A neural network edge detector

L.J. Spreeuwers

University of Twente, Department of Electrical Engineering
7500 AE Enschede, Netherlands.

ABSTRACT

Extracting edges from images is a widely used first step in image processing. A different view on the well known enhancement/thresholding approach for edge detection is presented in this paper. The structure of a two layer feed forward neural network is comparable to the structure of enhancement/thresholding edge detectors. It is possible to calculate an optimal edge detector with a certain predefined network structure and training set, by training the neural network with examples of edge and non-edge patterns. The back propagation learning rule is used for optimization of the network. The choice of the network structure and the training set are very important, because they determine the final behaviour of the network. The paper describes which network structures were selected and how the training sets were generated. Some of the experiments are described, and observations of the convolution kernels for edge enhancement that are formed during training. Finally the results are evaluated and compared with the results of edge detectors based on the Sobel, Marr-Hildreth and Canny edge enhancement algorithms. It appears that the neural network edge detector can be made very robust against noise and blur and in most tests outperforms the others.

1. INTRODUCTION

An edge can be defined as the border between regions with different properties in an image. These properties can be texture, grey levels, colours etc. In this report only borders between regions with different grey levels will be considered. In this case edge detection is a process for finding grey level changes in images. However, not all grey level changes correspond to edges. As an example, consider noise in an image, or a gradually changing level originated by sideways illumination of a plane. Furthermore in degraded images the exact localization of the edges becomes a problem. Depending on the type of scene, an image may contain different types of edges, e.g. straight or curved edges, steep edges or edges with a gradual slope etc.

Design of edge detectors involves three important stages. First a model of the edges to detect is required. The second stage is to choose and design an optimal operator to detect the patterns described by the edge model. The third stage is to evaluate the designed operator.

To find an appropriate mathematical description of edges appears to be very difficult. This complicates the systematic design of edge detectors. In recent years some interesting results were obtained, using one dimensional edge descriptions to design an optimal one dimensional edge detector.[1,2] However all of these used a rather questionable extension to the second dimension. Dickey and Shanmugham simply rotated their 1d operator to obtain a directional independent 2d operator. Canny created multiple directional dependent edge operators by convolving his 1d linear edge detection function, alligned normal to the edge direction, with a projection function parallel to the edge direction.

Neural networks are trained using examples of the desired input-output behaviour. A neural network may be trained to detect edges by providing it with examples of edge patterns and not-edge patterns. The advantage of using examples is that it does not require a strict mathematical description of edges. Edges can be described in a more relaxed way by a set of example patterns. Very often it is easier to generate a representative set of examples, than a precise mathematical description of a process. This seems to be the case with edge description too, at least for the two dimensional case. One could say that the edge model is implicitly present in the training data. An other advantage of using training data to describe edges is, that it can be easily adapted to a specific type of scene. The edge detection operator architecture proposed here is a layered neural network. The operator is optimized using the back propagation learning rule. Because the training set contains two dimensional example patterns, this is a true 2d optimized edge detector. In order to be able to analyze its

internal behaviour, an architecture for the network is adopted, that resembles the so called enhancement/thresholding approach, as described by e.g. Abdou and Pratt.[3]

For the evaluation of the neural edge detectors, an evaluation method is adopted that was proposed by F. van der Heyden.[6]

The enhancement/thresholding approach to edge detection is described in section 2. In section 3 the generation of training sets is described. Section 4 describes the evaluation method briefly. Sections 5 and 6 report on the experiments and results. A summary of the conclusions is given in section 7.

2. ENHANCEMENT/THRESHOLDING EDGE DETECTORS

In this section the enhancement/thresholding approach to edge detection is described. For a few well known edge detection techniques it is shown how they fit into the enhancement/thresholding scheme. It is shown that a two layer back propagation neural network also fits into the scheme.

2.1 The enhancement/thresholding scheme

In the enhancement/thresholding approach,[3] edge detection is divided into two processes. The first process is a feature enhancement process. A bank of filters is used to select and enhance certain features in the input signal. The input data will generally consist of the grey levels of the neighbouring pixels of the pixel to classify. The resulting features are combined into a single number by a so called point operator. The resulting number is often called the likelihood of the pixel being an edge. The second process is the actual classification. The classification is accomplished by comparing the likelihood with a threshold. The enhancement/thresholding scheme is illustrated in fig. 2.1.

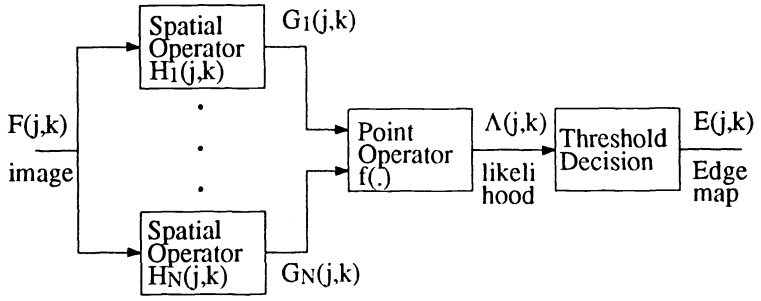

fig.2.1 Edge enhancement/ thresholding edge detection system

2.2 Sobel edge enhancement filters

As an example of an enhancement/thresholding edge detection operator, consider an edge detector, based on the Sobel edge operator.[4] In this case there are two filters for edge enhancement. The filters calculate the gradient of grey levels in the horizontal and vertical directions. The outputs of the filters are combined using one of the following 'point' operators:

$$\Lambda(i,j) = |G_1(i,j)| + |G_2(i,j)| \quad \text{or:} \quad \Lambda(i,j) = \sqrt{G_1(i,j)^2 + G_2(i,j)^2}$$

Finally the output is thresholded.

