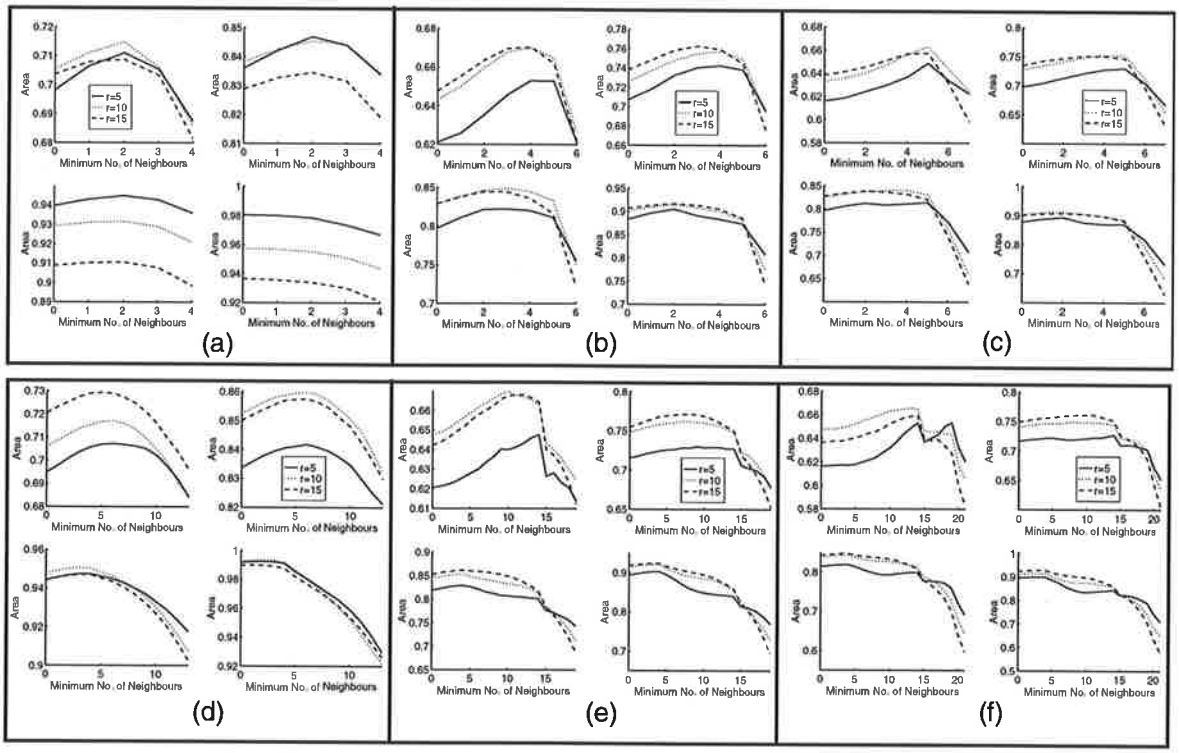


Figure C.18 Neighbourhood processing performance for multiplicative, Gaussian, and uniform noise (from left to right). Each subplot shows the area for ENR of -5, 0, 5 and 10 dB. The SICNN has symmetrical, **rectangular** weights, and optimal decay factor. Top row is for 3×3 and bottom row is for 5×5.

