A feed-forward neural network approach to edge detection

L.X. Zhou & W.K. Gu
Department of Information Science & Electronic Engineering, Zhejiang University, Hangzhou 310027, P.R. China
EMail: zhoulx@isee.zju.edu.cn

Abstract

This paper presents a novel edge detector based on Feed-Forward Neural Networks (FFNNs). The FFNN computing architecture has two stages, which is a feature enhancement stage as well as a structural boundary extraction stage. The first stage is a traditional supervised BP network, and the second one is manually designed without training. Experiments based on both synthetic and natural images show that the FFNN edge detector can produce accurate, continuous, and smooth edge chains.

1 Introduction

Edge detection is one of the most essential problems in computer vision as it is the common starting step for segmentation or feature detection. In natural images an edge is usually difficult to define precisely due to the perceptual component which is associated. In recent years some strategy based on neural networks have been exploited for selecting edge points from image directly. For example, Grossberg and Mingolla [1] employ a competition-cooperation network, the Boundary Contour System (BCS), to simulate mammalian vision. Pinho and Almeida [2] present a feed-forward network for edge detection which believe that the nonlinear

mapping between the source gray-level pixels and the desired edge elements can be trained and stored in a simple FFNN. Intajag and Paithoonwatanakij [3] apply a fuzzy neural network for edge detection from local features. This network has a four-layer feed-forward architecture. Lu and Shen [4] propose a hybrid network, which extracting boundary by a BP net as well as enhance boundary by a modified Hopfield neural network. Vrabel [5] treats edge detection as a problem in optimization, which can be solved with a recurrent Hopfield neural network.

Among the network types used above, we prefer the feed-forward neural networks (FFNNs). The choice of FFNN is motivated at least by the following two reasons: firstly, FFNN is very fast during the working procedure; and secondly, FFNN is also the best for hardware implementation.

This paper addresses the problem of extracting single-line edges in natural images by a feed-forward neural computing architecture. Our proposal is that an edge detection can be decomposed to two separate steps, edge enhancement and structural boundary extraction, respectively. In the first step, the possibility of a pixel being an edge element is obtained and normalized; in the second step, edge elements are linked to single-line, continuous, and smooth boundary chains with local structural information. Both steps are implemented by FFNNs.

2 Edge enhancement

According to the study of neurophysiology, the cerebral visual cortex cells are functionally cooperating [6]. For visual processing tasks, those are the simple cells and the complex cells. The former is strictly sensitive to the orientation and amplitude of local illumination, and the latter organizes the outputs into reasonable high-level perceptions. This phenomenon infers that edge detection can be done by a two-stage network, one for feature extraction and one for post-processing.

In this paper we provide two-stage FFNNs for edge detection. The system architecture is illustrated in Figure 1. The gray-level image is feed into the first edge enhancement network. This network produces local edge attributes (normalized image gradient amplitude and orientation) and feed them into the boundary extraction network. The latter generates final binary edge structures. In this section we discuss the first FFNN, and the second one will be detailed in next section.

For the edge enhancement network, the main aim is to calculate the probability of a pixel being an edge element, as well as to determine local

edge orientation. Traditionally the most common approach used to assess such properties are directional Sobel operators, as shown in Figure 2. These eight masks (only four of them are shown in Figure 2 discarding the symmetric ones) simulate the orientation-sensitive receptive fields in visual cortex. They provide local image gradient $g(r, c)$, where r and c are image row and column numbers, respectively.

As we known, the perception of an image edge depends on a reference threshold, th. If the gradient $g(r, c)$ is much more than th, there will be a very high probability for $g(r, c)$ being an edge element. If $g(r, c)$ is very small, the probability will be close to zero. We define the following formula to translate the gradient amplitude into such a probability:

$$p(r,c) = \begin{cases} \left[\frac{g(r,c)}{th}\right]^2 & g(r,c) \le \frac{th}{2} \\ 0.5 - \left[1 - \frac{g(r,c)}{th}\right]^2 & \frac{th}{2} < g(r,c) \le th \\ 0.5 + 0.5 \cdot \frac{g(r,c) - th}{255 - th} & g(r,c) > th \end{cases} . \quad (1)$$