2.3 Bayes classifier

An other example of an edge operator that can be mapped on the enhancement/thresholding scheme is the classifier based on classical statistic theory and normal conditional probability distribution functions. It is described here to show that the enhancement/thresholding scheme is quite general and also because the term likelihood in the scheme is descended from the *likelihood ratio* of a two class bayes classifier. Only the outline of the derivation of the Bayes edge classifier will be given.

The likelihood ratio is defined as the ratio of the conditional probability of a measurement vector being of the first class and it being of the second class:

$$L(\boldsymbol{x}) = \frac{p(\hat{\omega}_1 | \boldsymbol{x})}{p(\hat{\omega}_0 | \boldsymbol{x})} \tag{3.1}$$

The grey levels of a local neighborhood of a pixel will be ordered in an N dimensional measurement vector $\boldsymbol{x}$. Normal conditional probability distribution functions for $\boldsymbol{x}$ being an edge (ω_0) and for $\boldsymbol{x}$ not being an edge (ω_1) are asumed:

$$p(\hat{\omega}_0 | \boldsymbol{x}) = \frac{1}{(2\pi)^{1/2} |C_0|^{1/2}} \exp[-\frac{1}{2} (\boldsymbol{x}-\mu_0) C_0^{-1} (\boldsymbol{x}-\mu_0)^t] \tag{3.2a}$$

$$p(\hat{\omega}_1 | \boldsymbol{x}) = \frac{1}{(2\pi)^{1/2} |C_1|^{1/2}} \exp[-\frac{1}{2} (\boldsymbol{x}-\mu_1) C_1^{-1} (\boldsymbol{x}-\mu_1)^t] \tag{3.2b}$$

Where C_0 and C_1 are the covariance matrices and μ_0 and μ_1 are the mean vectors of $\boldsymbol{x}$.

An important property of the likelihood ratio of normal conditional distribution functions, is that is insensitive to linear transformation of the vector $\boldsymbol{x}$. A linear transformation can be found that simultaniously decomposes both conditional distributions, so that the covariance matrices of the transformed measurement vector are diagonal matrices (i.e. all elements of the vector are independent). If furthermore the logarithm is taken of the likelihood, this results in a simple expression for the log likelihood:

$$\Lambda(\boldsymbol{x}) = \log[L(\boldsymbol{x})] = K + \frac{1}{2} \sum_j [\frac{1}{\gamma_{jj}} (y_j - \mu_{0yj})^2 - (y_j - \mu_{1yj})^2] \tag{3.2}$$

Where K is a constant, y_j are the elements of the transformed measurement vector $\boldsymbol{y}$, γ_{jj} are the diagonal elements of the covariance matrix of the conditional distribution for ω_1 and μ_{0y} and μ_{1y} are the mean vectors of the conditional distributions of $\boldsymbol{y}$.

The linear transformation from the measurement vector $\boldsymbol{x}$ into the feature vector $\boldsymbol{y}$ can be accomplished via a bank of linear filters. The elements of the feature vector $\boldsymbol{y}$ can be combined into a log likelihood ratio, using expression (3.2) as a point operator. Finally the classification is done by thresholding the log likelihood ratio. If the log likelihood ratio is above the threshold the pixel is classified as an edge and otherwise as not-edge.

2.4 Back propagation neural network as an enhancement/thresholding edge detector

The units in a neural network, that applies the standard back propagation learning rule, perform a weighted sum of their inputs, and a nonlinear operation (mostly a sigmoid function) on this sum, see fig. 2.2.

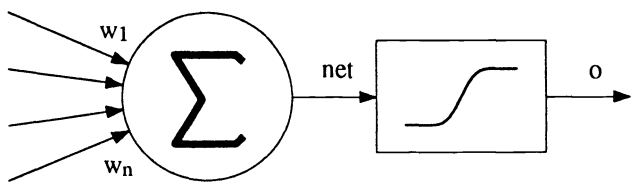

fig.2.2 Unit of a standard back propagation neural network

A back propagation neural network consists of a number of these units ordered in layers. The units in a certain layer may only project their outputs to higher layers, i.e. only feed forward connections are allowed. A simple two layer network fits into the described enhancement/thresholding scheme, as can be easily seen from fig. 2.3. The units in the first layer of the network act as the feature enhancement filters. The single unit in the second layer performs the 'point' operation to combine the outputs of the first layer. A description of the back propagation learning rule, that is used to optimize both the enhancement filters and the point operator, can be found in Rumelhart[5] and Spreeuwers.[7]

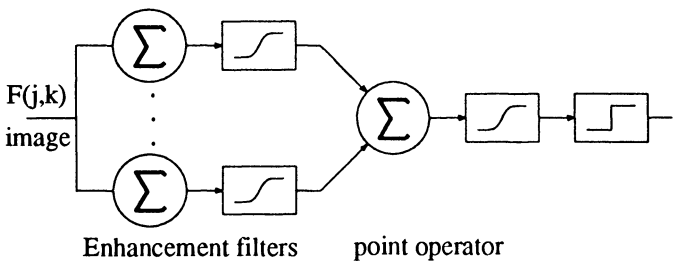

fig.2.3 A back propagation neural network as an enhancement/thresholding edge detector

The enhancement/thresholding scheme appears to be a fairly general structure for different edge detectors. The advantage of viewing all these operators as a variant of the same general architecture is that it becomes possible, not only to compare the performance of different edge detectors, but also their operation (e.g. the feature enhancement filters).

