Neural Networks as Self-Learning Image Features

A practical approach

Ordwin Notten

Catholic University of Nijmegen,
The Netherlands
University Hospital Nijmegen
Biophysics Laboratory
Department of Ophthalmology

Faculty of Mathematics and Informatics
Department of Computing Science
Experimental Informatics for Technical Applications

Master's Thesis, number **370**
April 1996

Preface

This thesis is the result of my graduation project to receive my Master's degree in Computer Science at the Catholic University of Nijmegen. This project which ran from September 1995 to April 1996, was a cooperation of the Department of Computing Science and the Biophysics Laboratory of the Department of Ophthalmology of the University Hospital Nijmegen.

The goal of this project was trying to develop a practical system that can learn its own image features to improve medical diagnostics and to find out if it can be used in practise. This goal was tackled by using a neural network strategy able to produce self learning image features, solving some practical problems and evaluating its practical use.

At this point I want to thank dr. ir. Han Thijssen, ir. Henkjan Huisman and dr. Theo Schouten, my supervisors at the Biophysics Laboratory and the Computer Science department respectively, for their support, advice and help. Furthermore, I like to thank the other people of the Biophysics Laboratory, my family and friends for their support and help.

Ordwin Notten
Nijmegen, April 1996

Contents

1. Introduction

At the Biophysics Laboratory of the Department of Ophthalmology at the University Hospital Nijmegen research is done in the fields of electrophysiology and ultrasound diagnostics. For the project: “Processing of echographic images” current research is directed to images with local lesions. These images are viewed by specialists to detect these abnormalities. However, a lesion is difficult to detect, therefore research is done to find new image features that enhance the diagnostics of detecting lesions. Maximizing information tends to using as many image features as possible. However, to prevent the peaking phenomenon[1] a classifier requires a low number of features. Therefore, feature reduction is often necessary. The problem that arises is which and how many image features have to be used for classification purposes. Reducing the number of features with a minimum of information loss means that every image feature has to contain as much information as possible with relation to the diagnostics. The remaining information could be explained as noise which is undesirable. Further, every image feature has to contain as much discriminating information as possible, compared to the other image features, otherwise the information contained in the sum of image features does not achieve its maximum. The “known” image features that meet these requirements are strongly problem dependent, sometimes difficult to find and hardly optimal. When a researcher has enough knowledge about a problem he could build his own image features. However, these would also be problem dependent. Using a self learning method that is capable to calculate every image feature by itself, then an optimum feature-set could be found for every problem.

Research has been done at the laboratory to implement this idea using a neural network strategy. The reason for choosing a neural network strategy is that it can learn any input-output relation (e.q. a feature) if it is large enough. In this research [Reijnen, 1994] a literature study was performed and several neural network strategies were compared. As a result of this research an operational strategy was found that was able to produce self learning image features. It did need tremendously large, unpracticable training sets, thus the goal attached to this thesis is trying to develop this strategy into a practical system.

This thesis gives a survey and the results of the consecutive steps that were performed during the development of this practical system.
In chapter 2 the background theory about ultrasound is explained. This chapter describes what ultrasound is and how to generate it. Besides that, it explains why and how ultrasound is being used in medical diagnostics. Finally, the backgrounds are given of representing data in ultrasonic images. Chapter 3 gives the background theory about a class of neural networks which is being used in this thesis: higher order neural networks(HONN). This chapter describes the history of higher order neural networks, the theory behind these networks and the advantage of a HONN. In chapter 4 the identified problems for this thesis are described. Chapter 5 describes an overview of the system. In chapter 6 a study is done for the HONN pre-processing. Chapter 7 describes the evaluation of the self learning image feature-system. In chapter 8 conclusions concerning the goal attached to this thesis and recommendations for further research are given.

---

[1] Given a finite size training set. The overall unconditional error rate of a classifier will first decrease when the number of features is increased. However, beyond a certain number of features, depending on the size of the dataset, the error rate will increase.

2. Ultrasound

In this chapter some background information is given on ultrasound. The first section gives an introduction to ultrasound, such as what it is and where to place it. In the rest of the sections some frequently used terms concerning this subject are explained. The second section describes the transducer, which is used to generate and receive sound-waves. In the third section the inhomogeneities in biological tissue are described. Knowledge about these inhomogeneities is important for the understanding of images that will be introduced in the fifth paragraph. The fourth section explains the beam that is generated by the transducer. Finally the fifth section describes some types of images that can be produced by the data received from the transducer.

2.1 Introduction to ultrasound

The science of sounds is called the science of acoustics. It comprises all mechanical waves of all possible frequencies and in all possible materials. A mechanical wave is a systematic disturbance of the condition of rest of a material that changes its position in space during a lapse in time.

Sound can be divided into three frequency areas.

1. Infrasonics.
   The sound with frequencies that are too low to be heard by humans.
2. Audiosonics.
   The sound that can be heard by humans.
3. Ultrasonics.
   The sound with frequencies that are too high to be heard by humans.

In medical diagnostics ultrasound with a frequency-area between 1 and 50 MHz is used. The ultrasound-wave is generated and received by a transducer.

2.2 Transducer

Most transducers contain one or more pieces of material that exhibits the piëzo-electrical effect. This means that when a voltage is applied to that material a sound-wave will be generated caused by the volume change in that material. The opposite is also possible, when a sound-wave is entering a transducer, the volume of the material changes and a voltage is generated. So, a transducer can act as a sender and as a receiver of ultrasound.

In medical diagnostics, ultrasound is used in biological tissue that has a non-homogeneous composition. When sending ultrasound through a non-homogeneous medium backscattered echoes will be generated. If this ultrasound is emitted as a short pulse, and shortly afterwards echo-signals are received by the same transducer then we speak of a process known as the pulse-echo method.

2.3 Biological tissue

A biological tissue has a non-homogeneous composition. An inhomogeneity can be defined as:
'A particle with a different acoustical impedance than the surrounding tissue.'. These inhomogeneities vibrate under the influence of incident sound-waves. There are two kinds of inhomogeneities, namely reflectors and scatterers. These will be explained below.

Reflectors

Inhomogeneities that are much larger than the ultrasound wavelength are called reflectors. Examples for this kind of inhomogeneities are blood vessels, organs, tumors, etc. In figure 2.1 an example of a reflector is given.

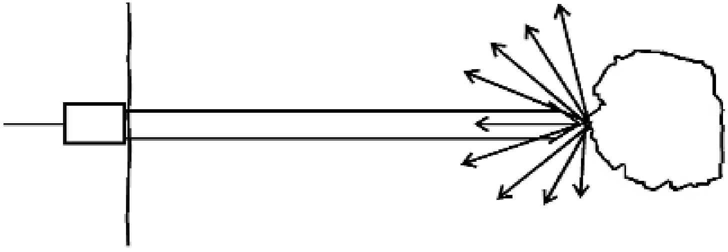


Figure 2.1 Reflector

Scatterers

Inhomogeneities that are small compared to the ultrasound wavelength are called scatterers. If ultrasound hits a scatterer, the ultrasonic wave is scattered in all directions. The amount of scattering is dependent on the type of tissues, such as the size and density of the scatterers, and the frequency of the ultrasound. Some of these scatterer echoes[2] are received by the transducer that will display them as inhomogeneities in the ultrasonic-image. These scatterer echoes are also known as the backscattering echoes. In figure 2.2 [Thijssen et al., 1990] an example of a scatterer is given. The part that is scattered back to the transducer is used for imaging.

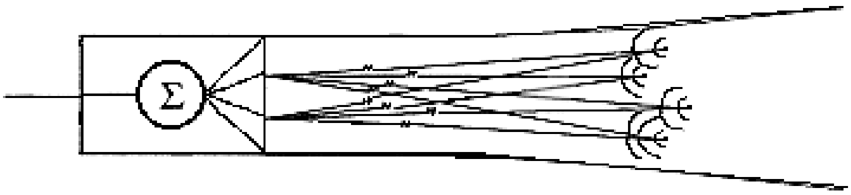


Figure 2.2 Scattering by small structures yields small omni-directional wave-fronts.

---

[2] The scatterer echoes that have a degree of 180° with respect to the direction in which the beam is transmitted.

Absorption

Part of the energy of the ultrasonic wave that travels through the tissue is transformed into heat. As mentioned in [Thijssen et al., 1990] absorption arises during the rapid pressure changes in the sound-waves. The more pressure changes in a time span (the higher the frequency) the higher the absorption. The higher the absorption the less energy there is left for reflection. The absorption is governed by the relaxation processes in biological molecules. Therefore the absorption is dependent on the type of tissue.

2.4 Ultrasonic beam shape

The intensity of ultrasound transmitted by a transducer is not the same in every direction. The transducer transmits it into a specific direction. The width of this beam depends on the distance to the transducer. A distinction is made between the Fresnel-zone and the Fraunhofer-zone.
A picture of this beam is given in figure 2.3 [Reijnen, 1994].

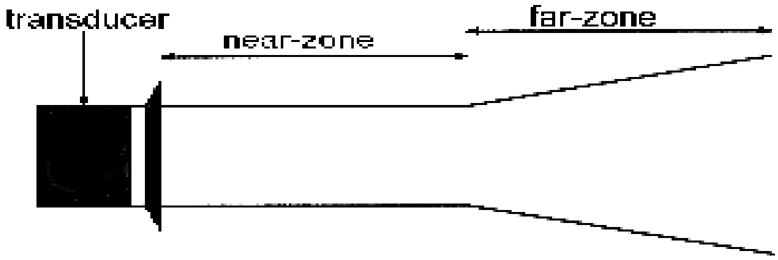


Figure 2.3 Ultrasonic beamshape

Fresnel-zone

The Fresnel-zone, also called the near-zone, is the sound field at a short distance from the transducer. The length of this near-zone is dependent on the size of the active surface of the transducer and the used wave-length. The width of the sound beam is proportional to the width of the transducer. Sound vibrations that arrive at a particular point in the Fresnel-zone can amplify or weaken each other [Thijssen et al., 1990]. This is possible because the vibrations, from the different parts of the transducer, arrive at a particular point in the Fresnel-zone at different times (due to the wavelength and the difference in distance). This is also known as interference.

Fraunhofer-zone

The Fraunhofer-zone, also called the far-zone, is the sound field at a large distance from the transducer. This beam diverges slowly. The transition from the near-zone to the far-zone is reached on the point where the difference in wavelength from the side of the transducer compared with the center of the transducer is exactly ½ λ(wavelength). This point can be calculated by $d^2 / (4*\lambda)$.

2.5 Ultrasonic images

There are different ways of imaging ultrasound. Two types will be explained.

A-mode images

An A-mode image is a one-dimensional representation of the amplitude of the echoes as a function of the depth. It is the simplest representation of RF-signals.

Before such an image is constructed, the RF-signal has undergone two operations:

1. Amplification.
   This is a time dependent compensation, also known as TGC (Time Gain Compensation), which can be in a non-linear way. It is used to compensate for the total loss of energy per unit length from the ultrasound wave as it is propagated through tissue. With this method echo signals received early in time are amplified slightly and echo signals received later in time are amplified more. In this way all the echo signals have more or less the same strength which is needed to anticipate saturation and loss of echo signals in the echogram.
2. Envelop detection.
   In this operation the signal is demodulated and filtered.

The result of these operations is visualized in figure 2.4 [Thijssen et al., 1991] which is related to Figure 2.2.

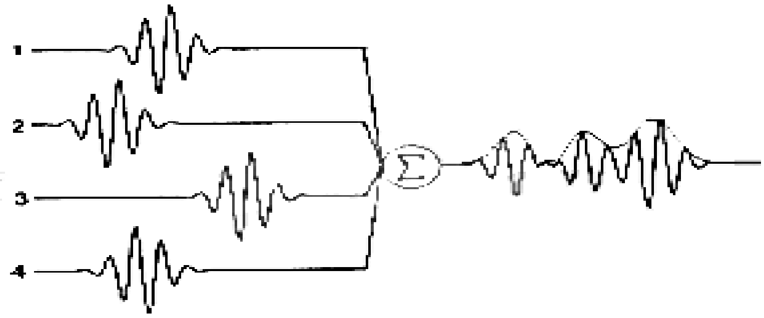


Figure 2.4 Construction of an A-mode image. The phase-sensitive sum of the backscattered echoes of the four scatterers is the RF-signal. The dotted line superimposed on the RF-signal is the envelop signal.

B-mode images

A B-mode image is a successive depiction of a series of A-mode images, whereby the amplitude is represented by brightness. By adding this second dimension the anatomical information of the image increases.

B-mode images display, first of all, the position and size of organs. In most cases there is a transition of acoustic properties between the different tissues. Due to these transitions parts of the ultrasound will be reflected. When a structure, for instance a liver, has enough transitions surrounding the whole structure then the structure can be seen in a B-mode image. An example of a B-mode image is given in figure 2.5.

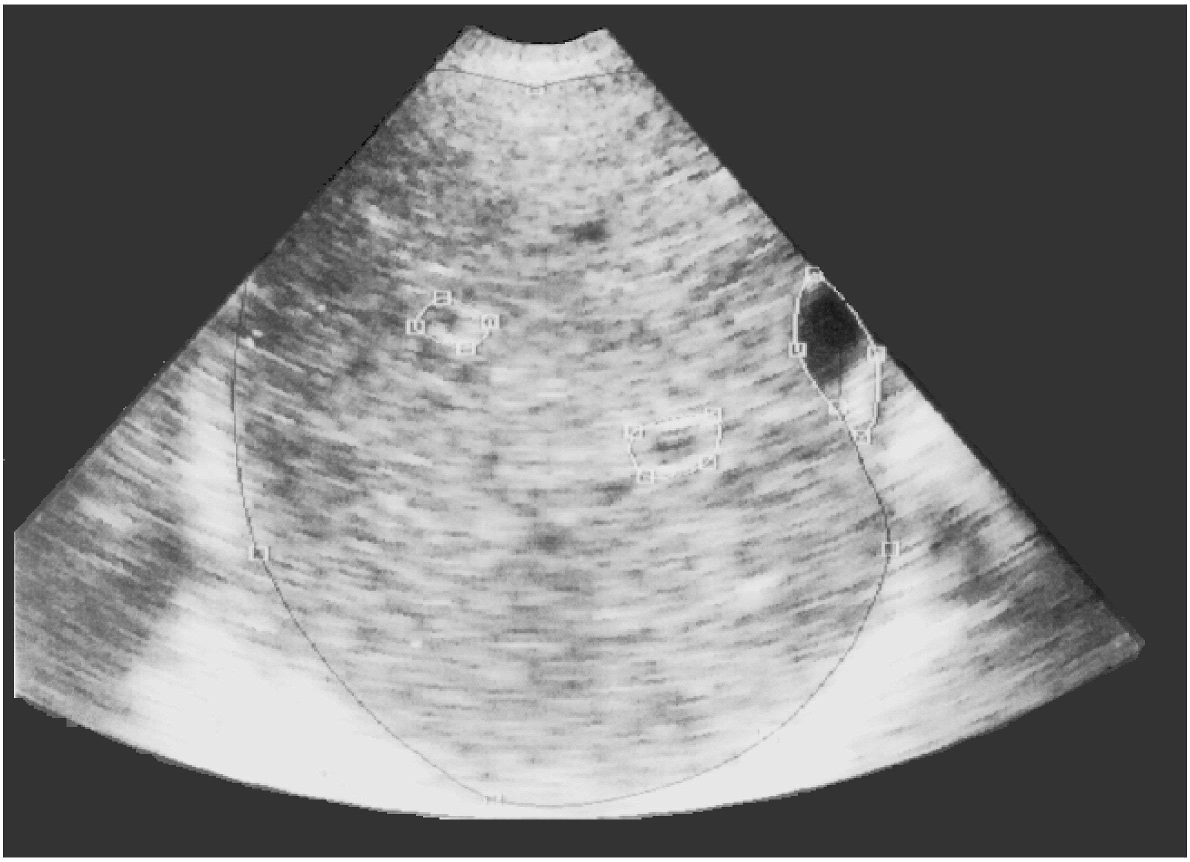


Figure 2.5 B-mode image

In this image you see the liver, surrounded by a solid line, and some blood-vessels in the small ellipses.