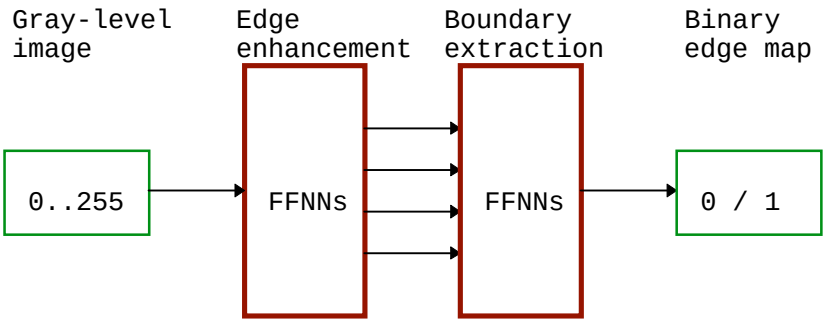


Figure 1: Edge detection using FFNNs: system architecture

-1	0	+1
-2	0	+2
-1	0	+1

-2	-1	0
-1	0	+1
0	+1	+2

-1	-2	-1
0	0	0
+1	+2	+1

0	-1	-2
+1	0	-1
+2	+1	0

Figure 2: Directional Sobel operators

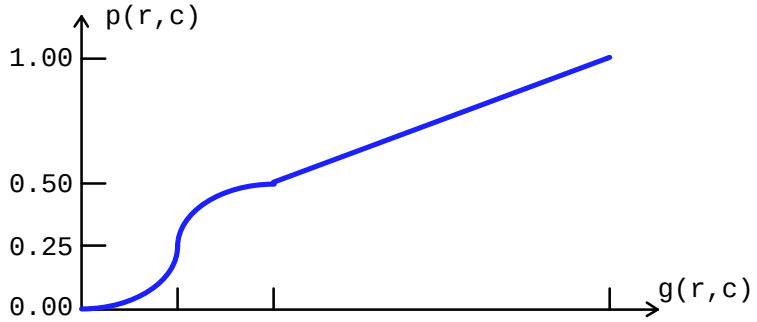


Figure 3: Gradient transformation

The transformation curve is shown in Figure 3. Notice that because the gradient has a positive or negative sign, the transformed $p(r,c)$ has the same sign. The aims of the edge enhancement network are now clear: for each 3x3 input area, output four normalized probabilities for each candidate edge orientation. The task is done by a four-layer FFNN. The input layer and the output layer contain 9 and 4 neurons respectively. Each of the two hidden layers have 20 neurons. The network is trained by a traditional BP algorithm.

3 Structural boundary extraction

3.1 Why training?

According to the literature in recent years, we find that the FFNNs are extremely suitable for the feature extraction stage as discussed in last section. The reason is quite straightforward: the features used for edge detection (gradient, covariance, moment, and so on) usually have simple analytical representations, and can be easily approximated by FFNNs. The supervised training signals are also definite. However, it is never so easy to train FFNNs for single-line boundary extraction because edges in an image usually present various profiles. The non-maximum restraining as well as the global edge structure linking are highly nonlinear. In previous literature recurrent networks (e. g., the Hopfield) have obtained much better boundary extraction result than simple FFNNs [4, 5].

On the other hand, we have developed a lot of rules for selecting edge pixels from the gradient map. For example, the gradient pixel should

be a local maximum along the gradient direction; the boundary chain should be continuous and smooth; etc. These rules are *formal knowledge*, however the rules in the neural network weight matrices are *hidden knowledge*. Our aim is to store the *formal knowledge* into neural networks. The traditional approach is to train the network by teacher patterns. However, as explained above, to train such a FFNN is not easy.

For this problem, we present a novel FFNN architecture *without* training. We provide a strategy for adding if-then-else-like *formal knowledge* into cascading FFNNs. The basic module is a Max-Min-net.