3. Generating training sets

Training a back propagation neural network for the task of edge detection means providing it with examples of edge patterns and not-edge patterns. A set of examples is called a training set. Each sample of a training set consists of a measurement vector and its class label (i.e. edge or not-edge). For edge detection, the measurement vector generally consists of the grey levels of pixels in a local neighborhood of the pixel to be classified. As a rule a square neighborhood of n*n pixels, with n odd, is used. The pixel in the center being the pixel that must be classified.

3.1 Edge properties

A training set must be representative for the application area of the edge detector. This means that all types of edges that may occur in the type of scenes, the edge detector is designed for, should be present in the training set. Furthermore the frequency of a certain type of edge in the training set should resemble its frequency in real scenes. Some important properties of edges that should be considered in connection with generating artificial images are listed below and illustrated in fig.3.1. This list is not a complete list of all possible edge properties. It is however a reasonable starting point to generate artificial images with edges that resemble edges in real world scenes.

1. orientation	- normally all orientations appear in a scene, and therefore should be present in a training set
2. curvature	- particular scenes may contain mainly curved or straight edges
3. connectedness	- an edge always consists of a number of connected pixels on the border of two regions. An edge can never be an isolated pixel
4. closedness	- most times the border or boundary of a region is defined by a closed line
5. shape	- depending on the type of scene there may be edges with different slopes, e.g. step edges, roof egdes and ramp edges
6. height,base,width	- generally edges with different height, width and ground level (base) will occur in an image

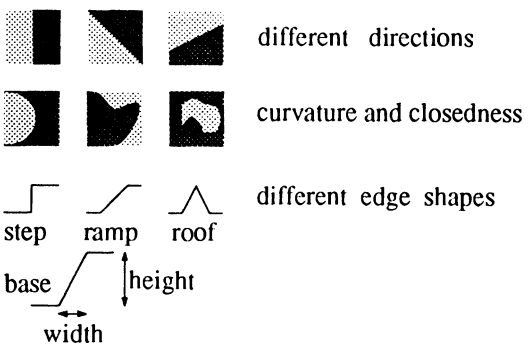

fig.3.1 Edge properties. Above: orientation, curvature, connectedness and closedness; below: (cross section of image) edge shape, base, width, height

3.2 The image generator

The image generator used here was proposed earlier by F. v.d. Heyden[6] for the design of edge detectors based on a statistical theory and for edge detector evaluation. It generates images with small islands with different grey levels using a random process. The edge density as well as the variance of the edge height can be controlled. The mean of the edge height is fixed. All edges have a step shape. No overlapping islands are generated. The image generator also generates an edge reference map with the exact location of all the edges. In fig. 3.2 an example of a training image and the corresponding edge reference map is shown. Clearly all contours are closed. Edges of all directions and different curvature are present in the image. The working of the image generator will be described briefly below.

An image with white noise is low pass filtered with a gaussian point spread function (psf) with standard deviation σ_w.The resulting image is thresholded. This results in a binary image with clearly separated segments. The distance between edges depends on σ_w. From each of the segments the inner boundary is detected. This inner boundary is later used in an interpolation process to increase the localization accuracy of the edges. A grey level image is obtained by assigning a randomly chosen grey level to each segment, using a random number generator with gaussian distribution function with variance σ_f and mean 75 or 175, depending on the level of the segment of the binary image. The next step is the interpolation process that uses the detected inner boundaries to move some of the intensity changes in accordance with the original low pass filtered noise image. Finally the edge reference map is obtained by using a level crossing operator on this grey level image.

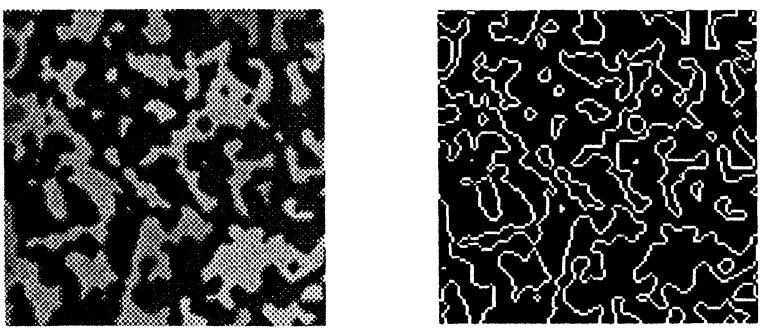

fig.3.2 Training image and edge reference map generated by the image generator.