2.6 Speckle

Tissues that are expected to be homogeneous are covered with some kind of texture, which is called speckle, resulting from the scattering in that tissue, and not noise. The speckle pattern in an image depends on the characteristics of the tissue scanned and the imaging system [Oosterveld, 1984]. This pattern does not need to have a one-to-one correspondence with echo-producing targets. As an example, consider figure 2.2: a transducer insonating a homogeneous medium that contains four point-like scatterers. These scatterers yield small omni-directional waves that arrive at slightly different times after the transmission of the ultrasound pulse at the transducer. As shown in figure 2.4 the four rf echoes will build an interference pattern, because the depth differences of the scatterers are smaller than the pulse length. The dashed line is the demodulated rf-signal that contains three peaks. Speckle can be described by using first- and second-order statistics [Wagner et al., 1983]. First-order statistics include the mean amplitude and the signal-to-noise ratio. The second-order statistics include the auto-covariance in both axial and lateral directions.

3. Higher Order Neural Networks (HONN) as SLFs

In this chapter some background information on higher order neural networks is given. The first section gives an introduction to higher order neural networks. In the second section the theory behind neural networks and the HONN is explained. The third section explains the combination of HONN and the invariances. In the last section some interconnection techniques, to solve the interconnection problem for higher order neural networks, are described.

3.1 Introduction to Higher Order Neural Networks

Since World War II, scientists have tried to develop techniques that allow computers to behave more like human beings. As mentioned in [Doolen et al., 1986] the research effort that usually goes by the fanciful name of "Artificial Intelligence", typically involves abstract problem solving, decision making, planning, machine learning and natural language understanding.

In the 1960's research was done into neural networks. This research concentrated on determining the usefulness of the perceptron for non-linear discrimination of data into classes representing the objects to be identified. It was concluded that a single-layer, first-order network was limited to linear discrimination and that either a multi-layer, first-order network or a single-layer, higher-order network was necessary for non-linear discrimination. At that time no satisfactory ways of training multi-layer, first-order networks were at hand. Higher-order neural networks were rejected as impractical due to the combinatoric explosion of higher-order terms.

Research from a few years ago has shown that multi-layer, first-order networks using the backpropagation algorithm for training are effective for distortion invariant pattern recognition. However, backpropagation trained networks require a large number of training passes and an exhaustive training set, but are still not achieving a 100 % recognition accuracy. Research on HONNs [Giles et al.,1987], [Giles et al., 1988], [Reid et al., 1989b] and [Linhart et al., 1992] has shown that the problem of combinatoric explosion of interconnections can be overcome. This can be done by building invariances into the architecture of the network using information about the relationships expected between the input nodes. The number of interconnections is usually [Giles et al., 1988] reduced by orders of powers of N, where N is the size of the input.

3.2 Theory behind HONNs

Neural networks are being used more and more. This is due to the learning properties of a neural network, by which it can be tuned to specific problem area's.

A neural network of the first order, abbreviated as FONN, can be formulated as:

$$y_i = \Theta\left(\sum_j w_{ij} x_j \right)$$

where :

Θ is a translation function that can be non-linear (i.e. a threshold-function).

x_j input node j

w_{ij} matrix with the weight of each input

y_i output of node i

This ANN does not take advantage of any known relationships between the input nodes, so translation, scale and rotation invariance are not present. Multi-layer, first order networks and higher-order neural networks are known that they can learn invariances. Multi-layer, first order networks will not be mentioned further on due to their large number of training passes and not full recognition accuracy, as mentioned above.

Following [Reid et al., 1989a] and [Reid et al., 1989b] the output of nodes in a general higher order network is given by:

$$y_i = \Theta\left(\sum_j w_{ij} x_j + \sum_j \sum_k w_{ijk} x_j x_k + \sum_j \sum_k \sum_l w_{ijkl} x_j x_k x_l + \cdots \right)$$

A direct implementation (see figure 3.1) of a HONN [Huisman et al., 1995] is still impractical, because of a combinatoric explosion of higher order products. Incorporating invariances leads to translation invariant higher order transformations (see figure 3.2).

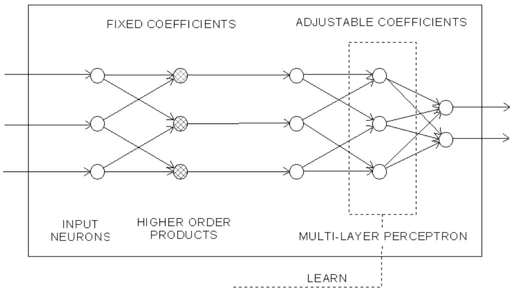


Figure 3.1 Theory HONN

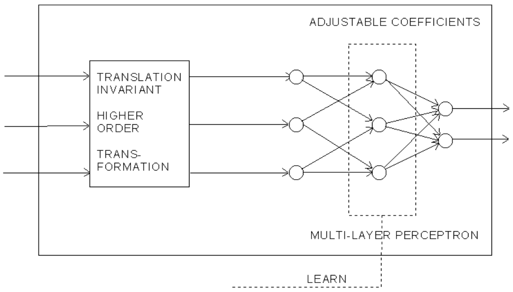


Figure 3.2 Practical implementation

Higher-order neural networks can perform the non-linear discrimination required for pattern recognition invariant to scale, translation and rotation. As mentioned above, the combinatoric explosion of interconnection problems with HONNs can be overcome by building invariances into the network architecture. In general translation invariance requires that:

$$w_{ijk} = w_{i(j-k)}$$

i.e.: the connections for equally spaced input pairs are all set equal.

A second-order neural network will be simultaneously invariant to scale and translation if the weights are set according to the function [Giles et al., 1988]:

$$W(i,j,k) = W\left(i, \left(y_k - y_j\right) \middle/ \left(x_k - x_j\right)\right)$$

This implies that W_{ijk} is set equal to $W_{ij'k'}$ if the line drawn in figure 3.3 between j and k equals the line between j' and k'.

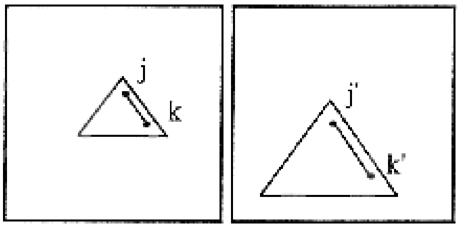


Figure 3.3 Translation and scale invariance

Rotational invariance can be included by using a third-order neural network, where the output is given by the function:

$$y_i = \Theta\left(\sum_j \sum_k \sum_l w_{ijkl} x_j x_k x_l\right)$$

Figure 3.4 shows all three invariances. As stated by [Reid et al., 1989a] and [Reid et al., 1989b] invariances to all three distortions can be achieved with a third-order network having an interconnection function of the form:

$$w_{ijkl} = w_{i\alpha\beta\gamma} = w_{i\gamma\alpha\beta} = w_{i\beta\gamma\alpha}$$

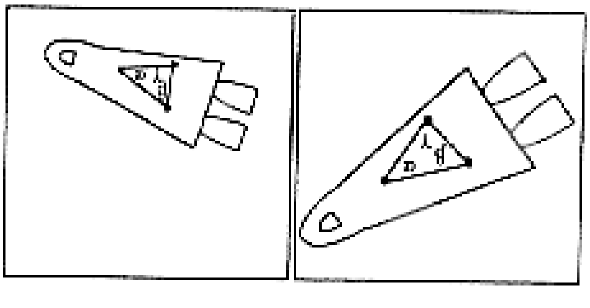


Figure 3.4 Translation, scale and rotation invariance are achieved by setting all third order weights equal for sets of inputs j, k and l that form similar triangles.

3.3 HONNs and invariance

As stated in [Reid et al., 1989a] HONNs are thus well suited for invariant pattern recognition because feature extraction is functionally built into the architecture. The invariances achieved require no learning to produce and apply to any input pattern learned by the network. Further [Reid et al., 1989b], a HONN can perform non-linear discrimination using only a single layer so that a simple perceptron learning rule can be used, leading to rapid convergence.

Recent research [Reijnen, 1994] has shown that HONNs can also be used for image processing as self-learning image features. As stated in [Huisman et al., 1995] a properly chosen HONN implementation operates as well as, or even better than, a properly chosen set of fixed features. As an example, translation invariance can be built into the second-order neural network with 4 nodes and 1 output node shown in figure 3.5. Assume that the input patterns (1010) and (0101) are to be identified as the same object. If $W_{i13} = W_{i24}$ then Y_i is the same for both inputs. Combinations of invariances can similarly be achieved.

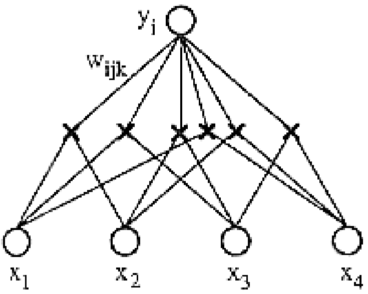


Figure 3.5 Second order neural network

According to [Schmidt et al, 1993] HONNs are a form of pre-processing for neural networks with the standard backpropagation that use geometrically motivated non-linear combinations of scene pixels to achieve invariant pattern recognition feature spaces.

They show that it is possible to directly incorporate geometric invariance properties such as translation and rotation invariance into the HONN if the HONN pixel combinations are properly chosen.

3.4 Non-fully interconnected techniques

Building invariances into the network, to overcome the problem of combinatoric explosion of interconnections in HONNs as mentioned above, greatly reduces the number of independent weights that must be learned. Although this also reduces the required storage space, some data still has to be stored. In a third-order neural network each triplet (for scale, translation and rotation invariance) of inputs has to be associated to a set of included angles. The most severe limitation that arises is that as the size of the input field increases, the number of triplet combinations that can be formed also increases. In a NxN pixel input field, three pixels can be combined in $(N^2)^3$ ways. Thus, for a 9x9 pixel input field, the number of combinations is 81^3 or 85,320. By increasing the resolution to 128x128 pixels, the number of combinations increases to $(128^2)^3$ or 7.3×10^{11}, a number too large to store on most machines.

To solve this interconnection problem, [Reid et al, 1990] have been evaluating various methods of connecting only a subset of input pixel triplets to the output node. They have evaluated four techniques, namely local, sampled, probabilistic and regional connectivity. These will be explained below.

Local connectivity

The assumption behind local connectivity is that local connections are adequate for distinguishing between objects and that distant information will be represented by some combination of local information. Thus, a connection for a triplet of pixels is formed only if all three pixels are within some bounded number of pixels away from each other.

Sampled connectivity

The assumption behind this technique is that only a fraction of all possible input pixel triplets are connected to the output. This fraction is not a function of the distance between the pixels, but a predetermined subset of interconnections.

Probabilistic connectivity

This method tries to improve on local connectivity, the images are sampled using a technique in which the probability of connection varies with distance.

Regional connectivity

This method combines the advantages of the other approaches while eliminating their disadvantages. Here, triplets of pixels are connected to the output only if the distance between all of the pixels falls within a set of pre-selected regions. These regions are chosen so as to sample the space sufficiently for accurate discrimination. Thus, triplets are formed from pixels that are local, distant and in-between. Because the connections are deterministic, it is guaranteed that the same information will be sampled from a test image, which may be rotated or translated, as from a training image. Besides the advantages of this technique there also is a disadvantage, namely that some experimentation’s are needed to determine an optimal set of regions.

4. Problem definition

In this chapter the problems contained in this thesis will be identified and solutions, to solve these problems, will be searched. The goal attached to this thesis is trying to develop a neural network strategy that is able to produce self learning image features into a system. To achieve this goal several questions concerning the problems have to be answered. Each of these questions will be discussed separately.

1. How to construct a good pre-processing unit? And which theories can best be used for it.
   The reason to use a pre-processing unit in the HONN is to reduce the input for the neural network to speedup the training process of the neural network. By reducing the dimensionality of the input for the neural network the number of needed training examples can also be reduced. Although reduction means data loss it does not imply that it is important data, but to make sure the following questions have to be answered: Which parts of the input may or may not be reduced? How to determine a pre-processing unit? How to verify that it is a good pre-processing unit? What is the maximum allowable reduction factor? To answer these questions, a study concerning this subject has to be done.

2. What size of window is used to focus on the image?
   Although the size of the window depends on the lesion chosen to be detected, it may not be too small or too large. If the size of the window is too small then a good resolution would be obtained, but a worse statistic. Making the window too big would mean a good statistic, but a worse resolution. The solution is to find a window size or a range of window sizes that have a good statistic and a good or reasonable resolution.

3. How long does a neural network have to learn?
   If a neural network is not trained long enough it is not able to classify correctly. On the other hand the possibility exists that the neural network is trained too long, so that it will not be able to classify new data: i.e. overlearning occurs. To determine how long a neural network has to learn, the following questions arise: How many training examples are statistically needed to learn a neural network to distinguish between the possibilities of a particular problem? How can be determined that the neural network has learned enough, and not too less or too much? A solution to test the trained neural network is to use an independent validation set to find out how good it classifies it.

4. Which labels will be assigned?
   To train a neural network to distinguish between different image regions, several example windows from these regions are required. Each window in the training set should have a label corresponding to the region to which it belongs. A question that arises: How to determine the label of a window? A solution for this is to use a function that estimates the label from an accompanying mask image.

5. What size of network will be taken? Meaning number of hidden layers etc.

When using a small neural network, for instance without a hidden layer, then the neural network can easily, depending on the dimensionality of the input, determine its weights. Using a larger neural network it becomes harder for the neural network to determine its weights, because a larger set of training examples are needed. However, the large neural network can learn more from the training set. A solution to take advantage of a larger, with for instance 1 or 2 hidden layers, neural network is to use a low dimensionality of input for the neural network.

6. How to incorporate the SLF (Self-Learning Image Features) into a practical system?
   Developing a practical system that can produce and use self learning image features needs a neural network. To let a neural network determine its own image features, the neural network has to learn to detect such image features. Therefore it has to learn the characteristics of those features. Questions that arise: How to develop such a system? What are the requirements of the system? Which programming language will be used? Are neural networks available in that programming language? And on which hardware will the system be developed? Answers for these questions will be given in chapter 5.

Not every question can be answered that easily, therefore the necessary studies have to be done, but first a design of the system will be constructed.

5. SLF System development

In this chapter the system for the image processing and analysis will be described. The first section will list the system requirements. In the second section a possible system design will be shown according to the system requirements.

5.1 Requirements

Before a system design can be developed and accepted the requirements have to be clear.