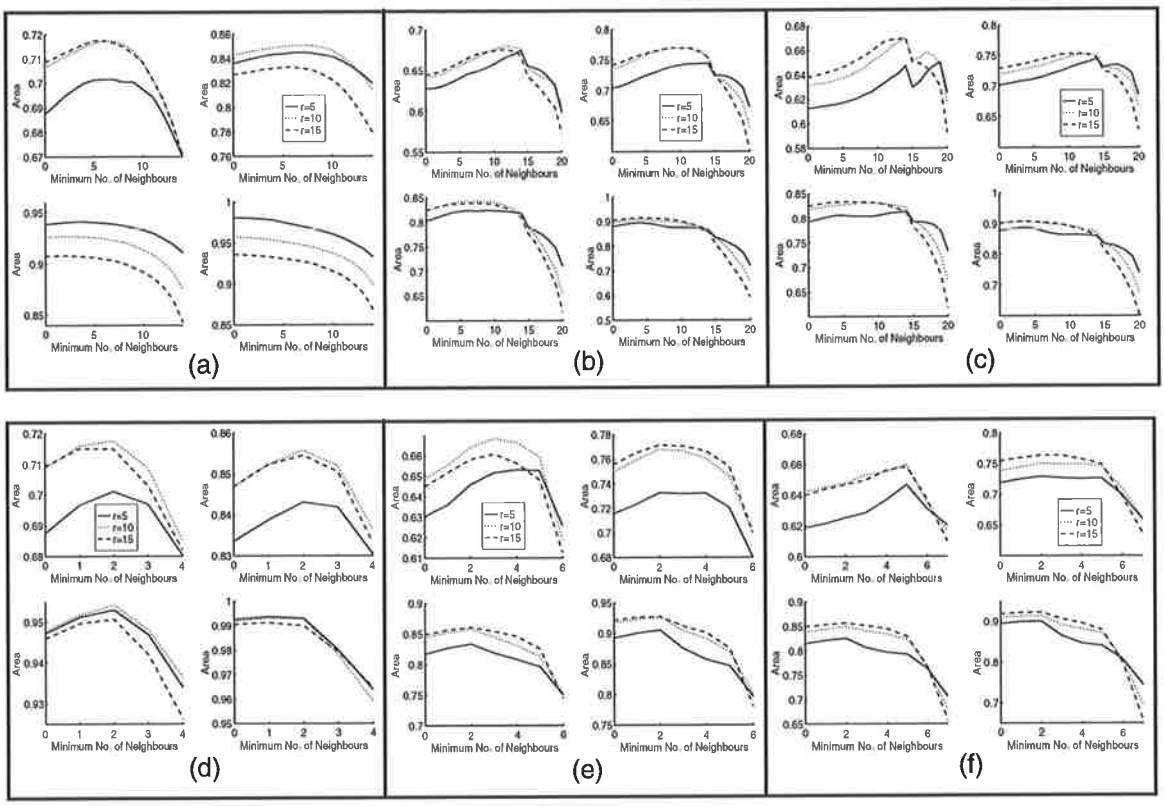


Figure C.19 Same as Figure C.18 but for the optimal weight distribution.

Appendix D Edge Detection Comparison Results

D.1 One Dimension

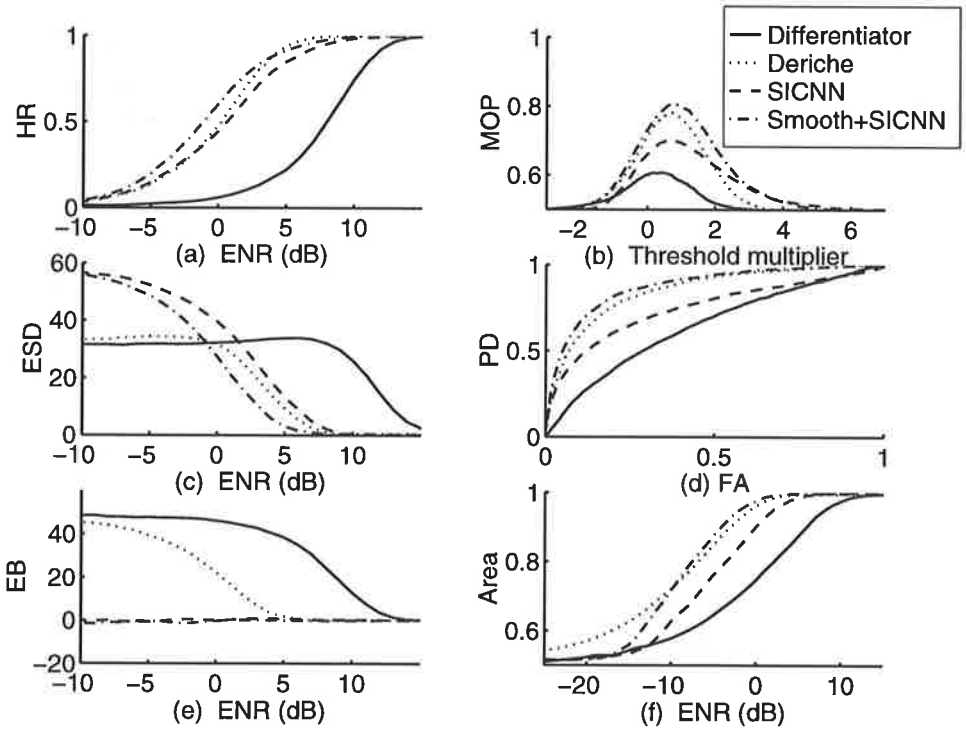


Figure D.1 Comparison of various edge detectors' performance for step edges with **multiplicative** noise. The SICNNs have asymmetrical rectangular weights, $r = 5$, and optimal decay factor. Subplots (b) and (d) are for $ENR = -5$ dB.