The generated images are not representative for any type of real scene (however they could be used to model e.g. raindrops on a window or oil stains on the sea). The main reason to use this image generator was that it was already available. Later other image generators were developed, but the results are not described in this report.

4. Evaluation

4.1 Edge detector evaluation methods - overview

In order to determine and compare the quality of edge detectors, an objective measure is required. This means the quality should be expressed in a number (or numbers). Generally this number is called a FOM (figure of merit). In literature several evaluation methods can be found e.g. Abdou and Pratt[3], v.d. Heyden[6], Peli and Malah[8], Kitchen and Rosenfeld[9] and Fram and Deutsch.[10] The evaluation method of Abdou and Pratt is referenced most often. It measures the localization error of an edge detector on a test image that consists of a single straight horizontal, vertical or diagonal edge of ramp shape with added gaussian noise. Since the test image is fixed, this evaluation method cannot be applied to measure edge detector quality for different types of scenes. Most other evaluation methods also use fixed test images. For example Kitchen and Rosenfeld[9] use images with a single straight vertical edge and images with circles to measure edge connectedness. Only the evaluation method proposed by v.d. Heyden accepts arbitrary test images. Furthermore it measures several errors, like shifts, missed edge pixels and edge thickness. Therefore this evaluation method was chosen to evaluate the neural edge detector. The method is briefly explained in the next paragraph. For a detailed description refer to v.d. Heyden.[6]

4.2 Edge detector evaluation method according to v.d. Heyden

The evaluation method requires an edge map, generated by the edge detector to evaluate, and a corresponding edge reference map, with the exact positions of all the edges. Both images are binary with values 0 for not-edge and 1 for edge. The edge map is subtracted from the edge reference map. The result is a three valued image with values 0 for correctly classified pixels, -1 for false alarm and 1 for missed edge pixels. A simple FOM could be obtained by counting the number of false alarms and missed edges. This would however result in very bad FOM's (i.e. very high FOM's) if the edge detector has imperfect localization. Slightly shifted edges cause two errors: false alarms on their own position, and missed edges on the real edge positions in the edge reference map. The object of the evaluation is to find out how close the detected edge is to the real edge. In order to obtain a better FOM that reflects this object, the difference image is low pass filtered with a gaussian psf with width σ, and the absolute value of the result is raised to the power of β. As a result displacement errors are cancelled to an extent in accordance to the rate of displacement. This effect is influenced by the parameter sigma. Also clustered errors are penalized heavier than isolated errors. This effect is influenced by both parameters sigma and beta. To obtain a FOM that is independent of the size of the test image the mean value of the levels of all pixels is taken:

$$\mathrm{FOM}(\sigma, \beta) = \frac{1}{NM} \sum_{j=1}^{N} \sum_{k=1}^{M} |g(j,k)|^{\beta} \qquad (4.1)$$

Where the image has size NxM pixels, and g(j,k) is the low pass filtered difference image. A block scheme of the evaluation method is depicted in fig. 4.1.

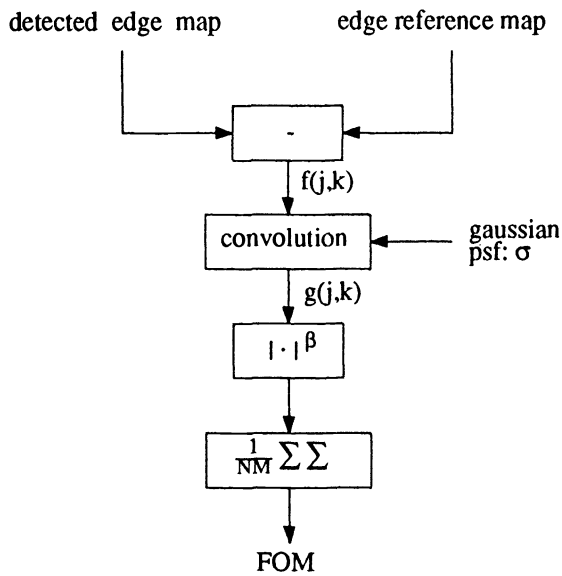

fig.4.1 Edge detector evaluation method as proposed by v.d. Heyden.

V.d. Heyden also investigated the influence of σ and β on different types of errors. Six types of errors can be distinguished, each of which is weighted with a weight factor w. For different values of σ and β different sets of weights will occur. The error types are:

1. misclassified isolated pixels (w_1)
2. misclassified pixels in a line segment (w_2)
3. displacement of a line segment over distance α ($w_{3,\alpha}$)
4. contour segments with thickness 2 (w_4)
5. contour segments with thickness 3 (w_5)
6. highly clustered misclassified pixels (w_6)

Because it appeared that the evaluation method weights the edge thickness rather heavy, no matter the choice of σ and β, before evaluation a localmax operation was used. The localmax operation determines the local maximum of the grey levels in certain direction in a small area and thereby guarantees a 1 pixel thick edge after thresholding. It was applied in the horizontal and vertical directions.

5. Experiments

The experiments aim to support the supposition that an edge model can be defined using a training set, and that a back propagation neural network can be used to find an optimal edge detector. A hypothetical type of scene is defined by the image generator described in section 3. Networks of different size and several different learning parameters are to be tested, and training with clean and disturbed images.