- All the different system-parts have to be separate units or modules.

- The system has to be developed in a modular way without having effect on the operation of the whole system. This means that units can easily be replaced by other units, for example replacing the pre-processing unit with another pre-processing unit.

- The system needs two kinds of batch-modes, one to let the system learn and one to let the system evaluate. In the learn mode the system will receive training images and their corresponding mask images to train the neural network. During the evaluation mode the neural network will evaluate the input images using its learned initialization.

- For test-facilities it is further desirable that all the units can be executed separately. For this purpose, the system should have a separate interface-window or control-unit where the user can execute them by prompt.

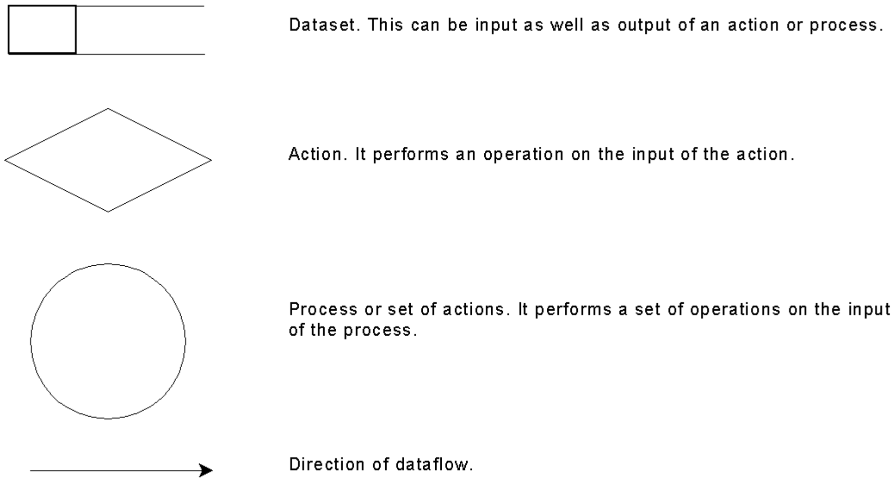
- The system will work with windows, or parts of the image, to perform the operations on.

- For image analysis several functions should be possible to apply to the images in the image processing part.

The system will be developed on a SUN workstation using the programming language MATLAB, a high-performance numeric computation and visualization programming language. Two reasons why MATLAB is chosen as programming language:
1. MATLAB is a modular based programming language with an experimental environment. Therefore it is easy to perform tests on separate units.
2. The programming language and knowledge about MATLAB are available in the laboratory.

5.2 System design

In this section a possible system design, according to the system requirements mentioned in section 5.1, will be given. Flowcharts and some explanation will be used to describe (part of) the system. Before a possible system design is presented, an explanation of the used symbols is given (see figure 5.1).



Dataset. This can be input as well as output of an action or process.

Action. It performs an operation on the input of the action.

Process or set of actions. It performs a set of operations on the input of the process.

Direction of dataflow.

Figure 5.1 Symbol explanation

Now that the symbols, used in the flowcharts, are explained the flowcharts that describe the system can be presented. First an overview and description of the total system will be given. After that, the flowcharts of each process are presented.

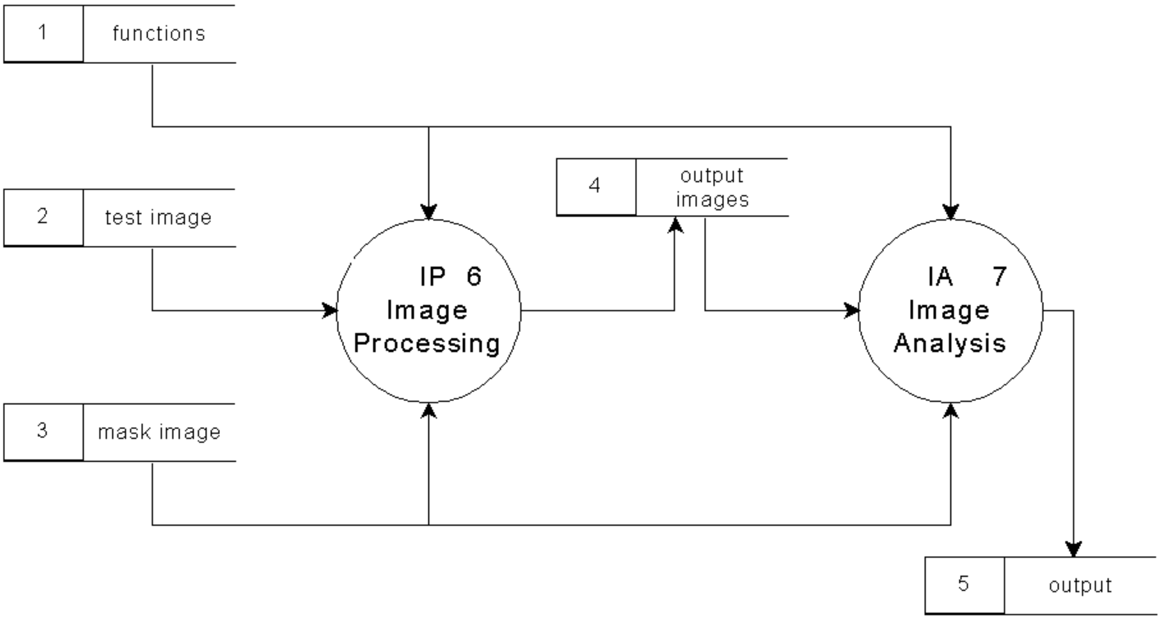


Figure 5.2 System overview

In figure 5.2 the overview of the total system is given. This flowchart contains two processes; the ‘Image Processing’-process (6) and the ‘Image Analysis’-process (7). The ‘Image Processing’-process (6) processes a test image according to the accompanying functions and delivers an output image for every function. The ‘Image Analysis’-process (7) analyses the classification of each image that results in an output. Each of these processes will be described in detail further on. The mask image of a test image is in the ‘Image Processing’-process (6) being used to train a neural network. In the ‘Image Analysis’-process (7) this mask image is being used to analyze the classification of the test image by the neural network.

To explain the ‘Image Processing’-process (6) further the flowchart describing this process will be used. The flowchart of this process is shown in figure 5.3.

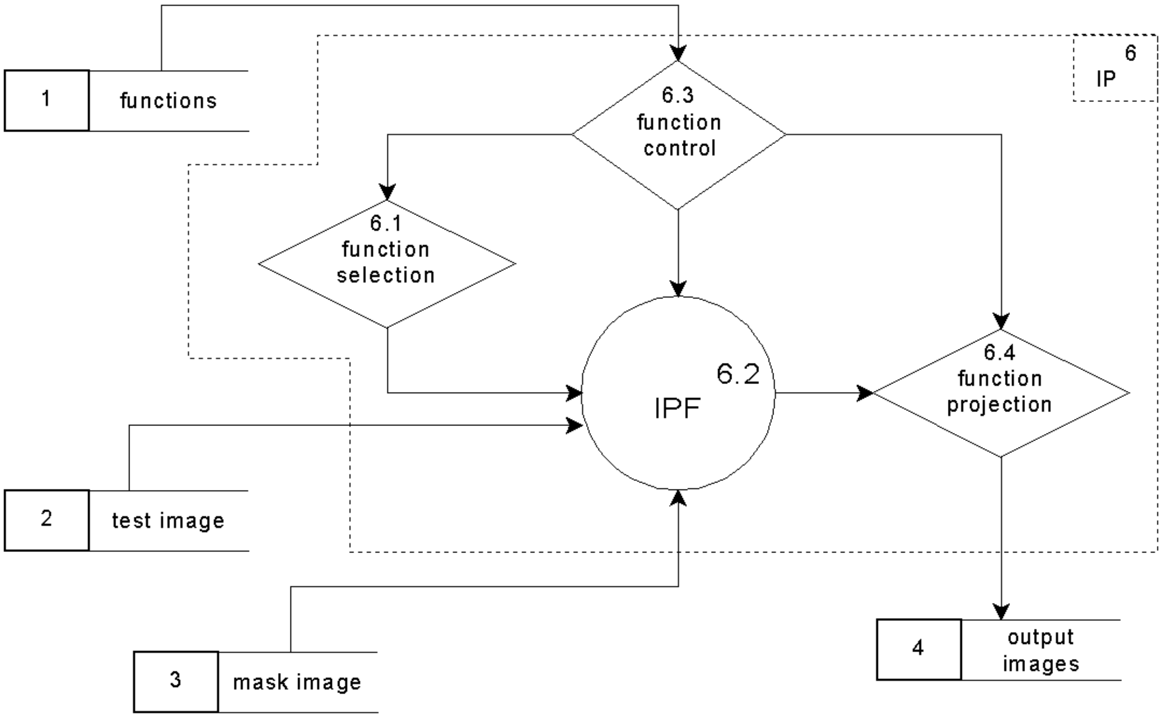


Figure 5.3 Image Processing overview

The input and output of the dashed box are the same as the input and output of the 'Image Processing'-process (6) in figure 5.2 and will therefore not be mentioned here. In this flowchart the accompanying functions, to be used on the test image, are split up, meaning that for every function the 'IPF'-process (6.2) is called to determine the output image. IPF stands for Image Processing on Function level. The function control (6.3) controls the set of functions out of which a function is selected by the function selection (6.1). This function selector selects a, not yet used, function to be used on the test image, and the function projection (6.4) connects the output of the 'IPF'-process (6.2) to the used function. From the function control (6.3) there is also an arrow going directly to the 'IPF'-process (6.2). This means that inside this process there are functions being controlled by the function control (6.3), but this will be explained further on.

To explain the 'IPF'-process (6.2) further the flowchart shown in figure 5.4 will be described.

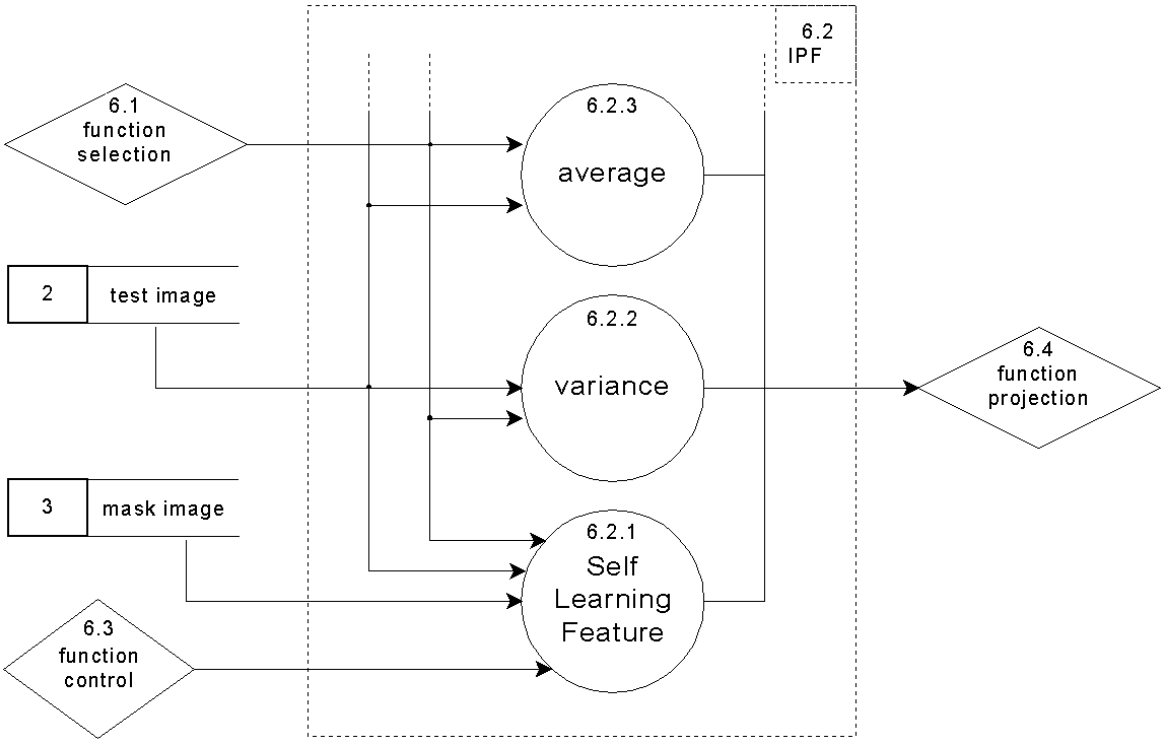


Figure 5.4 IPF overview

As shown in the upper right corner of the dashed box in figure 5.4 this box represents the ‘IPF’-process (6.2). In this figure some of the lines going to and coming from the processes have a dashed line, this means that only a subset of all processes have been put in this flowchart. The function selection (6.1) has one output-line with arrows on each process, this means that depending on the selected function the corresponding process will be evaluated, that uses the test image as input. For the output only the evaluated process will deliver data for the function projection (6.4). The process ‘Self Learning Feature’ (6.2.1) has more inputs compared to the other processes. This process uses the mask image to train the neural network and the function control (6.3) to control the functions used in this process. The processes ‘Average’ (6.2.3) and ‘Variance’ (6.2.2) are known parameters on images, and will therefore not be mentioned further.

The ‘Self Learning Feature’-process (6.2.1) will be explained by using the flowchart shown in figure 5.5.

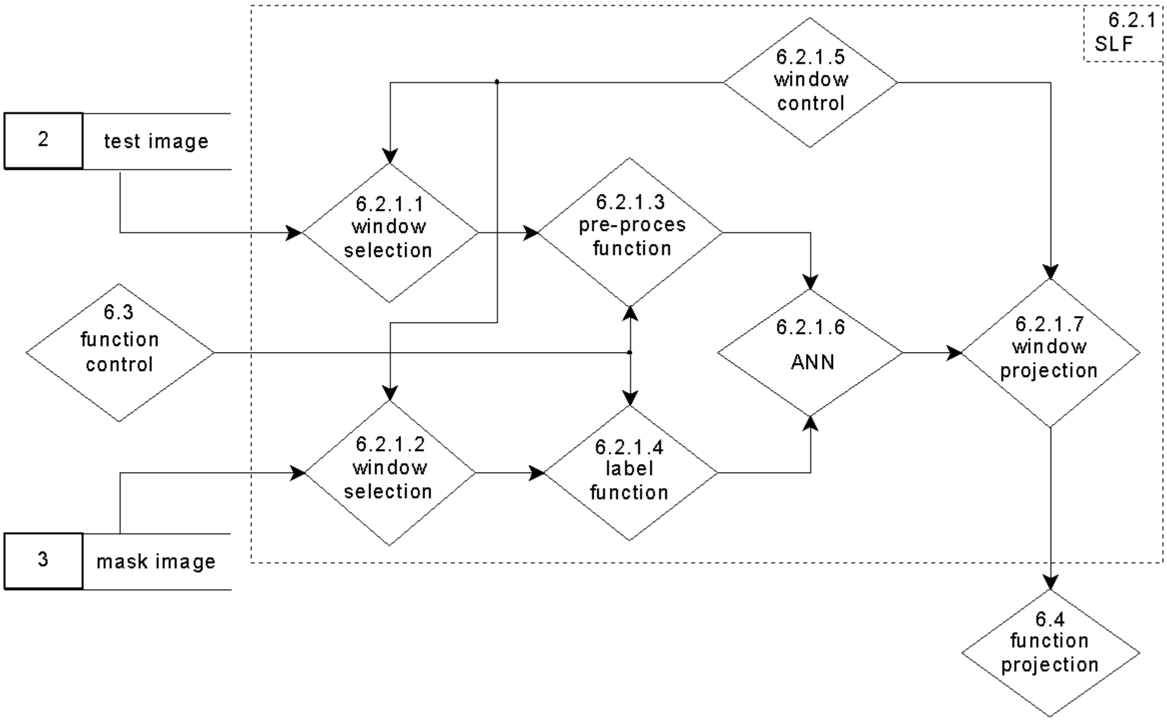


Figure 5.5 SLF overview

The function control (6.3) controls the pre-process function (6.2.1.3) and the label function (6.2.1.4) to be able to use other pre-processing or label functions. In this flowchart the images are split up into windows using a number of pixels horizontal and/or vertical as distance between the windows. The window control (6.2.1.5) controls the set of windows from the test image and the mask image, the window selection (6.2.1.1 and 6.2.1.2) selects a window from the test image and the mask image, and the window projection (6.2.1.7) connects the output of the ANN (6.2.1.6) to the selected window to construct a new image. This flowchart comprises two batch modes; the training mode to train the neural network and the operational mode to evaluate a test image. In the training mode[3] four steps are taken. First a window is selected from the test image and its mask image. Second, the window from the test image is being pre-processed to reduce the dimensionality of the window. Then the label function calculates a label from the window of the mask image. Several functions are possible for the label function[4]. In this thesis a 95 % rule is used to determine the label of a window. Windows that have a lower percentage receive an undefined label and will not be used in training. In the last step the neural network gets the pre-processed window and the calculated label as input. The neural network has to learn the pre-processed window so that, when evaluating this window, the calculated label is its answer. In the operational mode[5] the steps taken are almost the same as during the training mode except that in the operational mode the path from mask image (3) to ANN (6.2.1.6) is not used, because in this mode the neural network uses its learned data. For the ANN a feed forward neural network will be used.