3.2 MMnet: the Max-Min-net

The Max-Min-net (MMnet) is a FFNN module to find maximum and minimum component from the input vector. As we known, finding maximum and minimum are extremely nonlinear operations. All neurons in the MMnet are same, and can be written as

$$o = f\left(\sum_i w_i x_i - \theta\right), \quad i = 1,\ldots,N, \tag{2}$$

where $\sum_i w_i x_i$ is the input value, θ is a threshold, o is the output value, and $f(\cdot)$ is an S-type function ranged between –1 and +1:

$$f(a) = \tanh a. \tag{3}$$

When $|a|$ is very small or very large, we have

$$f(a) \approx \begin{cases} a & |a| << 1 \\ 1 & a >> 1 \\ -1 & a << -1 \end{cases}. \tag{4}$$

If the desired output is only the maximum component in the input vector, the MMnet is simplified to a so-called MAXnet, as shown in Figure 4. For the ease of writing, let us study the MAXnet at $N = 3$. The MAXnet is a four-layer FFNN. Let $W_{II,I}(i, j)$ represent the weight linking the i-th neuron in layer II and the j-th neuron in layer I. Assume that ε is a very small positive number. Set

$$W_{II,I}(1,1) = \frac{1}{\varepsilon}, \quad W_{II,I}(1,2) = -\frac{1}{\varepsilon},$$

The output of the first neuron in layer II will be

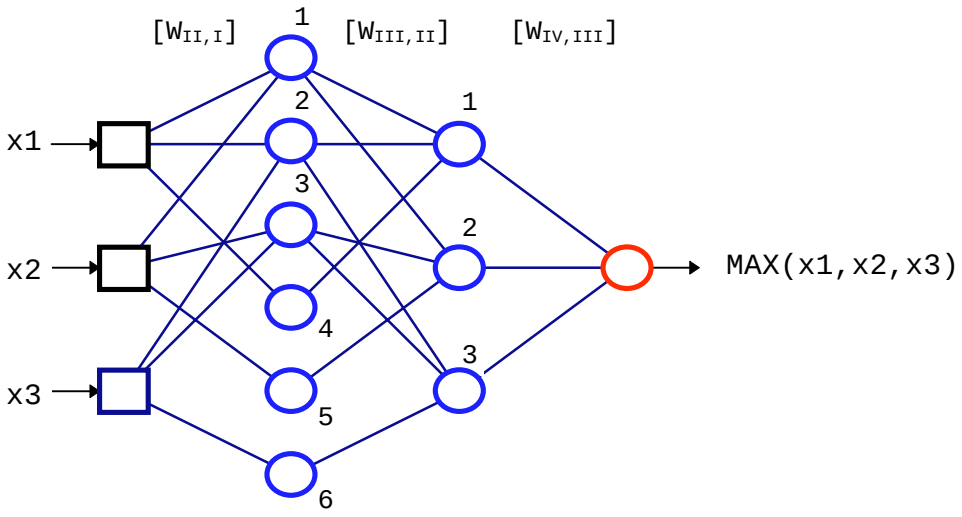


Figure 4: MAXnet structure

$$o_{II}(1)=\begin{cases}+1 & \text{if } x1>x2\\ 0 & \text{if } x1=x2\\ -1 & \text{if } x1<x2\end{cases}. \qquad (5)$$

Similarly, we set

$$W_{II,I}(2,1)=\frac{1}{\varepsilon},\quad W_{II,I}(2,3)=-\frac{1}{\varepsilon},$$

The output of the second neuron in layer II will be

$$o_{II}(2)=\begin{cases}+1 & \text{if } x1>x3\\ 0 & \text{if } x1=x3\\ -1 & \text{if } x1<x3\end{cases}. \qquad (6)$$

We set

$$W_{II,I}(4,1)=\varepsilon,$$

The output of the fourth neuron in layer II will be

$$o_{II}(4)=\varepsilon\cdot x1. \qquad (7)$$

Considering the first neuron in layer III, let us set

$$W_{III,II}(1,1)=\frac{1}{\varepsilon},\quad W_{III,II}(1,2)=\frac{1}{\varepsilon},\quad W_{III,II}(1,4)=1.$$

and let the threshold be $\theta=\frac{2}{\varepsilon}$, the neuron output will be

$$o_{III}(1)=\begin{cases}\varepsilon\cdot x1 & \text{if } x1>x2 \text{ and } x1>x3\\ -1 & \text{otherwise}\end{cases}. \tag{8}$$