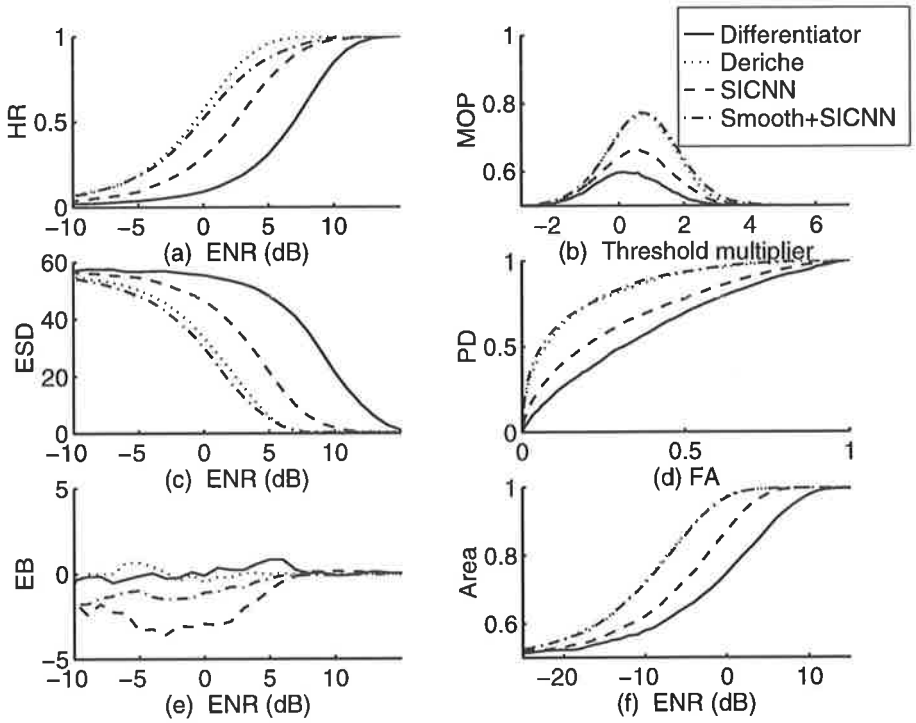


Figure D.2 As per Figure D.1 but for **Gaussian** noise.

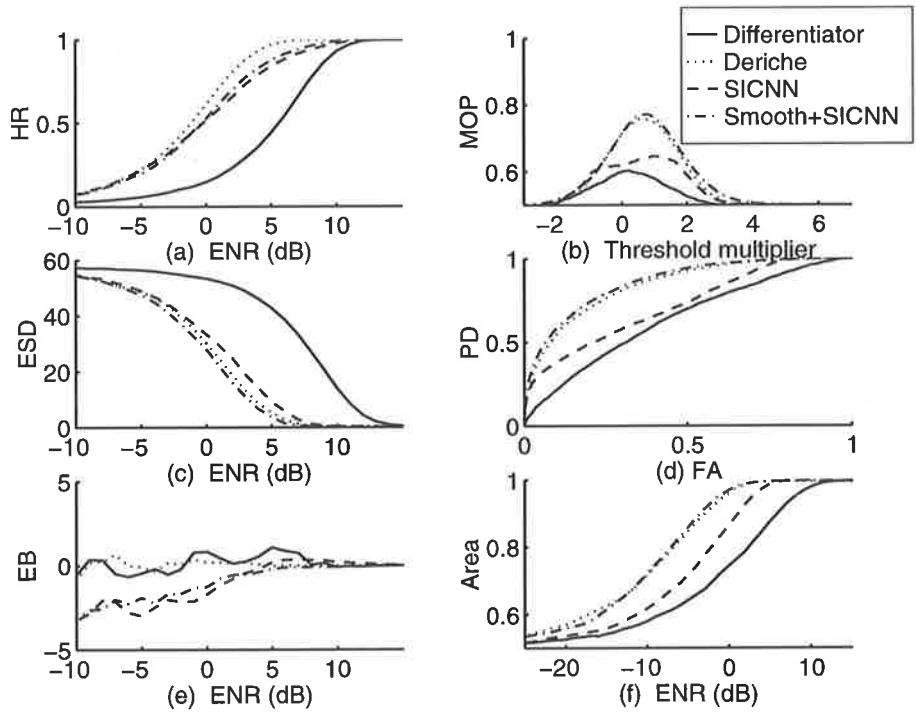


Figure D.3 As per Figure D.1 but for **uniform** noise.