---

[3] The used matlab script is present in appendix C as: “train_nn.m”.

[4] The used matlab script is present in appendix C as: “labelmask.m”.

[5] The used matlab script is present in appendix C as: “use_nn.m”.

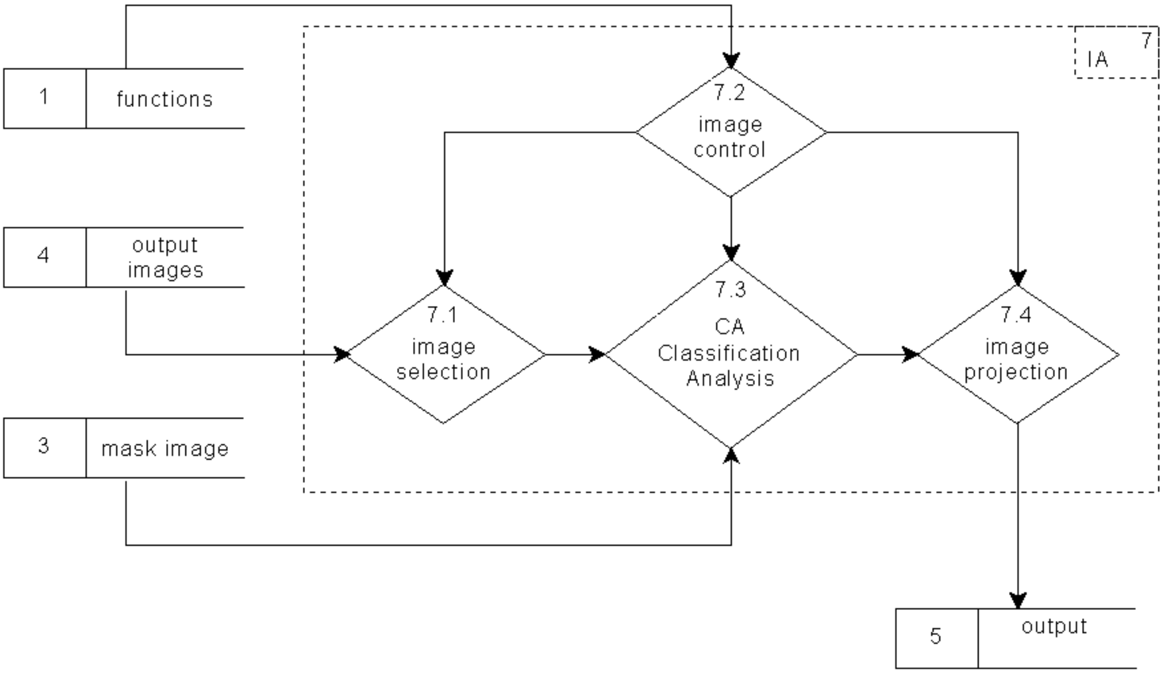


Figure 5.6 Image Analysis overview

Besides the ‘Image Processing’-process (6) the flowchart shown in figure 5.2 contained also an ‘Image Analysis’-process (7). The flowchart describing this process is shown in figure 5.6. In this flowchart the accompanying functions and their images are split up to perform a classification analysis between functions or images that results in an output analysis. This can, for instance, be a classification analysis between the average and variance and its classes or between the output of the neural network from a test image and its mask image.

6. SLF Pre-processing

In chapter 5 'SLF System development' the self learning image feature (SLF) was described. It contains a higher order neural network (HONN) that was constructed from a pre-processing part and an artificial neural network (ANN). The reason to use a HONN is that it reduces the size of a neural network compared to the 'brute force approach', that will not be described here. This reduction is possible because no training is required to achieve invariances as they can be built into the architecture of the network. The pre-processor selects part of the higher order terms. After pre-processing these, these will result in for instance a histogram, an auto-covariance, or a co-occurrence. Besides these transformations several other transformations can be used, such as those used in the ultrasound (i.e. backscatterspectrum).

The above approach has been tested [Huisman et al., 1995], but was not suited to be used in a practical situation because the transformation still resulted in a high dimensional input to the ANN. Two problems arise due to the high dimensional input:

1. By using the transformation on local windows in an image, as described in the previous chapter, an upper bound is introduced to the number of transformed values in making a statistical correct estimate. The reason for this results from the limited set of values in a window (i.e. a 10 x 10 window with 100 values is not suited to estimate a histogram with 256 bins).
2. The neural network itself is the second problem. Using a large number of inputs results in a large number of weights and when training a neural network to obtain a low generalization error, one needs a large number of training examples. As stated by [Hertz et al., 1993] the number of training examples has to be an order of magnitude larger than the number of weights (10, $=(0.1)^{-1}$, to expect a generalization error less than 0.1 (=10 %)). So when training a neural network with no hidden layers, 256 input neurons and a generalization error expectation of less than 10 % then 256 x $(0.1)^{-1}$ = 2560 uncorrelated windows are needed. Using a neural network with 1 hidden layer that has 256 neurons then 256 x 256 x $(0.1)^{-1}$ = 655360 uncorrelated windows are needed. This would result in a practically unfeasible set of data to be learned by a neural network.

A solution to this problem has to be found in the reduction of the dimension of the transformation.

This chapter describes a method to reduce the dimensionality of the pre-processed data. To test this method several experiments have been done. In the first section a description will be given of the reduction problem: difference-information loss. The second section will describe a method to quantify the difference-information and a necessary pre-transformation step to prevent unwanted conditions in the quantification method. In the third section experiments are done to find out if a pre-reduction can be done. The fourth section describes the methods used to reduce the high dimensional input to the ANN. In the fifth section the results from 8 experiments are shown. The sixth section gives a total overview of the HONN pre-processing. In the last section a summary resulting from the HONN pre-processing study is presented.

6.1 Information

In the introduction of this chapter the term experiment was used. During this study several experiments will be done. For the first experiment images will be used that have a Gaussian distribution with the following characteristics:

- $\mu = 0$ and $\sigma = 1$, further on called type I images.
- $\mu = 0$ and $\sigma = 2$, further on called type II images.

An example image, and its mask image, are shown in figure 6.1 and figure 6.2. In these figures the upper part is of type I and the lower part of type II.

For these experiments a window size of 5x5 is used. To determine if a window is of type I or of type II, the mask pixels in that window are evaluated. A requirement to classify a window as type I or type II is that the pixels of the mask window are uniformly labeled. For every image such a mask image is included.

In order to find methods to reduce the dimension of the transformation it is necessary to know what may and may not be reduced. For this study the choice has been made to use the histogram transformation. To be able to distinguish two types of histograms, they should be different. When this difference decreases, the two types of histograms become harder to distinguish. This difference will be called the difference-information from the two types of histograms. Because this difference is used to distinguish the two histograms, the transformation has to minimize the loss of the difference. As an example the difference histogram[6] of experiment 1 will be shown. Before this histogram can be produced, the histograms of type I and of type II have to be calculated. To get a good representative histogram of type I, a large set of type I histograms has to be averaged. A set of 100 images with 48 windows each has been used. Because for each window a histogram is calculated, this results in 100 x 48 = 4800 histograms of type I. The same procedure has been followed to determine a mean histogram of type II.

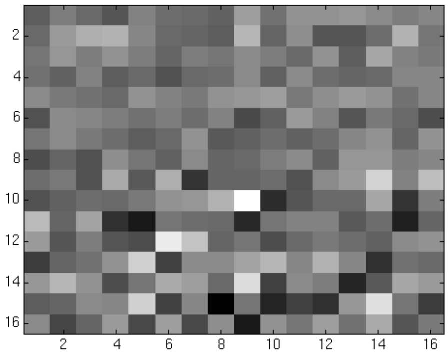


Figure 6.1 Example image with type I and type II data

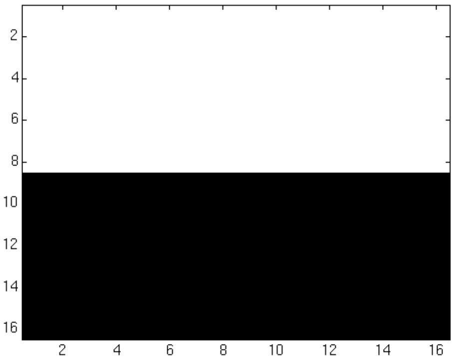


Figure 6.2 Mask image of figure 6.1

---

[6] The used matlab script is present in appendix C as: “verschilhist.m”.

The average histograms of type I and type II are shown in figure 6.3a and figure 6.3b. The difference, subtracting type II from type I, is shown in figure 6.3c. It shows were the difference-information concerning the discrimination between the two histograms is located.

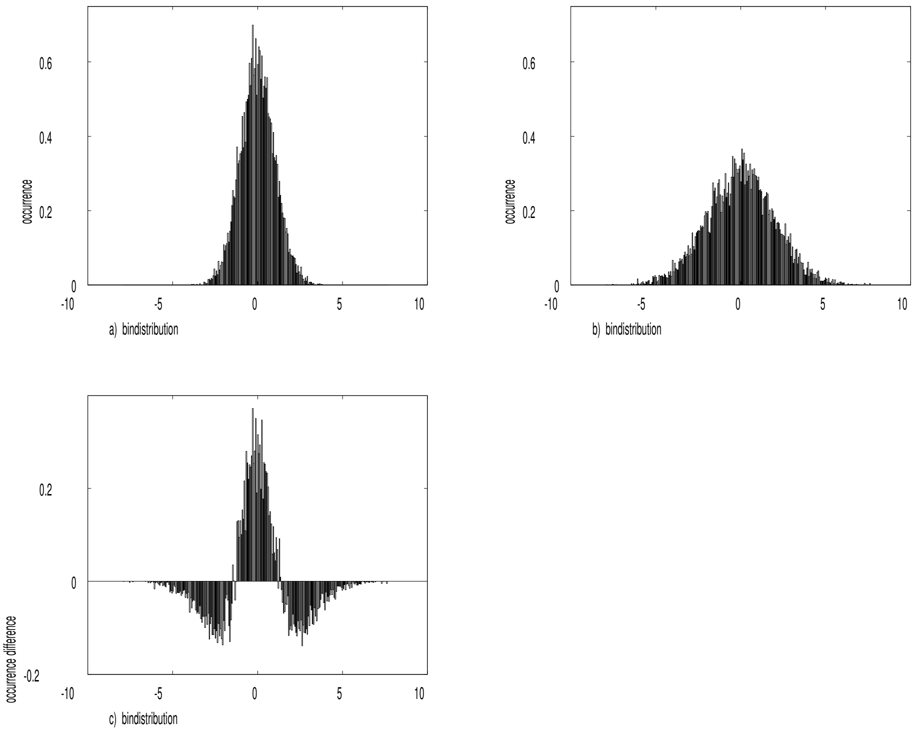


Figure 6.3 a) average histogram type I with window 5x5 and #bins=256.
b) average histogram type II with window 5x5 and #bins=256.
c) difference (type I - type II) histogram with window 5x5 and #bins=256.

In the above examples 256 bins were used to determine the histograms. However, this requires a large number of windows. The two problems, as stated in the introduction of this chapter, are not solved in this situation. A possible solution has to be found in the reduction of the number of bins. The kind of impact this would have on the difference-information is unknown, therefore a method to quantify the difference-information has to be found. This will be done in the following section.

6.2 Difference-information quantification

The amount of difference-information, contained in the difference between the two histograms, is low if the difference is small and increases if the difference is growing. A method that uses this difference is the chi-square-test[7] [Papoulis, 1991]. It was designed to test the difference between a histogram and a known distribution of occurrences. This known distribution of occurrences could be interpreted as a special kind of histogram. Namely, one from whom the distribution of occurrences is known, and seems therefore suitable in this situation. The chi-square-test uses the following formula:

---

[7] The used matlab script is present in appendix C as: “chi_square.m”.

$$\chi^2 = \sum_{i=1}^{M} \frac{\left(K_i - NP_{0,i}\right)^2}{NP_{0,i}} \qquad (1)$$

where

χ^2 : Average chi-square

K_i : Number of occurrences in bin i

N : Total number of occurrences

$P_{0,i}$: Probability of an occurrence in bin i

However, for this situation the above Eq. 1 has a disadvantage. When NP_0 is interpreted as a histogram then it calculates an asymmetric difference between the occurrences of two histograms. As an example figure 6.3a and figure 6.3b could be used as K and NP_0 in this chi-square formula that results in a chi-square value x. Using figure 6.3a as NP_0 and figure 6.3b as K in the chi-square formula would result in a chi-square value different from x. Therefore the following symmetric interpretation of the chi-square-test is used:

$$\chi^2 = \sum_{i=1}^{M} \frac{\left(K_{1,i} - K_{2,i}\right)^2}{\left(K_{1,i} + K_{2,i}\right)/2} \qquad (2)$$

where

χ^2 : Average chi-square

$K_{1,i}$: Occurrences of bin i in histogram 1

$K_{2,i}$: Occurrences of bin i in histogram 2

The histograms shown in the previous section were calculated using a linear bin-distribution for the histogram-function. As result of this it is possible that some bins in the denominator of equation 2 are very small. These small denominator values blow up the chi-square value. Therefore, the chi-square value becomes highly sensitive to noise (see figure 6.3c). If this occurs the chi-square-test would be useless, because the chi-square values would not converge to an optimum. To solve this problem, two solutions can be distinguished. The first solution would be to generate more test images to increase the number of occurrences at those bins that are not likely to occur. This in turn will increase the test size that was set out to be kept small. The second solution is to prevent the denominator to become very small by applying a transformation on the denominator.