5.1 Description of experiments

In order to investigate the training of back propagation neural networks for the task of edge detection, networks of different size, clean and disturbed test images and several settings of the learning parameters were tested. The problem with this kind of experimental work is that one cannot investigate exhaustively. Only a rough impression of the influence of different parameters can be obtained. The following experiments were setup:

Architecture: as described in section 2, to fit in the enhancement/thresholding scheme, the networks should have 2 adaptive layers. The first one performs edge enhancement, the second is the point operator that combines the features into the likelihood number. The second layer consists of a single unit. This means the number of units in the first layer must be chosen and also the number of inputs, i.e. the size of the measurement vector (or local neighbourhood). To investigate this, the following architectures were tested (a network architecture is described as: <nr.of inputs>-<nr. of units in first layer>-<nr. of units in second layer>):

size of local neighbourhood:	size of network:
1. 9-8-1	1. 25-8-1
2. 25-8-1	2. 25-16-1
3. 49-8-1	3. 25-8-4-1

Learning parameters: the learning parameters of the back propagation learning rule are the learn rate (η) and the momentum (α). The learn rate determines the step size for the weight updates. The momentum term determines how much of the previous weight update is used in the new weight update. The momentum is actually not used in the basic back propagation learning rule. To limit the number of experiments, we have set the momentum to 0. Rumelhart et al[5] suggest a value between 0.1 and 0.75 for the learn rate. We have tested values of:

$\eta = 0.01,\ 0.1,\ 1.0$

Clean and disturbed test images: the test image generator generates ideal images, without noise and with perfectly sharp edges. In reality, images acquired with a camera may be noisy and blurred somewhat. To simulate this, a test image was low pass filtered with a gaussian psf and gaussian noise was added. It was decided that the edge detector should be able to handle different amounts of noise and blur. Therefore a test image was designed using the following standard deviations for the additive gaussian noise (σ_n) and psf of the low pass filter (σ_{psf}):

σ_n : 0, 5, 10, 20
σ_{psf}: 0, 0.5, 1.0, 2.0

This results in 16 sub images with different combinations of noise and blur. The resulting image is shown in fig. 5.1. The test images used to train the networks were generated with $\sigma_w = 2$ and $\sigma_f = 50$ for the low pass filter and the level assignments process of the image generator.

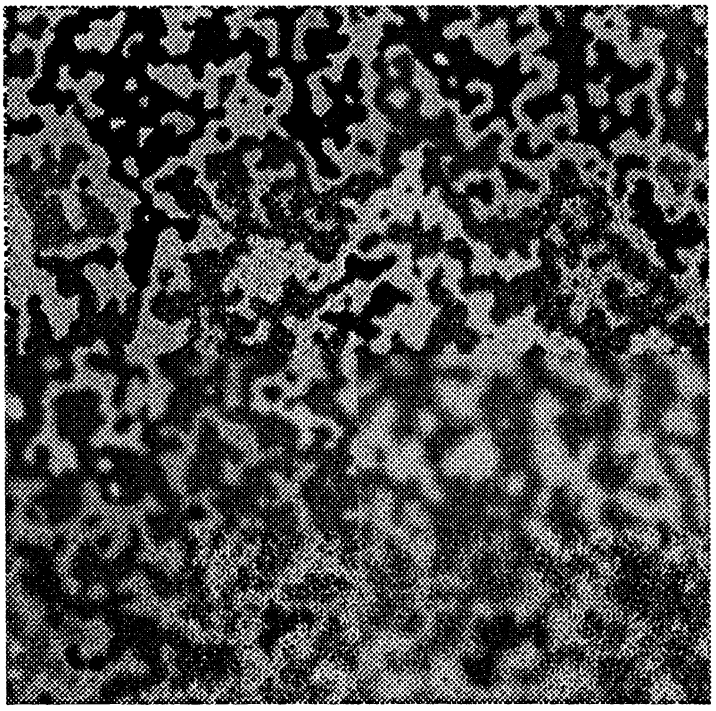

fig.5.1 Training image with different combinations of blur and noise: σ_{psf} = 0, 0.5, 1.0, 2.0 for quarter images from top left to bottom right; σ_n = 0, 5, 10, 20 for each 16th of an image in a quarter image, again from top left to bottom right.

5.2 Measurements

Apart from visual inspection of the edge operator results, the following measurements were performed:

Convergence: in order to evaluate the neural edge operators, we first need to know if the optimization by means of the back propagation learning rule was successful. An error measure must be defined and registered during learning. The error measure chosen here is the one defined by Rumelhart et al.[5] For our one output network it reduces to:

$$E = \frac{1}{N}\sum_{j=1}^{N} |y_j - d_j| \qquad (5.1)$$

Where y_j is the output of the network, d_j is the desired output, which is 1 if the pixel to be classified is an edge pixel and 0 otherwise. N is the number of training samples in the training set. The error is accumulated over all training samples. At the end of one cycle (epoch) through the training set it is recorded as one point in a graph. The graph supplies information concerning the convergence. It is to be expected that the error decreases approximately as a negative exponential function.