D.2 Two Dimensions

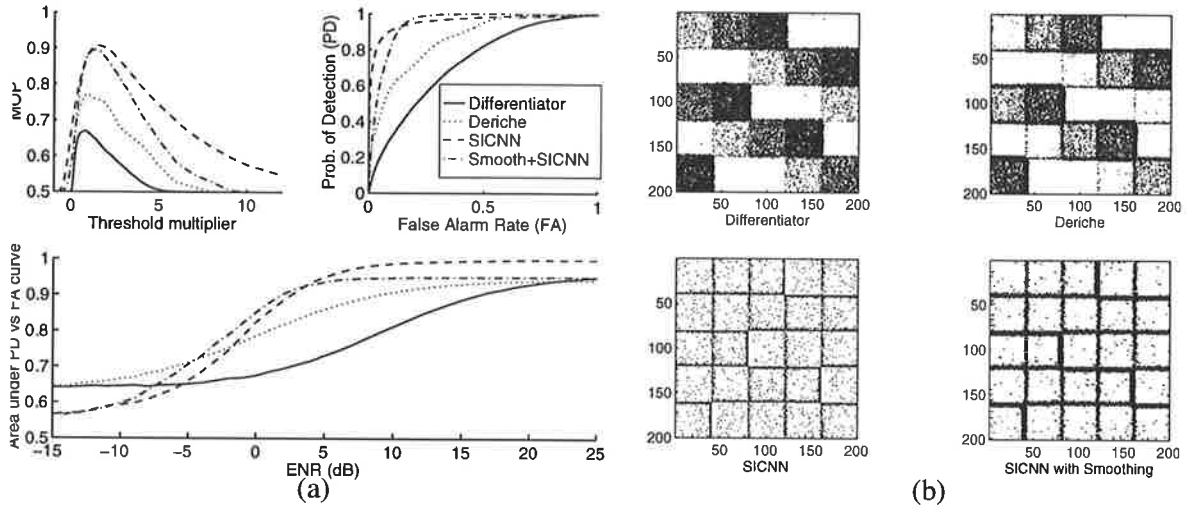


Figure D.4 (a) compares edge detectors' performance for **multiplicative** noise. Both SICNNs have asymmetrical rectangular weights, $r = 5$, and optimal decay factor. The MOP and PD vs. FA curves are for $ENR = 5$ dB. (b) shows the edge maps for $ENR = 5$ dB and a threshold that maximises the MOP in (a). The NSR values are (a) 12.76, (b) 8.57, (c) 2.23, and (d) 4.25.

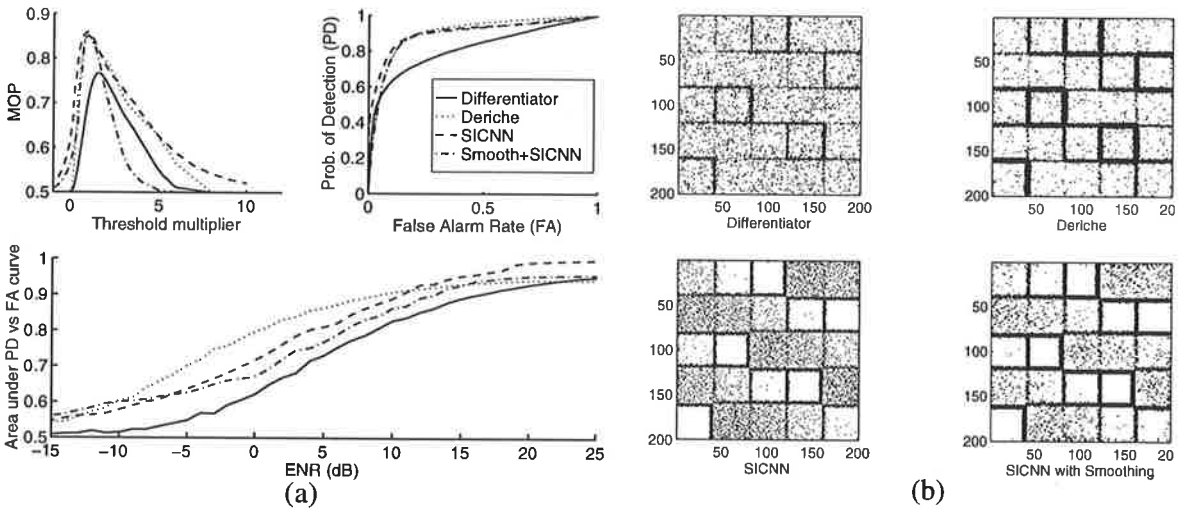


Figure D.5 As per Figure D.4 but for **Gaussian** noise. The MOP, PD vs. FA curves and the images in (b) are for an $ENR = 10$ dB. The NSR is (a) 5.37, (b) 4.59, (c) 4.28, and (d) 4.51.