The transformation, called the pre-transformation, that realizes this requirement is the equalization transformation on the denominator of the chi-square-test. This transformation will try to equalize the occurrences in the denominator by assigning a non-linear bin-distribution. The equalization transformation results in an equalized bin-distribution of the sum of the two histograms to get a correct denominator in the chi-square-test. However, performing an equalization transformation without obtaining ‘holes’ [Gonzalez et al., 1987] in the histograms requires that images have been digitized in floating point precision. Visualizing the pre-transformation[8] results in figure 6.4.

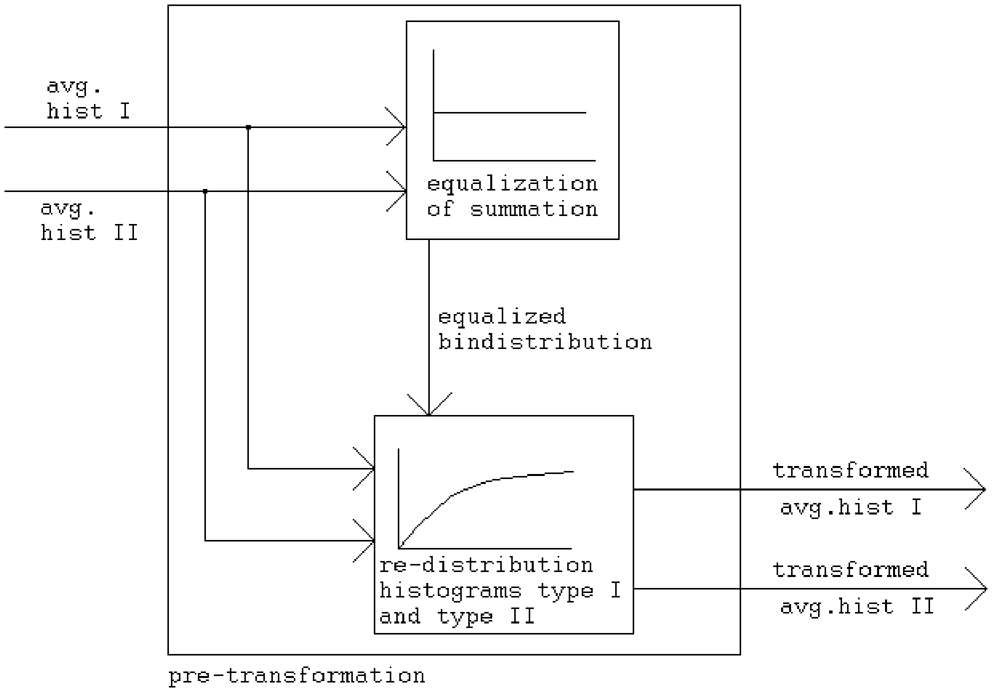


Figure 6.4 Pre-transformation

To summarize this section on how to calculate the difference-information between two histograms. The first step is to determine the equalization transformation[9] that is derived from the average sum of the two histograms. Second is to apply this transformation on both the input histograms. The third step is to use the transformed[10] histograms on Eq. 2 that calculates an estimate of the difference-information between these two histograms. Further is assumed that images are digitized in floating point precision.

6.3 Pre-reduction possibility

To solve the problems, as stated in the introduction of this chapter, a reduction of the number of bins is needed with a minimum of difference-information loss. At this point two possibilities can be distinguished for the reduction:

1. Reducing the number of bins during the histogram calculation
2. Reducing the number of bins during the equalization transformation.

In this section experiments are carried out to test how well the reduction performs when this is done immediately during histogram calculation or during equalization transformation. The histogram calculation as well as the equalization transformation are necessary to calculate and quantify the

---

[8] The used matlab script is present in appendix C as: “pre_trans.m”.

[9] The used matlab script is present in appendix C as: “equtransf.m”.

[10] The used matlab script is present in appendix C as: “verdeelhistdata.m”.

difference-information. In the first possibility the histogram is calculated with a reduced number of bins followed by the histogram equalization. For the second possibility the histograms are calculated at maximum number of bins and a bin-reduction is achieved during the equalization. First the reduction during histogram calculation is considered. The histograms and the accompanying difference are shown in figure 6.5, figure 6.6 and figure 6.7 for a reduction to 10, 50 and 256 bins respectively.

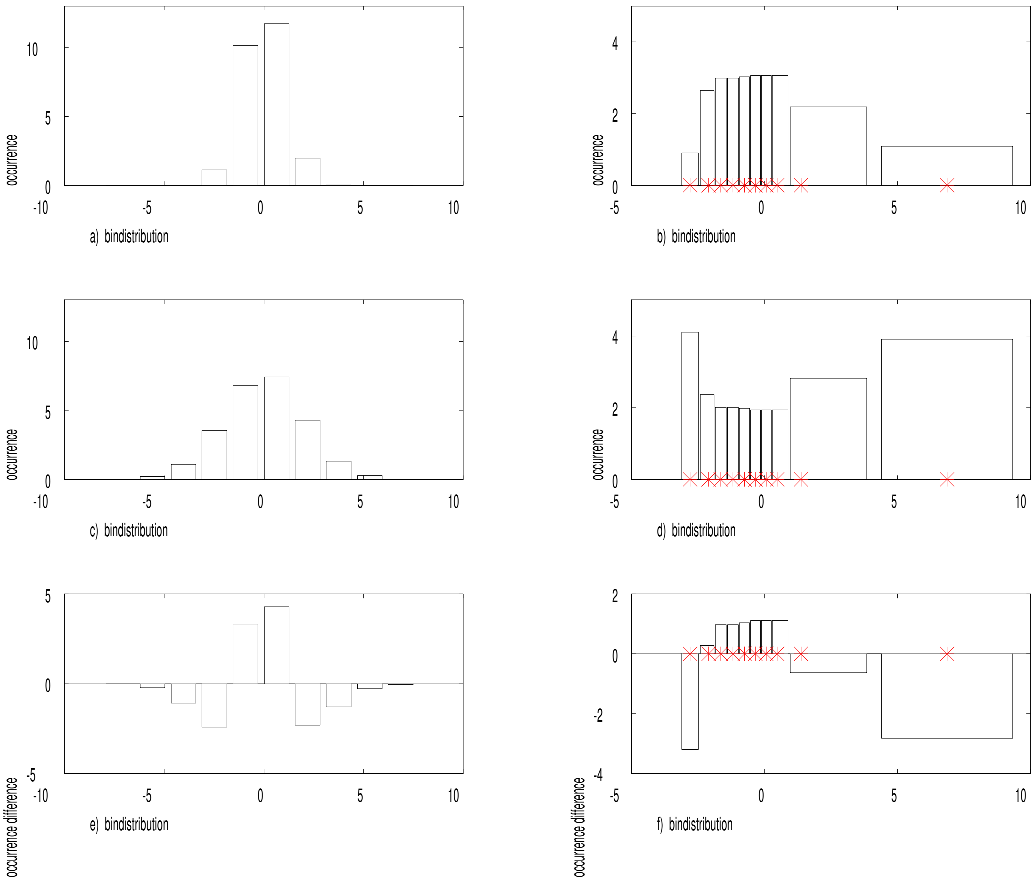


Figure 6.5 window 5x5 and #bins=10.
a) histogram type I before equalization. b) after equalization.
c) histogram type II before equalization. d) after equalization.
e) difference histogram before equalization. f) after equalization

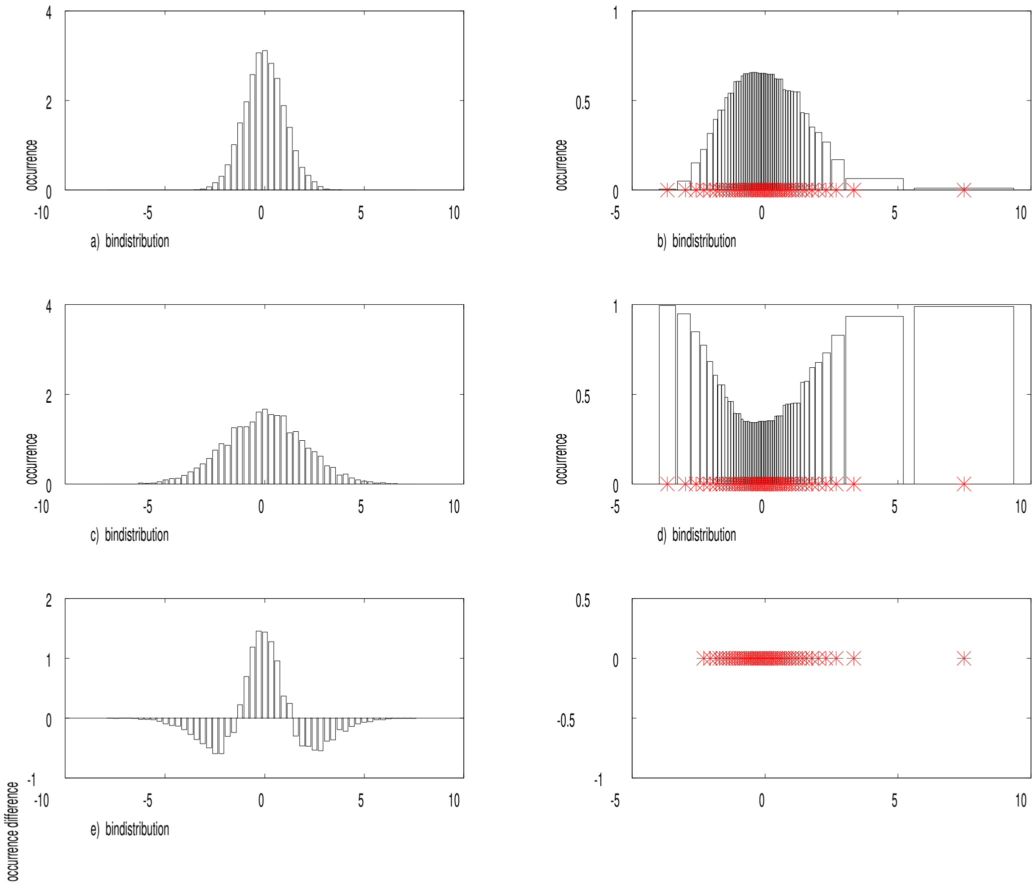


Figure 6.6 window 5x5 and #bins=50.

a) histogram type I before equalization. b) after equalization.
c) histogram type II before equalization. d) after equalization.
e) difference histogram before equalization. f) after equalization.

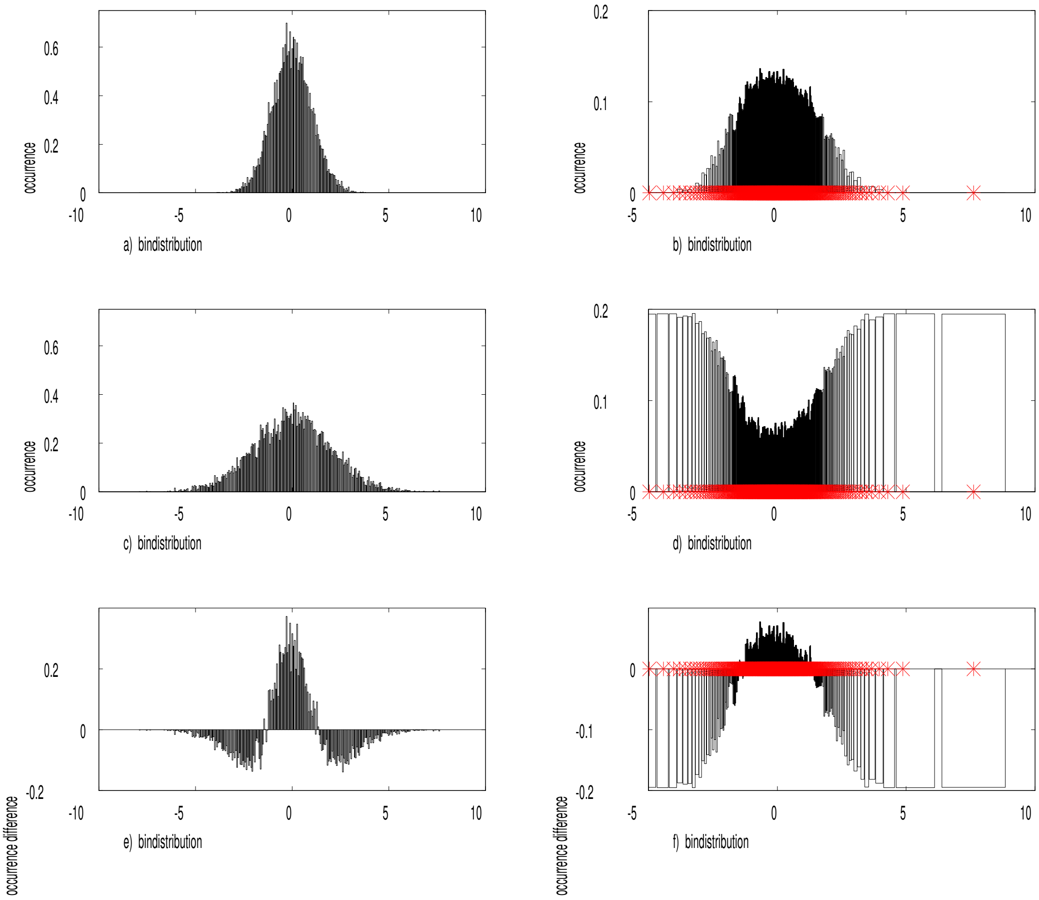


Figure 6.7 window 5x5 and #bins=256.
a) histogram type I before equalization. b) after equalization.
c) histogram type II before equalization. d) after equalization.
e) difference histogram before equalization. f) after equalization.

Images[11] used have a dynamic range of 8 bits, so the maximum number of bins is 256. The chi-square-value will reach a maximum at that point, this maximum will be called the optimal chi-square-value. The result of the chi-square-test on the above three situations, after equalization as already mentioned, are shown in table 6.1.

	Chi-square	% difference-information loss to optimum
10 bins	10.1946	34.18 %
50 bins	14.9377	3.56 %
256 bins	15.4886	0 %

Table 6.1 Chi-square values of results in figure 6.5, figure 6.6 and figure 6.7.

---

[11] Images are assumed to be digitized in floating point precision.

Looking at table 6.1, the following conclusions can be drawn. First, the reduction to 10 bins during the histogram calculation is unacceptable, because about $^1/_3$ of the difference-information is lost. Second, the reduction to 50 bins during the histogram calculation is a lot better, but the loss is still too much when evaluating the subsequent reduction algorithm. Thus, it can be concluded that reducing the bins during the histogram calculation results in difference-information loss, except for the situation with 256 bins. Because the images are assumed to be digitized in floating point precision, it is possible and therefore optimal to use 256 bins for the histogram calculation.

The second option to reduce the input, in this section, is a reduction during the equalization transformation. Results of the combined reduction and equalization from 256 to 50 bins are shown in figure 6.8.