Evaluation using fom: second in the evaluation of the neural edge operators is their mutual comparison and comparison to other edge detectors. This is done using the fom described in section 4. The parameters sigma and beta of the fom are set to σ=2 and β=1.2. This results in the following weighting factors for the different errors:

misclassified isolated pixels	w_1 =0.44
misclassified pixels in line segment	w_2 =0.66
displacement of line segment over 1 pixel	w_{31}=0.45
displacement of line segment over 2 pixels	w_{32}=0.70
displacement of line segment over 3 pixels	w_{33}=0.88
contour segment with thickness 2	w_4 = 0.66
contour segment with thickness 3	w_5=1.48
highly clustered misclassifications	w_6=1.00

The edge detectors that were used to compare with are: Sobel,[4] Marr-Hildreth[11] and Canny.[2] The implementation of the Marr-Hildreth that was used uses gradient information to determine edge strength, and and the zero crossings of the $\nabla^2 G$ operator to determine the exact position of the edges (this gives much better results than using the $\nabla^2 G$ operator only). In all cases the resulting edge maps were post processed using a localmax operator to obtain one pixel thick edges. Fom graphs were constructed for blur with σ_{psf} 0.2 and 1.0, and snr's ranging from 0 to 3.5. For each edge operator, the threshold is selected that gives the best fom.

Examination of the obtained filters: finally the obtained filters can be examined by displaying the weights of the units of the first layer (the enhancement layer) as grey levels. As an example the convolution kernels of a sobel edge operator are shown below. The sobel edge operator[4] is defined as:

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, \text{for vertical edges and} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}, \text{for enhancement of horizonal edges}$$

The grey level representation is shown in fig. 5.2.

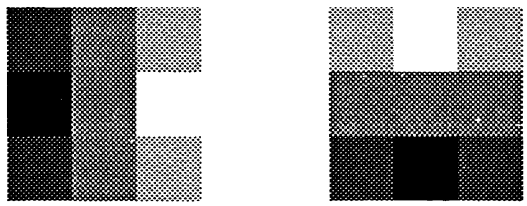

fig.5.2 Grey level representation for sobel edge operator edge enhancement filters.

6. Results

6.1 Learning curves.

The learning curves show how well the training set is learned. If E=0, the network is able to classify the training set perfectly. E=0.1 means that the mean distance of the output of the network to the correct value (i.e. 0 for not edge and 1 for edge) is 0.1. The conclusions are described below. The learning curves for training with disturbed training image and clean training image are given in fig.6.1. The learning curves of the other experiments (all using the disturbed training image) are very similar to the upper curve in fig.6.1.

A high learning rate results in faster learning. A very high learning rate (η=1.0) results in an irragular curve and slightly worse performance. A larger local neighbourhood improves performance, and speeds up learning, but using larger networks than the 25-8-1 network (i.e. more hidden neurons and or layers) does not seem to improve performance or learning speed much. Finally, training with a clean image results in a very low final error. Thus a network trained with a clean image should be able to classify a noise and blur free image almost error free. However it is to be expected that it will perform worse on a disturbed image, compared to a network trained on a disturbed image. The experiments confirmed this.

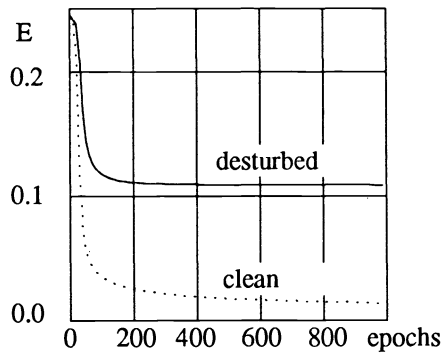

fig.6.1 Learning curves for training with disturbed training image (fig.5.1) and clean training image. The upper curve is typical for all other tests.

6.2 Fom graphs

For evaluation of the edge operators different test images were used than for training the networks, however they were generated using the same image generator and with the same statistics . To construct the fom graphs, the fom's for the edge operators were determined for the test image with additive white noise and blur. Two values were chosen for the standard deviation of a gaussian psf of the blur: σ_{psf} =0.2 and σ_{psf} =1.0. Snr ratio was varied from 0 to 3.5.

An important conclusion from the comparison between the learn curves and the fom plots was that a better final result for the error E does not always imply a better fom. Comparison of the fom graphs of the different neural network edge operators showed that the local neighbourhood size had the largest impact on the final result. The 49-8-1 network performed much better for low snr and high blur. Training with the disturbed training image resulted in a more robust edge operator. A high learning rate (η=1.0) resulted in slightly worse performance.

Fig. 6.2 shows the comparison of a 49-8-1 neural network edge operator to the Sobel, Marr-Hildreth and Canny operator. The Marr-Hildreth operator is the improved version that uses the gradients too, as described in section 5.2. It appears that, for blur=0.2, the neural network performs best upto a snr of approximately ^{10}log(snr) = 1.5, which implies for mean signal level 100 a noise level of approx. 18. This is very close to the maximum noise level used in the training image (σ_n=20). Thus for the range of snr's the network is trained for, it outperforms all others. For blur=1.0, the network outperforms the other edge detector operators for all snr's. It can also be observed that the improved Marr-Hildreth is slightly more accurate than the Canny operator for high snr, while the Canny operator is the best at low snr. For all snr's the Sobel operator performs worst.