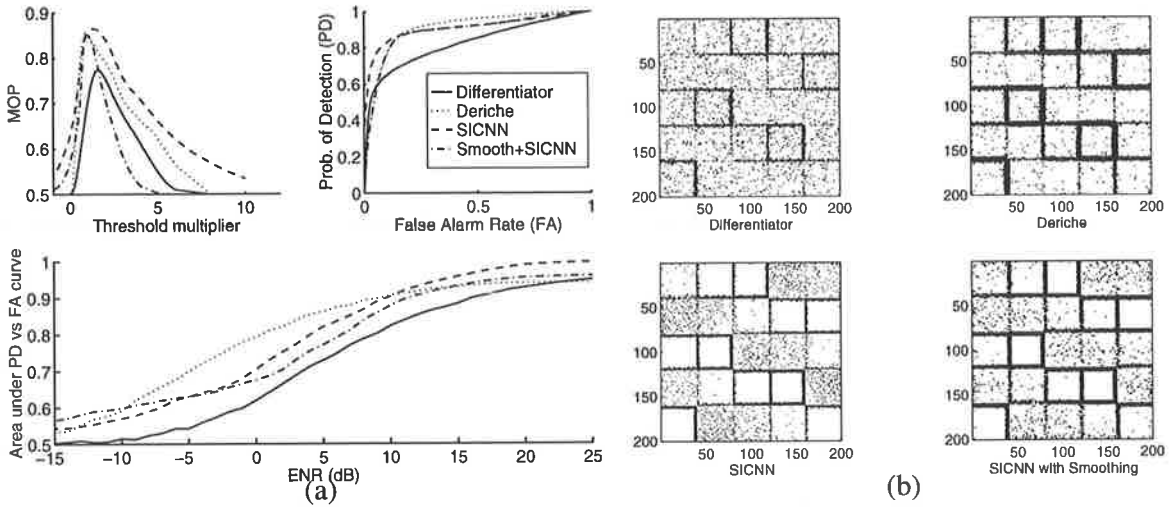


Figure D.6 As per Figure D.4 but for **uniform** noise. The MOP, PD vs. FA curves and the images in (b) are for an *ENR = 10* dB. The NSR is (a) 4.29, (b) 4.43, (c) 3.06, and (d) 4.33.

Appendix E Derivation of Optimal Enhancement Schemes

E.1 Derivation of Optimal Parameters

We showed in Section 9.4.2.1 how to choose the SICNN decay factor to maximise its output EEP for 1 iteration. Using the same optimization criterion, we now derive the optimal values for α of the GLE scheme (Guillon et al., 1996). Consider Figure E.1 which shows a 1-D step edge of mean intensity I_0 and contrast c.

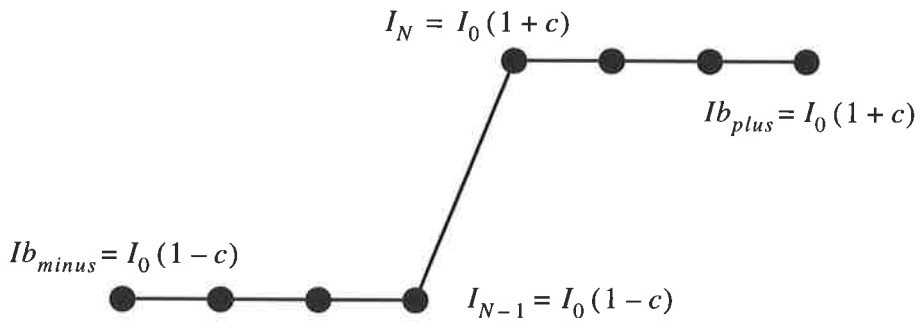


Figure E.1 A step edge of mean intensity I_o and contrast c.