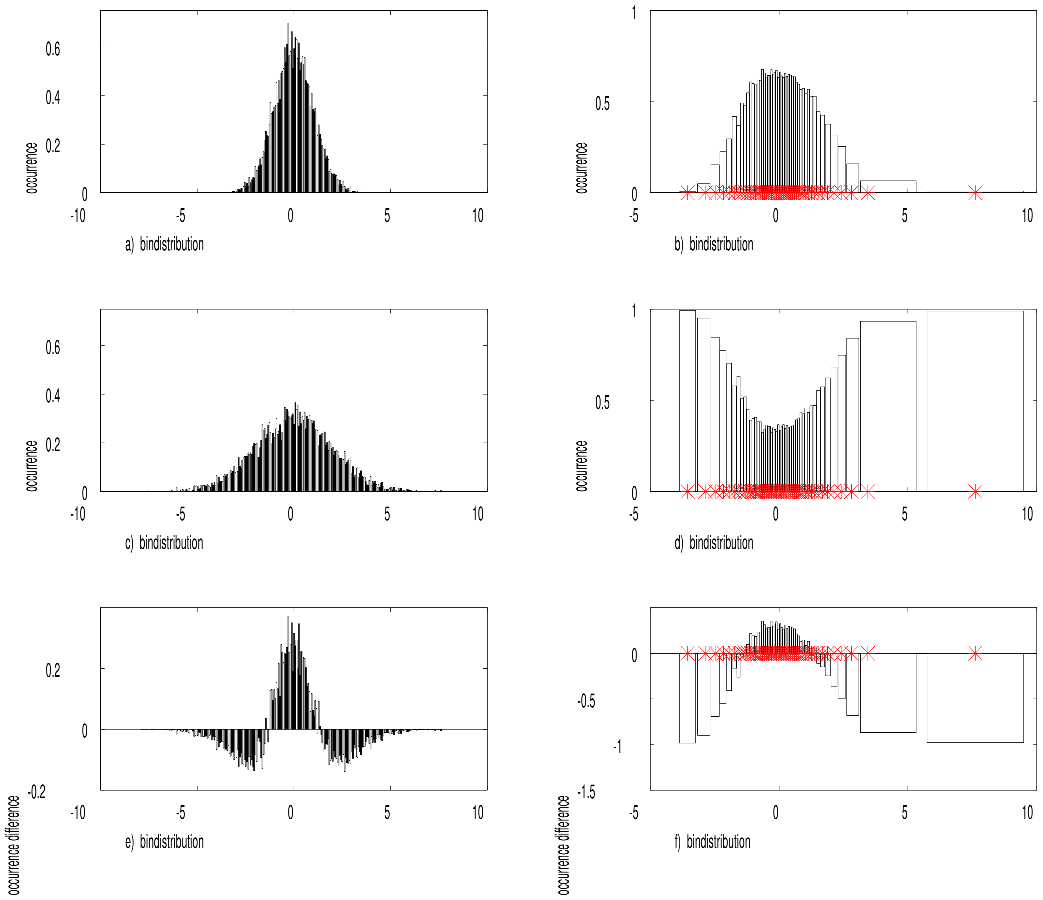


Figure 6.8 window 5x5 and #bins=256.

a) histogram type I before equalization. b) after equalization and reduction to #bins=50.
c) histogram type II before equalization. d) after equalization and reduction to #bins=50.
e) difference histogram before equalization. f) after equalization and reduction to #bins=50.

The result of the chi-square-test on the above situation (see figure 6.8) is shown in table 6.2.

	Chi-square	% information-loss to optimum
50 bins	15.1976	1.88 %
256 bins	15.4886	0 %

Table 6.2 Chi-square values of result in figure 6.8

Reducing during the equalization is about a factor of 2 better than when reducing during the histogram calculation, but still too much when regarding the equalization to 256 bins. The conclusion drawn for the reduction during the histogram calculation can also be applied here; Because images are assumed to be digitized in floating point precision, it is possible and therefore optimal to use 256 bins for the equalization transformation.

For this section it can be concluded that doing the reduction in the process of obtaining and quantifying difference-information results in unwanted difference-information loss. Besides that, the more difference-information stays available, the better the best reduction method in section 6.4 can be determined. Only the situation that the histograms are calculated at maximum number of bins and an equalization is done at maximum number of bins results in no difference-information loss (see figure 6.7). Because images are assumed to be digitized in floating point precision, it is possible and therefore optimal to use the maximum number of bins.

6.4 Reduction methods

At this moment it is known how to obtain and quantify the difference-information in an image without loss of difference-information. In this section a reduction method has to be found that reduces the high dimensional input to the ANN with a minimum of difference-information loss. Three kinds of methods can be distinguished to reduce the number of bins:

1. Reducing the bins in the equalized bin-distribution in a linear way. This method, called the linear method[12], divides the bin-distribution in equidistant bins where the bin-width depends on the number of bins.
2. Reducing the bins in the equalized bin-distribution by using a transformation function, equal to the cumulative distribution of the absolute difference histogram, that produces a histogram whose occurrences have a uniform density [Gonzalez et al., 1987]. This method, called the equalization method[13], divides the y-axis of the cumulative distribution in equidistant parts where the width depends on the number of bins.
3. Reducing the bins in the equalized bin-distribution by using an optimization method that minimizes the difference-information loss. This method, called the minimization method[14], uses the bin-distribution of the linear method and the equalization method as initialization to find the best bin-distribution for a given number of bins. This optimization method uses the simplex search algorithm of Nelder and Mead [Nelder et al.] and tries to find the minimum of a scalar function[15]. The scalar function is in this situation a function that determines the difference between the optimal chi-square and the chi-square of a bin-distribution, determined by the optimization method.

A requirement to use the above methods is that these methods have to be ‘trained’ with a data-set. This so called training means that to correctly reduce the bins with the above methods, these methods

---

[12] The used matlab script is present in appendix C as: “bepaallinbin.m”.

[13] The used matlab script is present in appendix C as: “bepaalequbin.m”.

[14] The used matlab script is present in appendix C as: “bepaalminbin.m”.

[15] The used matlab script is present in appendix C as: “bepaalverdeel.m”.

have to use an average difference histogram calculated by subtracting two types of averaged histograms. These averaged histograms have been calculated by averaging a large set of histograms, and the larger this set is the smaller the influence of noise will be.

The resulting bin-distributions from the reduction methods will be referred to as transformation vectors, and the best transformation vector will be used in the pre-processing unit. Visualizing this reduction process[16] results in figure 6.9. Using the three reduction methods, for the #bins=2..10, on the first experiment (see begin of section 6.1) results in the graph shown in figure 6.10.

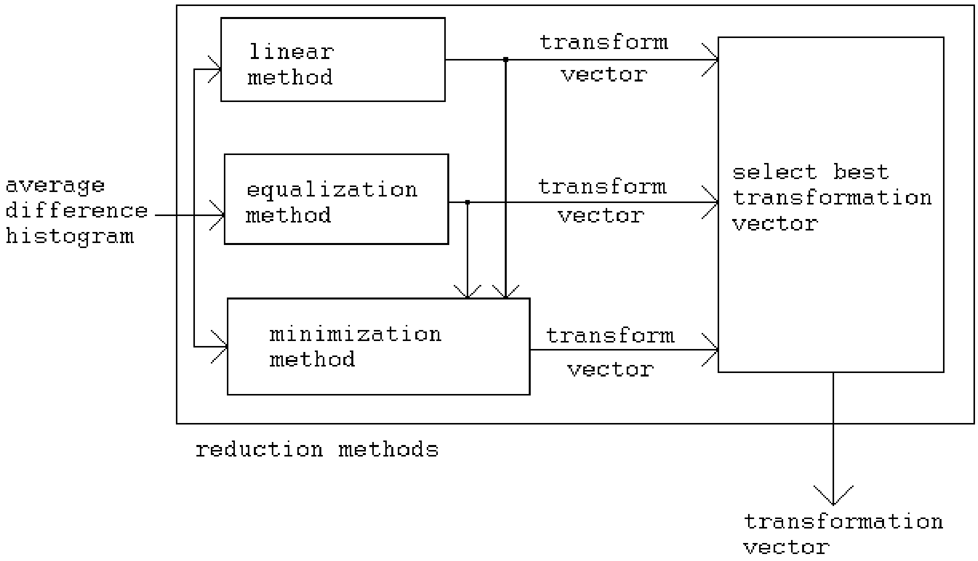


Figure 6.9 Reduction methods

---

[16] The used matlab script is present in appendix C as: "reduct_meth.m".

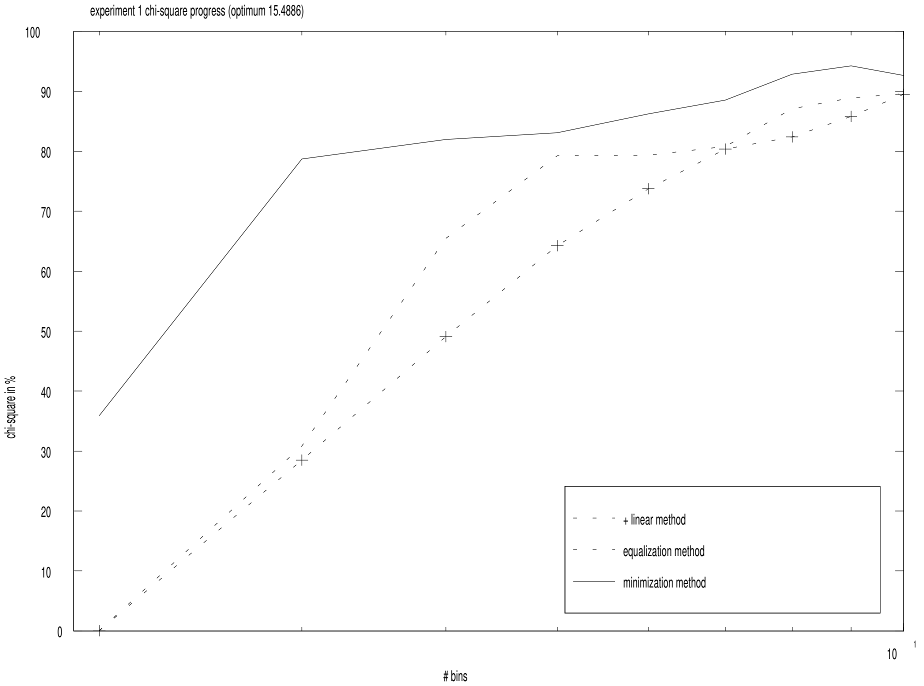


Figure 6.10 Chi-square results from experiment 1 (#bins=2..10).

For this experiment all the three methods, see figure 6.10, have a difference-information loss of 10 % or less when the number of bins is reduced to 10. Because the maximum available difference-information is used, all three methods have their best results for this experiment. The equalization method performs slightly better than the linear method, but the minimization method has the best results. This method has a difference-information loss of less than 10 % at a reduction to 8 bins. Therefore, when the high dimensional input is reduced to a low dimensional input, the best performance in this experiment is obtained by the minimization method.

6.5 Other experiments

In the first experiment that was done, a Gaussian distribution with $\mu = 0$ and $\sigma = 1$ (type I) and $\sigma = 2$ (type II) was simulated. For that experiment some conclusions concerning the results of the reduction methods were drawn. In order to draw more general conclusions on the reduction methods a total of 8 more experiments were done. The first four experiments have a Gaussian distribution with a different μ and σ compared to experiment 1. After that, four experiments are done with a Rayleigh distribution.

6.5.1 Experiments 2 to 5

These experiments are concerned with a Gaussian distribution. Table 6.3 gives an overview of the parameters in these four experiments.

	Type I-image	Type II-image

Experiment 2	$\mu = 1, \sigma = 1$	$\mu = 2, \sigma = 2$
Experiment 3	$\mu = 1, \sigma = 2$	$\mu = 2, \sigma = 2$
Experiment 4	$\mu = 1, \sigma = 2$	$\mu = 3, \sigma = 2$
Experiment 5	$\mu = 1, \sigma = 2$	$\mu = 5, \sigma = 2$

Table 6.3 Other experiments with Gaussian distribution.

The chi-square results of these experiments are displayed in figure 6.11. In every graph the x-axis represents the number of bins in the range 2..10 and the y-axis represents the estimated difference-information, or chi-square value, in percentages of the optimum chi-square value.

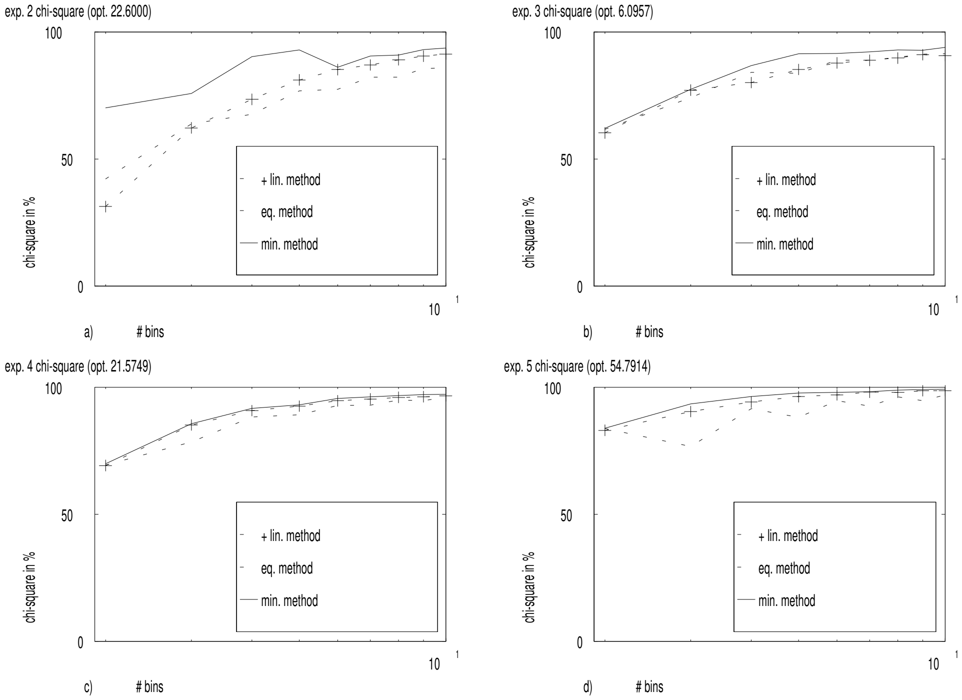


Figure 6.11 a) Results experiment 2. b) Results experiment 3. c) Results experiment 4. d) Results experiment 5.

For these experiments, see figure 6.11, all three methods, except the equalization method in experiment 2, have a difference-information loss of 10 % or less when the number of bins is reduced to 9. The linear method generally performs better than the equalization method, but the minimization method has the best results.

6.5.2 Experiments 6 to 9

Because ultrasound involves Rayleigh[17] distributed data some experiments have to be done with that distribution. Table 6.4 gives an overview of the parameters in these four experiments.

	Type I-images	Type II-images
Experiment 6	$\mu = 1$	$\mu = 2$
Experiment 7	$\mu = 2$	$\mu = 3$
Experiment 8	$\mu = 3$	$\mu = 4$
Experiment 9	$\mu = 2$	$\mu = 4$

Table 6.4 Experiments with Rayleigh distribution

The chi-square results of these experiments are displayed in figure 6.12. In every graph the x-axis represents the number of bins in the range 2..10 and the y-axis represents the estimated difference-information, or chi-square value, in percentages of the optimum chi-square value.