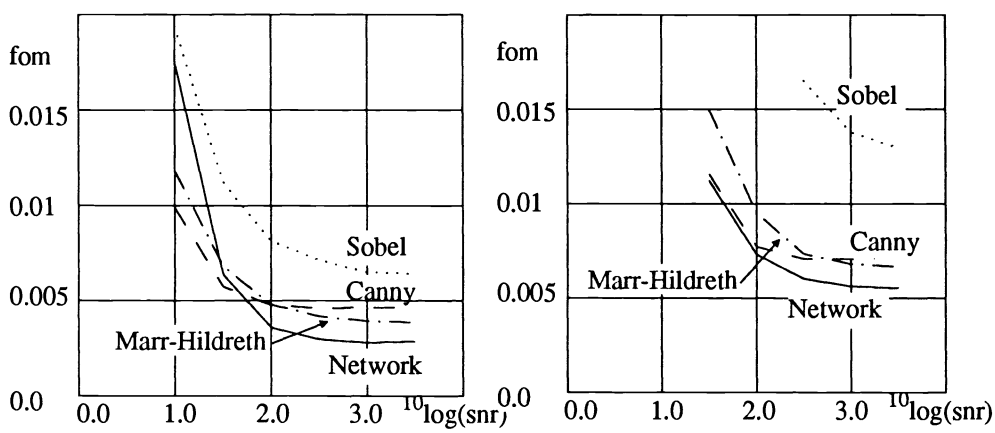

fig.6.2 Evaluation of edge operators using a FOM; Comparison of performance of the 49-8-1 neural network edge operator with Sobel, Marr-Hildreth and Canny operators. Left: test image blur is 0.2, right: blur is 1.0.

6.3 Weight kernels

In fig.6.3 the weights of the units in the first layers of the 49-8-1 network are displayed.

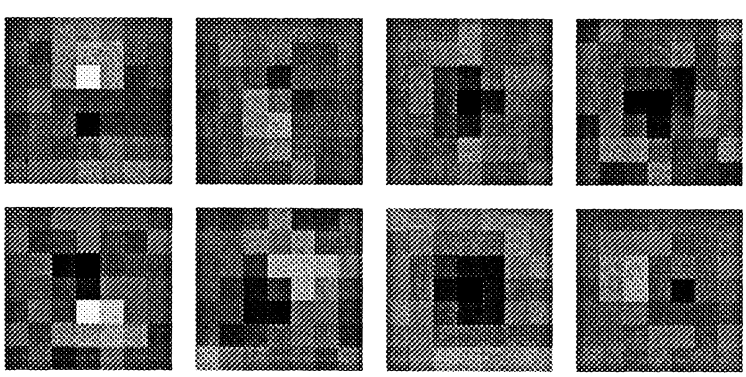

fig.6.3 Graphical representation of the weights of the first layer of a 49-8-1 neural network edge operator.

As described in section 2, the units in the first layer can be considered as convolution filters. The filters do not show very clear patterns. From the illustrations some of the kernels can be recognized as directionally dependent gradient like filters. Some of the kernels seem to do a low pass filter operation, and some act more or less as symmetrical high pass filters. It is not easy to interpret the exact behaviour of the neural network edge detector operator from the weight kernels.

6.4 Results on images

Fig.6.4 shows the results of the thresholded output of the Sobel, Marr-Hildreth, Canny and 49-8-1 neural network edge operators and the corresponding edge reference map. The output of the operators is postprocessed using a localmax operator. The test image was generated with the image generator described in section 3, with blur 1.0 and additive gaussian noise with σ_n=10.0.