Let M_j be a filter mask of total length L centred on the pixel j. Each coefficient m_j^k of this mask, where $k \in [j - L/2, j + L/2]$, indicates the level of confidence that each pixel in that mask belongs to the mask. Thus, $m_j^k \to 1$ for pixels with intensities similar to that of pixel j, and $m_j^k \to 0$ otherwise. So, we need to define a function for the mask which satisfies $M_j = \{m_j^k \in [0, 1] \,|\, k \in \left[j - \frac{N}{2}, j + \frac{N}{2}\right]\}$. A suitable function is (Guillon et al., 1996)

$$m_j^k = e^{\frac{-(I_k - I_j)^2}{2\sigma^2}}$$

where σ is the width of the Gaussian. The overall output of the GLE scheme at point j, which has intensity I_j, is

$$Y_j = I_j^{LP}\,(1+\alpha I_j^{HP})$$

where

$$I_j^{LP} = \sum_{k\in M_j} m_j^k I_k \Big/ \sum_{k\in M_j} m_j^k$$

$$I_j^{HP} = \sum_{k\in M_j} (m_j^k - \bar{m}_j)\, I_k$$

$$\bar{m}_j = \frac{1}{N}\sum_{k\in M_j} m_j^k$$

At the edge pixel N, we have

$$\begin{aligned} m_N^k &= \sum_{k\in M_N} e^{\frac{-(I_k - I(1+c))^2}{2\sigma^2}} \\ &= \sum_{k=1}^{r} e^{\frac{-(I_0(1-c) - I(1+c))^2}{2\sigma^2}} + \sum_{k=0}^{r} e^{\frac{-(I_0(1+c) - I(1+c))^2}{2\sigma^2}} \\ &= re^{\Psi} + (r+1), \end{aligned}$$

$$\bar{m}_N = \frac{1}{L}\sum_{k\in M_N} m_N^k = \frac{re^{\Psi} + (r+1)}{2r+1},$$

$$I_N^{LP} = \frac{\sum_{k\in M_N} m_N^k I_k}{\sum_{k\in M_N} m_N^k} = \frac{re^{\Psi} I_0(1-c) + (r+1) I_0(1+c)}{re^{\Psi} + (r+1)},$$

$$\begin{aligned} I_N^{HP} &= \sum_{k\in M_N} (m_N^k - \bar{m}_N)\, I_k \\ &= [re^{\Psi} I_0(1-c) + (r+1) I_0(1+c)] \\ &\quad - \left(\frac{re^{\Psi} + (r+1)}{2r+1}\right)(rI_0(1-c) + (r+1) I_0(1+c)), \end{aligned}$$

where $\psi = \left(\frac{-4c^2 I_0^2}{2\sigma^2}\right)$. The overall output is then

$$Y_N = I_N^{LP}(1+\alpha I_N^{HP}).$$

For the lower edge pixel at position $(N-1)$, we do the same procedure as above to obtain:

$$m_{N-1}^{k} = (r+1) + re^{\Psi},$$

$$I_{N-1}^{LP} = \frac{(r+1)I_0(1-c) + re^{\Psi}I_0(1+c)}{re^{\Psi} + (r+1)},$$

$$I_{N-1}^{HP} = ((r+1)I_0(1-c) + re^{\Psi}I_0(1+c)) - \left(\frac{re^{\Psi} + (r+1)}{2r+1}\right)(rI_0(1+c) + (r+1)I_0(1-c)),$$

$$\Rightarrow Y_{N-1} = I_{N-1}^{LP}(1 + \alpha I_{N-1}^{HP}).$$

The background intensities remain unaffected by the GLE algorithm as the local contrast is zero in these regions. Thus,

$$Yb_{plus} = Ib_{plus} = I_0(1+c) \ ,$$

$$Yb_{minus} = Ib_{minus} = I_0(1-c) \ .$$

The difference between the intensity of the edge pixels and the difference in the background intensities are, respectively:

$$\Delta y_p = Y_N - Y_{N-1} = \frac{-4I_0^2 cr(-1-2r-r^2+(1+r)(1+re^{\Psi})e^{\Psi})\alpha}{(1+2r)(1+r+re^{\Psi})} - \frac{2I_0 c(-1-3r-2r^2+re^{\Psi}(1+2r))}{(1+2r)(1+r+re^{\Psi})}$$

$$\Delta y = Yb_{plus} - Yb_{minus} = 2cI_0 \ .$$

Thus, the EEP is

$$EEP = 4\left(1 - \frac{\Delta y}{\Delta y_p}\right)\frac{\Delta y}{\Delta y_p}$$

$$= \frac{-8r[I_0(-1-2r-r^2+\delta_1 e^{\Psi}+r\delta_2 e^{2\Psi})\alpha + \delta_2 e^{\Psi}](\delta_1 + re^{\Psi})\delta_2}{[2I_0 r(-1-2r-r^2+\delta_1(1+re^{\Psi})e^{\Psi})\alpha - (1+3r+2r^2) + \delta_2 re^{\Psi}]^2}$$

where $\delta_1 = (1+r)$ and $\delta_2 = (1+2r)$. Solving for α when $EEP = 1$, which is the maximum 1-D step edge enhancement, we obtain the optimal value of α, i.e., the value of α which maximises the GLE's output EEP. This optimal value is:

$$\alpha_{opt} = \frac{1+3r+2r^2+3re^{\Psi}+6r^2e^{\Psi}}{-2rI_0(-1-2r-r^2+(r+1)e^{\Psi}+r(r+1)e^{\Psi})} \qquad \text{EQ (E.1)}$$