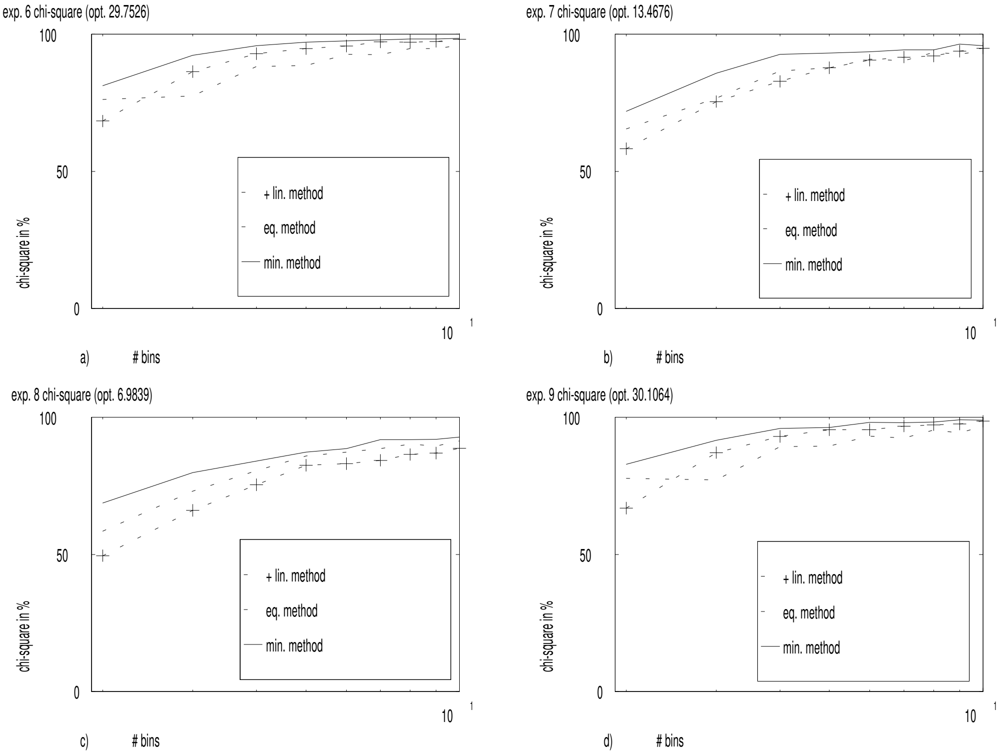


Figure 6.12 a) Results experiment 6. b) Results experiment 7. c) Results experiment 8.
d) Results experiment 9.

For these experiments, see figure 6.12, all the three methods, except the linear method in experiment 8, have a difference-information loss of 10 % or less when reducing the number of bins to 10. The linear method performs better than the equalization method when the chi-square is high, see experiment 6 and 9. At a low chi-square, see experiment 8, the equalization method performs better

---

[17] The used matlab script is present in appendix C as: “randrayl.m”.

than the linear method. The results of experiment 7 lie in-between. The method that performs best in these experiments is the minimization method.

When all results in this chapter are compared it can be concluded that the linear method performs better than the equalization method when the chi-square is high. At a low chi-square the equalization method performs better than the linear method. However, the best performance in obtained by the minimization method.

6.6 Pre-processing overview

Now that all theory and methods, necessary for the HONN pre-processing unit, have been described an overview will be given how this will be used in the system. In figure 6.13 a global overview of the HONN[18] is given.

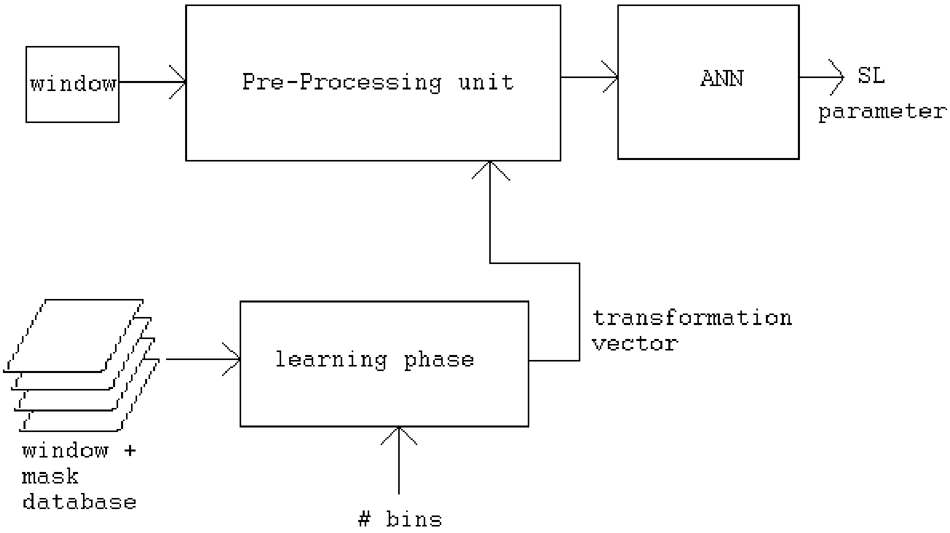


Figure 6.13 Global HONN overview

The pre-processing unit gets a window as input, and delivers the pre-processed window to the neural network, that determines a Self Learning Image Feature. This pre-processing unit pre-processes the input by reducing the number of bins in the histogram of the input. In order to pre-process the input correctly and consistently the pre-processing unit[19], see figure 6.14, has to learn a transformation vector for the re-distribution or reduction.

---

[18] The used matlab script is present in appendix C as: “honn.m”.

[19] The used matlab script is present in appendix C as: “preproces.m”.

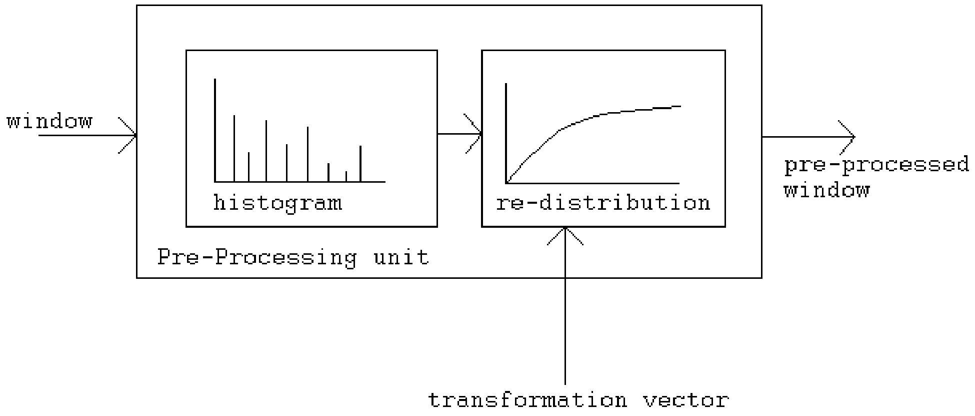


Figure 6.14 Overview HONN Pre-process

To correctly initialize this transformation vector[20] a large number of windows, and the masks, have to be used, and several steps have to be taken, see figure 6.15.

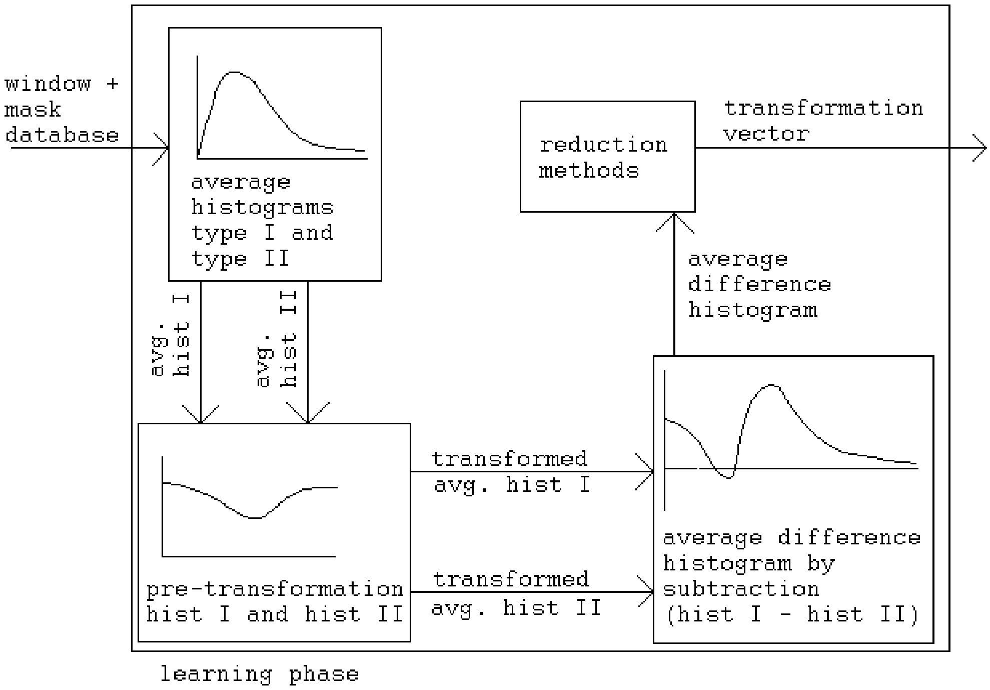


Figure 6.15 Overview learning phase

The first step is to calculate the histograms of the windows, determine if a histogram is of type I or type II, and then to average the histograms of both types. Results of this operation are the average histograms of type I and type II. In the second step these histograms will be used as input for the pre-transformation, explained in section 6.2, that results in the transformed average histograms of type I

---

[20] The used matlab script is present in appendix C as: “learnpreproces.m”.

and type II. The third step calculates the average difference histogram by subtracting both types of averaged histograms. This average difference histogram will then be used, as input for the reduction methods (see section 6.4), to determine the transformation vector for the pre-processing unit.

6.7 Summary

Using high dimensional input for a neural network results in a practically unfeasible amount of data to be learned by a neural network. Therefore a transformation on the input has to be done to reduce the input. A requirement for this transformation is that it minimizes the difference-information loss. The difference-information in two histograms is contained in the difference between these two histograms. To control the difference-information loss a method is needed to quantify the difference-information. The chi-square-test, designed to test the difference between two histograms, is suitable to be used as quantifier. However, before this chi-square-test can be used it is required that a symmetric interpretation of this test is used and that an equalization transformation is done on the denominator to prevent small denominator values.

Reducing the number of bins during the histogram calculation and/or equalization transformation results in difference-information loss. Because images used are assumed to be digitized in floating point precision, it is possible and optimal to use 256 bins for the histogram and the equalization. Besides this reduction, experiments have been done to reduce the number of bins using three reduction methods. From all experiments it can be concluded that the linear method performs better than the equalization method if enough difference-information is available, otherwise the linear method is equal or worse than the equalization method. The minimization method performs best in all experiments. However, the minimization method uses the results of the linear method and the equalization method as initialization.

The method to reduce the dimensionality of the pre-processed data consists of two parts. First a histogram of the window is calculated, and after that a transformation on the histogram is done to reduce the input for a neural network. This transformation uses a learning phase to reduce the input correctly and consistently. Therefore it uses the three reduction methods to determine the transformation vector with the best chi-square value. Because the transformation uses a learning phase to reduce the input, a possible fourth method to reduce the number of bins could be using a neural network. A separate study is needed to investigate this possibility as a method that performs maybe even better than the minimization method.

7. SLF System Evaluation

In this chapter the developed SLF system will be evaluated. The goal is to find out how the SLF behaves in a realistic ultrasound environment compared to the fixed features. Before the SLF can be compared some ANN parameters have to be tuned. These parameters are SLF parameters such as network configuration (i.e. #input neurons, #hidden layers and #neurons in the hidden layers) and the #bins that are used.

To make a realistic comparison between the performance of the SLF and the fixed features a realistic environment is necessary. In order to create such an environment, images of a realistic size will be used. Besides that, the number of images will be limited to a realistic number, because there are seldom enough images available in practice. In some situations this limitation could lead to insufficient training of the neural network.

7.1 Datasets

Before the SLF can be used to analyze images, the neural network of the SLF, as already mentioned, has to be trained. For this training a training set with a particular problem is used. To validate if the neural network has learned enough, too much or too less about a particular problem an independent set, the validation set, is used having the same problem.

In this chapter simulated ultrasound images with a realistic size and a realistic number are used. The goal of these simulations is to study the effect of a decreasing difference in scatter density between normal and abnormal tissue. Therefore training- and validation sets will be simulated[21] with a scatter density of 3, 6, 10, 15 and 20 in a resolution volume. The normal (or healthy) tissue will have a scatter density of 20 and for the abnormal tissue the scatter density will vary. For every difference in scatter density between normal and abnormal a dataset is constructed. When the difference in scatter density between 3 and 20 is considered, the images in the training- and validation set will both have a part with a scatter density of 3 and another part with a scatter density of 20.

For the simulated ultrasound images the speckle[22] will be sampled to a fixed size of 1x1 pixel[23]. The reason is that the number of speckles, and not the number of pixels, is determining the effective extent of data. An advantage for using such a speckle size is the reduction in necessary data to get a realistic environment. Therefore, it is possible to use a set of 10 images with a size of 100x100 pixels[24] for every training- and validation set. In this situation a set of images needs about 3 Mb hard drive space instead of 63 Mb when a speckle size of 3x7 pixels was used. The total necessary hard drive space is therefore only 150 Mb instead of 3150 Mb. This decrease in necessary data also results in a decrease of time needed to perform an experiment. Performing an experiment on the largest difference in scatter density with a window size of 5x5 speckles and 5 bins, a little over 3 hours[25] are needed now. Using a speckle size of 3x7 pixels this time would, in the first place, increase because of the increase in data. However, the time could also increase because of an increase in memory usage. As a result

---

[21] The used matlab script is present in appendix C as: “nesbeeld3.m”.

[22] The used matlab script is present in appendix C as: “ses.m”.

[23] In practice the speckle has a size of 3x7 pixels.

[24] In practice the image has a size of 300x700 pixels.

[25] This measurement is done on a Sun Sparc 20 with 64 Mb memory and with a speckle size of 1x1 pixel.

of this, the possibility exists that hard drive space is used temporarily to store data, the so called "swapping" effect. Other factors that have an influence on the used time are the window size, the #bins and the problem in the dataset.

In the set of simulated images every scatter density will have 4 training- and validation sets. The reason to do this is to determine a mean correct classification percentage as result of an experiment and a raw estimation of the variability. This also prevents that in some experiments high results are obtained by coincidence. A better average could be determined by enlarging the dataset with, for instance, a factor of 4. However, the objection is that the dataset is not realistic anymore. Furthermore, the possibility that the generalization error appears also decreases.

7.2 Classification

To distinguish between the two classes, a parametric or a non parametric classifier could be used. The reason to use a classifier is to determine how well it is possible to correctly classify a datapoint in a particular experiment. A parametric classifier is the Mahalanobis distance that is described in appendix A. Before the Mahalanobis distance can be used, all data must have an equal probability [McLachlan, 1992] of belonging to the available classes. Besides that, the features must have a multi variate normal distribution. A nonparametric classifier is the k-Nearest Neighbour (k-NN) [McLachlan, 1992] that is described in appendix B. In this chapter the k-NN classifier will be used as classifier for the experiments of the fixed features and the SLF. The reason not to use the Mahalanobis distance to estimate the correct classification percentage in these experiments is twofold. First of all, it is not fair to compare the performance of the fixed features and the SLF, because the neural network has learned the difference between the two classes. Second, the fixed features and the SLF must be multi variate normal distributed, this is not always true.