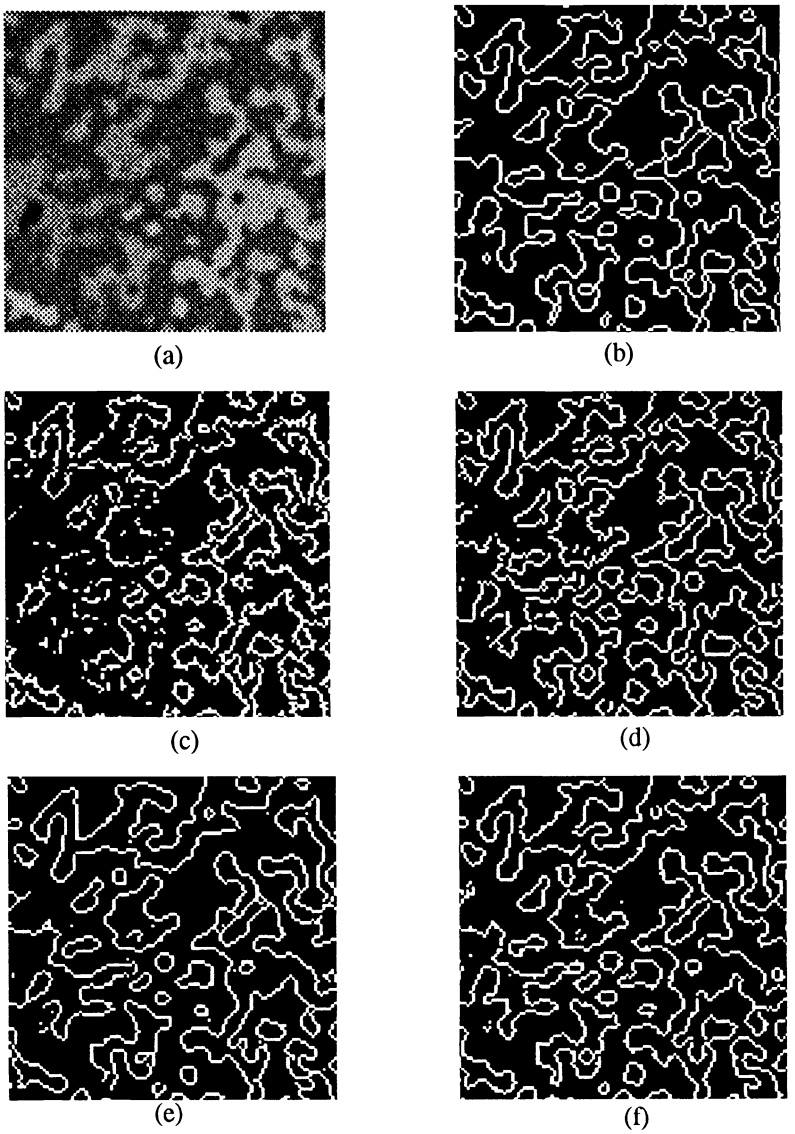

fig.6.4 Resulting edge maps for several different edge detectors. From top left to bottom right: (a) grey level test image; (b) edge reference map; (c) Sobel edge detector; (d) Marr-Hildreth edge detector; (e) Canny edge detector; (f) 49-8-1 neural network edge detector

It is quite clear that the Sobel operator produces unacceptable results, because it does not suppress noise. The other three operators give quite acceptable results. The 49-8-1 neural network edge operator is somewhat more accurate than both the

Marr-Hildreth and the Canny edge operators. Furthermore the neural network edge detector operator supresses the background better than the other operators. Therefore the final result is less sensitive for the choice of the threshold.

7. Conclusions

The aim of this project was to investigate if a back propagation neural network can be used for edge detection. The enhancement/thresholding edge detector structure is used as a general framework of the used edge detectors. This enables us to compare the network edge detector operator to other edge operators. Several networks were trained for this task and the results were evaluated using a figure of merit and compared to other edge detectors. The networks were trained using a labeled training set of edge pattern examples.

It appeared that back propagation neural networks can be used for edge detection quite successfully. A network can be trained using a grey level test image with a corresponding edge reference map. Thus it is relatively easy to train a network for a specific type of edge if the edge reference map is available. Training is best done with a low learning rate. This slows down the learning process, but gives the best results. Training with a disturbed training image results in more robust edge operators. The interpretation of the internal behaviour of the network was a difficult point. The "enhancement filters" of the neural network are rather irregular and therefore it is not easy to interpret its exact internal behaviour. Some of the filters seem to act as direction dependent gradient operators and others as direction independent, symmetrical high pass filters. Thus the approach of the enhancement/thresholding structure appeared not to help very much in understanding the internal behaviour of the networks. The approach appeared however to be successful for designing edge detector operators. Several different network sizes were tested and a 49-8-1 network (7x7 local neighbourhood 8 units in a hidden layer) appeared to be a good choice for the network architecture. The 49-8-1 network outperforms Canny, Marr-Hildreth and Sobel edge detector operators for the type of edges it is trained for.

References

1. Shanmugam, K.S., Dickey, F.M., Green, J.A., An Optimal Frequency Domain Filter for Edge Detection in Digital Pictures, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.PAMI-1(1), pp.37-49, 1979
2. Canny, J., A Computational Approach to Edge Detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.PAMI-8(6), pp.679-698, 1986
3. Abdou, I.E., Pratt, W.K., Quantitative Design and Evaluation of Enhancement/ Thresholding Edge Detectors, *Proceedings of the IEEE*, vol.67(5), pp.735-736, 1979
4. Pratt, W.K., *Digital Image Processing*, Wiley, New York, 1978
5. Rumelhart, D.E., Hinton, G.E., Williams, R.J., Learning representations by back-propagating errors, *Nature*, vol.323, 1986
6. Heyden, F. van der, Quality of Edge Detectors, *Proceedings of CSN89, SION, Utrecht, Netherlands,1989*
7. Spreeuwers, L.J., Mathematics of the Backpropagation Learning Rule, Part 1: Basic Mechanisms, *Department of Electrical Engineering, University of Twente, Netherlands, BSC Report* 90M132, 1990
8. Peli, T, Malah, D., A Study of Edge Detection Algorithms, *Computer Graphics and Image Processing*, vol.20, pp.1-21, 1982
9. Kitchen, L., Rosenfeld, A., Edge Evaluation Using Local Edge Coherence, *IEEE Transactions on Systems, Man, and Cybernetics*, vol.SMC-11(9), pp.597-605, 1981
10. Fram, J.R, Deutsch, E.S., On the Quantitative Evaluation of Edge Detection Schemes and their Comparison with Human Performance, *IEEE Transactions on Cumputers*, vol.C-24(6), pp.616-627, 1975
11. Marr, D., Hildreth, E., Theory of Edge Detection, *Proceedings of the Royal Society London B*, vol.207, 187-217, 1980