At last, we set

$$W_{IV,III}(1)=W_{IV,III}(2)=W_{IV,III}(3)=\frac{1}{\varepsilon},$$

and let the output neuron threshold be $-\frac{2}{\varepsilon}$. The whole MAXnet weight matrices can be considerably determined as discussed above and the global network output is

$$o_{IV}=\mathrm{MAX}\{x1,x2,x3\}. \tag{9}$$

Until now we have built a MAXnet as a simplification of the MMnet. Generally speaking, A MMnet with full functionality is also a four-layer FFNN. It can be modeled as a black box module as shown in Figure 5. The input is an *N*-dimensional vector. The output contains the maximum, the minimum, and an *N*-dimensional Boolean vector:

$$o_i=\begin{cases}+1 & \text{if the } i\text{ - th component is the maximum}\\ -1 & \text{if the } i\text{ - th component is the minimum}.\\ 0 & \text{otherwise}\end{cases}$$

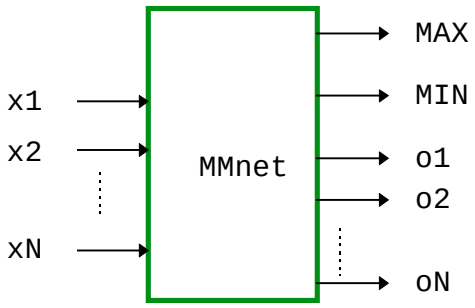


Figure 5: The MMnet module

3.3 Build FFNNs for structural boundary extraction

An image edge point is defined as a local gradient maximum in plain words. However, in visual recognition tasks these independent pixels must be linked to understandable edge chains, otherwise the grade of edge detection would be considerably lower. Given a certain pixel, all possible edge chains passing through it in 3x3 neighborhood are shown in Figure 6. For example, let us see the left figure which indicates nine possible horizontal edge chains. The intensity of a local edge chain is defined as the sum of the probabilities (see Eqn (1)) on the pixels that the chain passing through. Among all the nine links, there is a maximum intensity. Thus the intensity of the central pixel in the horizontal direction is measured. Similarly the intensities of the up and down pixels (the void vertices in the figure) are measured. If the central intensity is the maximum among them, it is considered to be a candidate edge point. In previous steps the gradients of eight directions have all been obtained, the current pixel is finally determined as a true edge pixel if it is also the maximum among all the intensity of eight directions.

We have set forth the formal operations to determining whether a pixel be an edge point or not. In this paper, the approach to achieve such an aim is to construct a FFNN by cascading several MMnet modules and some auxiliary neurons. The system architecture is illustrated in Figure 7.

As shown in Figure 6 left, the probabilities of left three pixels are u1, u2, and u3; the probabilities of right three pixels are v1, v2, and v3; and the probability of the central pixel is w. They are respectively fed into some MMnets and are added to be the local chain maximum CM(k), where the number k is the current edge direction. The intensity of the inverse direction (k+4) can be obtained by a dual neuron. Therefore, CM(1), CM(2), …, CM(8) are fed into another MMnet, the maximum intensity is gotten. In the same time, the output o2(k) for each direction is obtained, where o2(k) indicates if the central pixel has the maximum

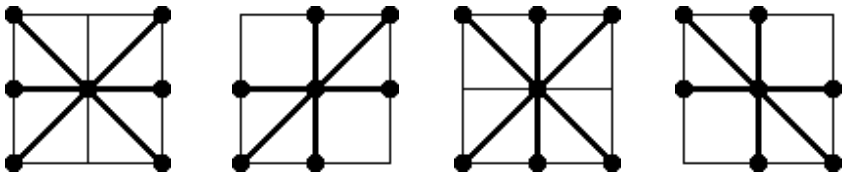
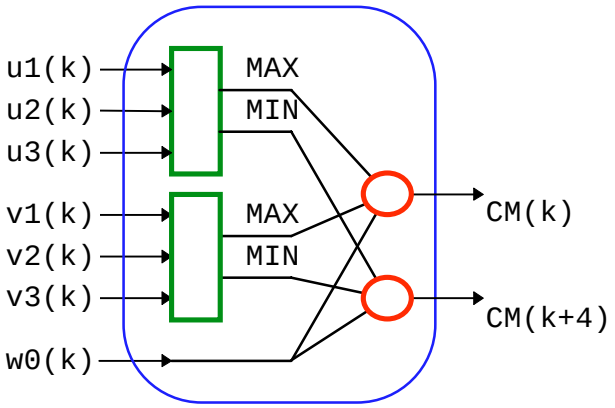