E.2 Optimal *n* for Contrast Transformation Functions

We showed in Section 9.4.2.1 and Section E.1 how to choose the decay factor to maximise the SICNN output EEP for 1 iteration and how to choose α to maximise the GLE's EEP, respectively. Using the same optimization criterion, we now derive the optimal values n for contrast enhancement edge enhancers such as $F(c) = \tanh(nc)$, $F(c) = \ln(1+nc)$, and $F(c) = c^n$, where c is the edge contrast.

The general algorithm for local contrast enhancement is given by Gordon & Rangayyan (1984). Consider the step edge shown in Figure E.1. Let the contrast transformation be $F(c_j)$, where c_j is the local contrast at pixel j, such that F satisfies: $c_j \in [0, 1]$: $F(c_j) > c_j$ and $F(c_j) \in [0, 1]$. We assume that the edge occurs at positions N and $N-1$ as shown in Figure E.1. The local mean, at these pixels, computed over a window of size $2r+1$ is

$$m_N = \frac{(r+1)I_0(1+c) + rI_0(1-c)}{2r+1} = \frac{I_0(1+2r+c)}{2r+1},$$

$$m_{N-1} = \frac{rI_0(1+c) + (r+1)I_0(1-c)}{2r+1} = \frac{I_0(1+2r-c)}{2r+1}.$$

The corresponding local contrasts at these pixels, as defined by Gordon and Rangayyan (1984), are

$$c_N = \frac{I_N - m_N}{I_N + m_N} = \frac{I_0(1+c) - m_N}{I_0(1+c) + m_N} = \frac{rc}{1+2r+rc+c},$$

$$c_{N-1} = \frac{m_{N-1} - I_{N-1}}{m_{N-1} + I_{N-1}} = \frac{m_{N-1} - I_0(1-c)}{m_{N-1} + I_0(1-c)} = \frac{rc}{-1-2r+rc+c}.$$

The intensity of these pixels after transforming the contrast is given by Gordon & Rangayyan (1984)

$$Y_N = \frac{m_N(1+F(c_N))}{1-F(c_N)},$$

$$Y_{N-1} = \frac{m_{N-1}(1+F(c_{N-1}))}{1-F(c_{N-1})}.$$

The input background intensity, $Ib_{plus} = I_0(1+c)$ and $Ib_{minus} = I_0(1-c)$ remain unchanged, as the contrast in the background regions is zero. Thus, $Yb_{plus} = Ib_{plus}$ and

$Yb_{minus} = Ib_{minus}$. The difference in the background intensities and the peak-to-peak difference in intensities are:

$$\Delta y = Yb_{plus} - Yb_{minus} = 2cI_0,$$

$$\Delta y_p = Y_N - Y_{N-1} = \frac{m_N(1+F(c_N))}{1-F(c_N)} - \frac{m_{N-1}(1+F(c_{N-1}))}{1-F(c_{N-1})}.$$

The EEP is then

$$EEP = 4\left(1-\frac{\Delta y_p}{\Delta y}\right)\frac{\Delta y_p}{\Delta y}$$

Solving $EEP = 1$, the maximum possible EEP for a step edge, we obtain the optimal value of n, i.e. the value of n which maximises the EEP. The above procedure can, in theory, be used to find the optimal value of n for any transformation $c_j' = F(c_j)$. We present the EEP and optimal n for a number of different transformations, below. It is almost impossible to determine the optimal value of n analytically without the use of a symbolic maths software.

We will see that all of the expressions for the optimal n depend only on the neighbourhood size r, and the input edge's overall contrast c. Thus, to speed up the execution time, the optimal value of n can be stored in a lookup table as both r and c vary.