7.3 Experiments

The experiments that are performed in this chapter are related to the following items:

1. Calculating the fixed features (mean, variance, skewness and kurtosis) for a variation in window size (5x5, 7x7 and 10x10 speckle's). Thereby determining the influence of the window size on the correct classification percentage of the fixed features. However, this is not the most important reason. Primarily, these results are used as reference values for comparison with the SLF results.

2. Determining the SLF for a variation in #bins (5..10) and window size (5x5, 7x7 and 10x10 speckle's). Thereby determining the influence of the #bins on the correct classification percentage for every window size. Further an optimal #bins can be determined for every window size.

3. Determining the SLF for a variation in #neurons in the hidden layer. For this the proportion between the #neurons of the input layer and the #neurons in the hidden layer will be changed. Thereby determining the influence of the #neurons in the hidden layer on the correct classification percentage. Further an optimal #neurons in the hidden layer can perhaps be determined.

7.4 Choice of the neural network

In [Reijnen, 1994] and [Huisman et al., 1995] it was shown that a multi layer feed forward neural network would be a suitable choice to be used in a self learning image feature system. Research was done to find out the most suitable trainingmethod when the system has to learn to distinguish between two classes. Especially learning small differences in the data turned out to be decisive for the choice of the trainingmethod.

In the used signal processing environment [Matlab,1992] various trainingmethods are available for feed forward neural networks. Two of these will be reviewed. The first is the fast backpropagation neural network: “trainbpx” [Matlab,1993]. This one is referred to as fast backpropagation because it has some optimizations compared to normal backpropagation networks. These optimizations are momentum and adaptive learning rate. Using the fast backpropagation to learn to distinguish between, for instance, a Rayleigh distribution with $\mu_1 = 3$ and a Rayleigh distribution with $\mu_2 = 4$ then after 3000 epochs the results in column two of table 7.1 are obtained. For this experiment the training- and validation sets consisted of 14 images each 50x50 pixels in size. The neural network had a 5-10-2 architecture and a window size of 5x5 pixels was used.

Data	mean correct classification percentage for fast backpropagation method after 3000 epochs	mean correct classification percentage for Levenberg-Marquardt after 30 epochs
training set	93.2 %	93.0 %
validation set	92.8 %	92.6 %

Table 7.1 Mean correct classification percentage trainingsmethods.

A second trainingmethod is Levenberg-Marquardt: “trainlm”, [Matlab,1993] [Bishop,1995] and [Norgaard, 1995]. The results of this method are shown in column three of table 7.1. These results were obtained after 30 epochs. However, the time needed to train an epoch with Levenberg-Marquardt is not equal to the time needed to train an epoch with fast backpropagation. An epoch with Levenberg-Marquardt takes approximately a factor 4 more time. Using the training- and validation set to be evaluated by both neural networks, after these have been trained, results are obtained as shown in figure 7.1. Besides that, both neural networks try to minimize the sum squared error. This curve is shown in figure 7.2.

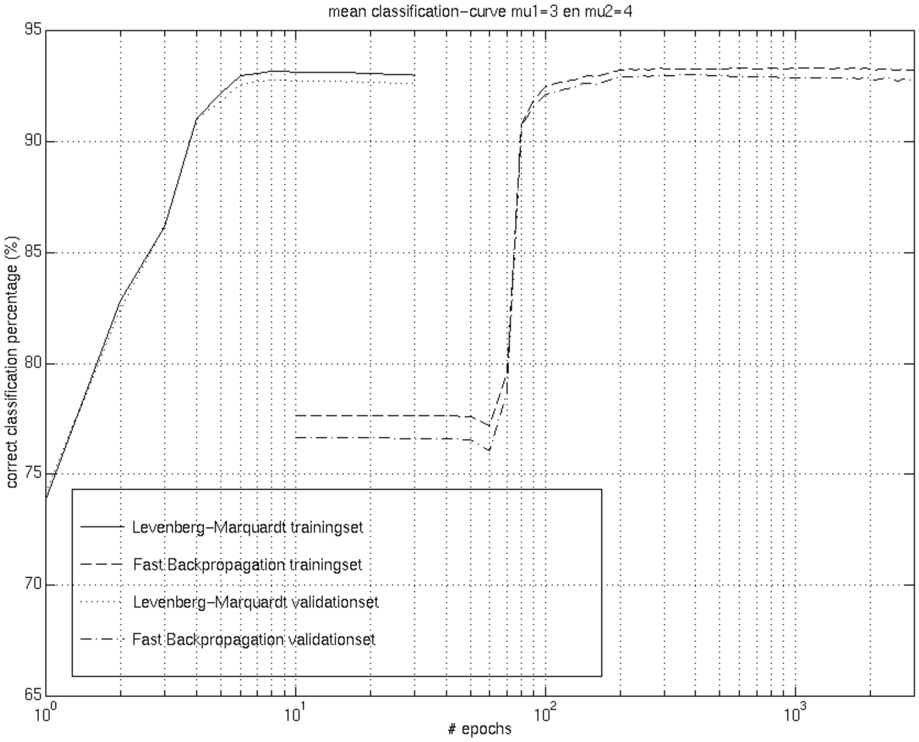


Figure 7.1 Mean correct classification percentage curve Rayleigh distributed datasets.

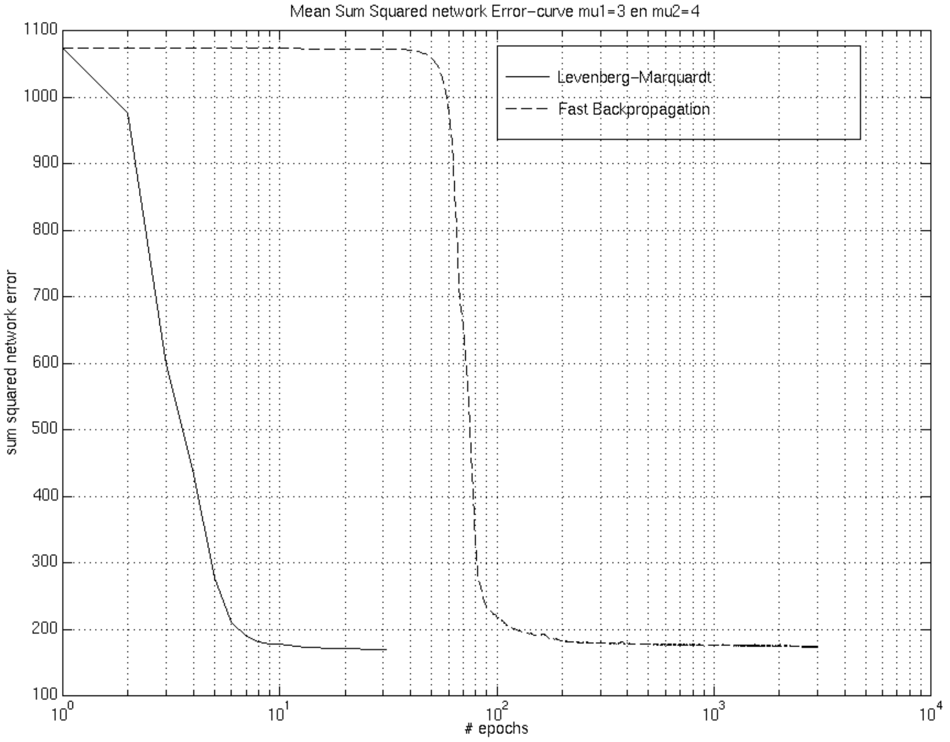


Figure 7.2 Mean SSE curve Rayleigh distributed datasets.

At 30 epochs the Levenberg-Marquardt achieves a mean sum squared error of 169 and a mean Mahalanobis distance of 2.96 for the training set and 2.9 for the validation set. The fast backpropagation achieves a mean sum squared error of 173.4 and a mean Mahalanobis distance of 2.99 for the training set and 2.93 for the validation set.

Due to the learning rate of Levenberg-Marquardt compared to fast backpropagation, as is shown in figure 7.1 and figure 7.2, Levenberg-Marquardt will be used in further experiments, and the neural network will be trained for a maximum of 30 epochs.

7.5 Experiment with fixed features

In this experiment a mean correct classification percentage is estimated for each window size and for each dataset. The fixed features that will be used are: mean, standard deviation, skewness and kurtosis. Items that will vary in this experiment are the dataset, shown in table 7.2, and the window size. The windows that will be used have a size of 5x5, 7x7 and 10x10 speckle's. Thereby doubling the #speckle's in a window when increasing the window size.

Dataset	Scatter density type I	Scatter density type II
dataset 1 (3-20)	3	20
dataset 2 (6-20)	6	20
dataset 3 (10-20)	10	20
dataset 4 (15-20)	15	20

Table 7.2 Datasets (valid for training- and validation set).

The result obtained for the fixed features (mean, standard deviation, skewness and kurtosis) with the dataset that has the smallest difference in scatter density is primarily described by the first two features. An explanation is that the data of the two classes are almost identical and almost Rayleigh distributed. A comparison between the results of the first two features and all four features is given in table 7.3. From this table a surprising result can be derived. It appears that using more features does not guarantee a better or equal result. From the used dataset it appears that using the four features, compared to the two features, results in a growing decrease in performance when using window sizes of 7x7 speckle's or larger.

Window size	mean correct classification percentage for the features mean and standard deviation	mean correct classification percentage of the 4 fixed features
5x5 training set	78.4 %	79.5 %
7x7 training set	84.7 %	84 %
10x10 training set	92 %	85.7 %
5x5 validation set	78.5 %	78.9 %
7x7 validation set	84.5 %	83.7 %
10x10 validation set	91.5 %	86.9 %

Table 7.3 Comparison results features mean & standard deviation with the fixed features.

Because of the growing decrease in performance at an increasing window size, only the features mean and standard deviation will be used as fixed features.

Now that all parameters are available for this experiment the correct classification percentage can be estimated for the fixed features. The results shown in figure 7.3 and figure 7.4 are used as reference values for comparison with the SLF results. For each mean correct classification percentage its minimum and maximum values are also given. These are represented by the two short horizontal lines connected by a vertical line.

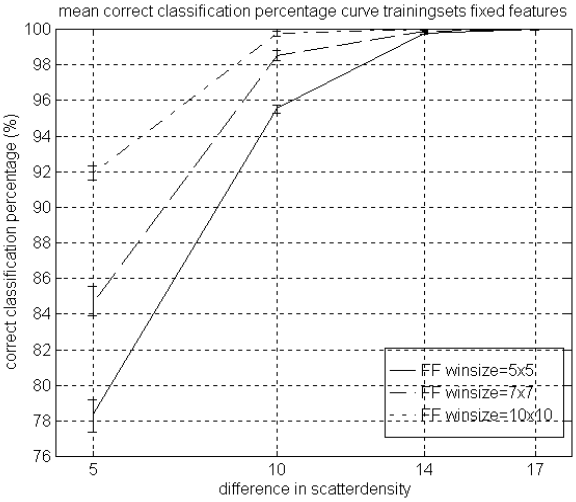


Figure 7.3 Mean correct classification curve training set fixed features.

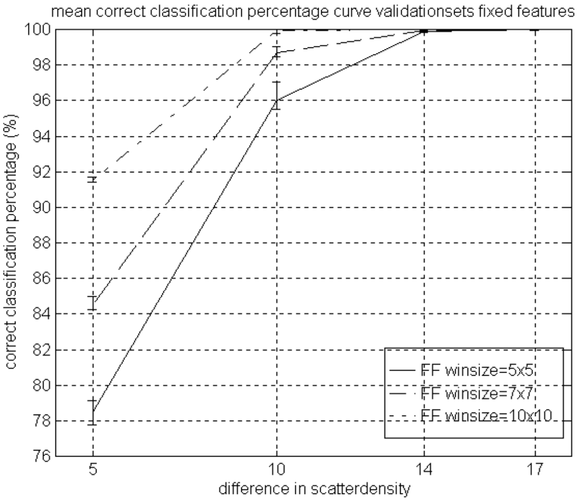


Figure 7.4 Mean correct classification curve validation set fixed features.

Looking at the results shown in figure 7.3 and figure 7.4, the following statements can be made:

1. Slight differences can be found between the results of the training- and validation sets. Actually there should be no differences because fixed features are not trained on the training set. This minor difference can therefore only be explained as a slight difference in the datasets of the training- and validation sets. The following statements are referring to the validation set.
2. At a decreasing difference in scatter density a trend can be discovered in the decrease of the correct classification percentage. The explanation is that the difference between the two classes decreases.
3. Besides that, it appears that a better correct classification percentage can be obtained at an increasing window size. The explanation is that features can be estimated more accurate by a window that contains more data. This is possible because the standard deviation decreases when the window size increases, thereby increasing the ability to distinguish the features.
4. Finally, it appears that the difference between the correct classification percentages of the used window sizes increases at a decreasing difference in scatter density. This is caused by the non linear relationship between the Mahalanobis distance and the correct classification percentage. The possibility exists that a dataset has a Mahalanobis distance of, for instance, 20 and another dataset has a Mahalanobis distance of, for instance, 10, but both have a mean correct classification percentage of approximately 99 %.

7.6 Experiments with self learning image features

7.6.1 Experiment 1: variation #bins and window size

For each dataset in this experiment a mean correct classification percentage of the SLF is estimated for every window size and each #bins. The datasets used in this experiment are shown in table 7.2 and the #bins will vary from 5 up to and including 10. Other items that will vary in this experiment are the network size and the window size. For the network size an architecture of #bins - (2 * #bins) - 2 will be used. The windows that will be used have a size of 5x5, 7x7 and 10x10 speckle's. Thereby doubling the #speckle's in a window when the window size increases.

Now that all parameters are known for this experiment the correct classification percentage for the SLF can be estimated. Because this experiment has a three dimensional problem the results will be spread over several graphs. The results are shown in figure 7.5 up to and including figure 7.10. For each mean correct classification percentage its minimum and maximum values are also given. These are represented by the two short horizontal lines connected by a vertical line.

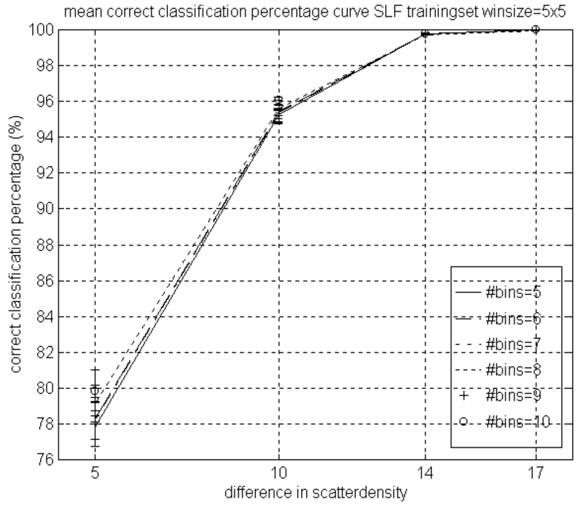


Figure 7.5 Results SLF training set window size: 5x5 speckles.