Tanh Contrast Enhancement

Consider the *tanh* contrast transformation function, $F(c) = \tanh(nc)$. With this transformation, the EEP can be derived as

$$EEP = \frac{8\left(1-\frac{2cI_0}{\delta_1-\delta_2}\right)cI_0}{\delta_1+\delta_2}$$

$$\delta_1 = \frac{I_0(1+2r+c)(1+\delta_4)}{(1+2r)(1-\delta_4)}, \quad \delta_2 = \frac{I_0(1+2r-c)(1+\delta_3)}{(1+2r)(1-\delta_3)}$$

$$\delta_3 = \tanh\left(\frac{nrc}{-1-2r+rc+c}\right), \quad \delta_4 = \tanh\left(\frac{nrc}{1+2r+rc+c}\right).$$

The value of n which maximises the EEP, i.e., the value which gives an EEP of 1 is:

$$n_{opt} = \left(\ln\left(\frac{4c+8rc+e^{2Z^*}(1+2r-c)}{1+2r+c}\right)+2Z^*\right)\left(\frac{(rc+c)^2-(2r+1)^2}{rc^2(1+r)}\right), \qquad \text{EQ (E.2)}$$

where Z^* is the root of

$$2(-1-2r+rc+c)Z-(1+2r+rc+c)\delta_5 = 0$$

$$\delta_5 = \ln\left(\frac{4c(1+2r)+e^{2Z}(1+2r-c)}{1+2r+c}\right).$$

Logarithmic Contrast Enhancement

Consider now the *logarithmic transformation* given by $F(c) = \ln(1+nc)$, where n is a positive number. We add 1 to the argument to avoid the case of take the logarithm of 0. Using the method outlined above, the EEP of the output using this transformation is

$$EEP = \frac{-4(-2rc+\delta_1\delta_3+\delta_2\delta_4)c(1+2r)(1+\delta_3)(-1+\delta_2)}{[(1+2r)\delta_3+(1+2r)\delta_4+c(1+\delta_3\delta_4)]^2},$$

$$\delta_1 = 1+2r-2rc-c,\ \delta_2 = 1+2r+2rc+c,$$

$$\delta_3 = \ln\left(\frac{-1-2r+(1-n)rc+c}{-1-2r+rc+c}\right),\ \delta_4 = \ln\left(\frac{1+2r+rc(n+1)+c}{1+2r+rc+c}\right).$$

Solving EEP for n which give the maximum enhancement of 1, we obtain

$$n_{opt} = \frac{-1-2r-rc-c+(1+2r+rc+c)\delta_1}{rc} \qquad \text{EQ (E.3)}$$

$$\delta_1 = \exp\left(\frac{(-1-2r+4rc+2c)Z^*+4rc+c}{c(3+4r)Z^*+1+2r+4rc+2c}\right)$$

where Z^* is the root of

$$(-1-2r+rc+c)e^Z+(1+2r+rc+c)\delta_3-2c(1+c) = 0$$

$$\delta_3 = \exp\left(\frac{(-1-2r+4rc+2c)Z+4rc+c}{c(3+4r)Z+1+2r+4rc+2c}\right)$$

Contrast Power Enhancement

Consider now the *contrast power transformation* given by $F(c) = c^n$, where n is a positive number. Using the method outlined above, the EEP of the output using this transformation is

$$EEP = 8\left(1 - \frac{2cI_0}{\delta_1}\right)\frac{cI_0}{\delta_1}$$

where

$$\delta_1 = \frac{I_0(1+2r+c)\left(1+\left(\frac{rc}{1+2r+rc+c}\right)^n\right)}{(1+2r)\left(1-\left(\frac{rc}{1+2r+rc+c}\right)^n\right)} - \frac{I_0(1+2r-c)\ (1-\delta_2)}{(1+2r)\ (1+\delta_2)},$$

$$\delta_2 = \frac{-rc}{-1-2r+rc+c}.$$

Solving EEP for the value of n which give the maximum enhancement of 1, we obtain

$$n_{opt} = \frac{Z^*}{\delta_4} \qquad \text{EQ (E.4)}$$

where Z^* is the root of

$$(1+2r+4cr+2c)\,e^Z + c(3+4r)\,e^{\delta_6} + (1+2r-4rc-2c)\,\delta_5 - c(1+4r) = 0$$

$$\delta_4 = \ln\left(\frac{rc}{1+2r+rc+c}\right),\ \delta_5 = \exp\left(\frac{-Z\ln\left(\frac{-rc}{-1-2r+rc+c}\right)}{\delta_1}\right)$$

$$\delta_6 = \exp\left(Z\ln\left(\frac{-rc}{(-1-2r+rc+c)\ (1+2r+rc+c)}\right)\right)$$