Figure 6: 8-direction local edge links in 3x3 neighborhood



Chain Maximum (CM), k = 1, 2, 3, 4

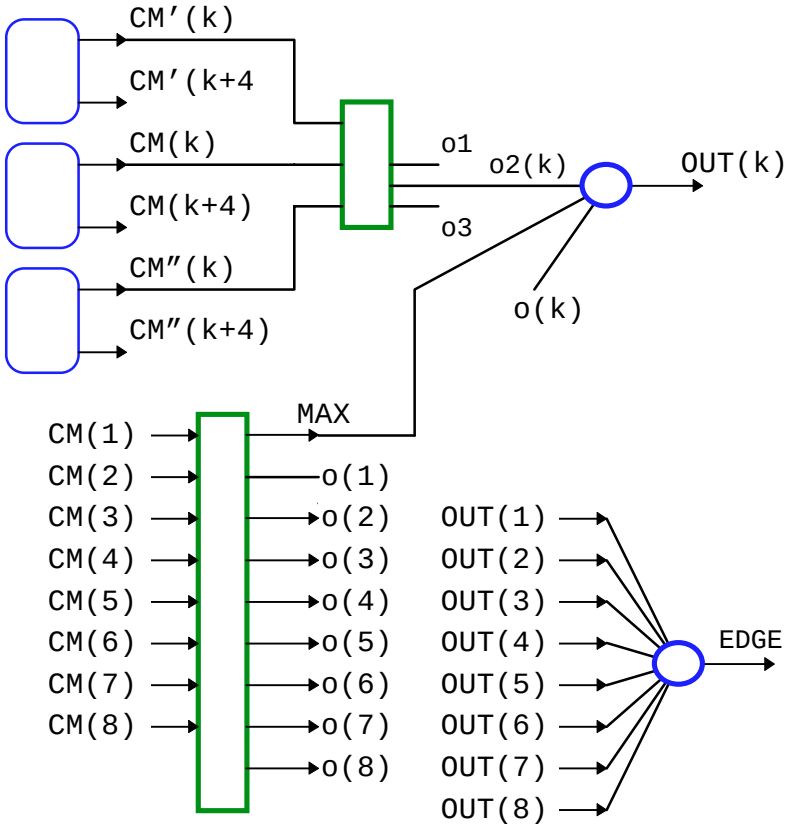


Figure 7: Structural boundary extraction network

intensity among its neighborhoods. The neurons before OUT(k) have constant thresholds, therefore OUT(k) is +1 if and only if it corresponds to a valid edge pixel. Finally, the edge signal is extracted using a simple perceptron by checking the sum of OUT(1) to OUT(8).

4 Experimental results

In this paper, we have introduced two FFNNs. The first one is depend on traditional supervised training, and the latter is manually built without training. While training the first network, a standard BP algorithm is used. For the ease of construct supervised patterns, we simply select a natural image, apply the modified Sobel operators on it make the translation by Eqn (1). Thus the training pattern is an area with 3x3 input gray level pixels and the output possibility. In practice a 64x128 area in the *Girl* image (see Figure 9) is used. The training procedure converges after 10000 cycles.

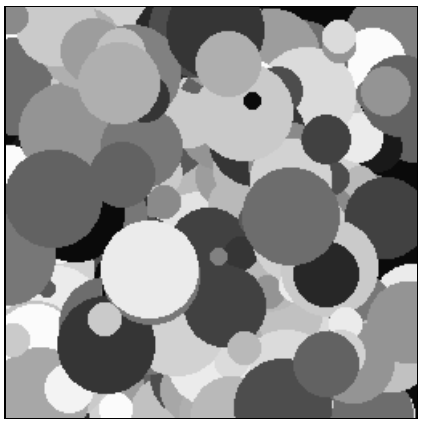
The edge detection experiments are made for both synthetic and natural images. The synthetic images are shown in Figure 8, where different levels of white noise are added to them. An objective measure, the Pratt's Figure of Merit (FoM), is used for evaluating the quality of edge detection. The FoM is define as [7]:

$$\mathrm{FoM} = \frac{1}{\max\{E_{ID}, E_{AC}\}} \cdot \sum_{i=1}^{E_{AC}} \frac{1}{1+\alpha d^2}, \tag{10}$$

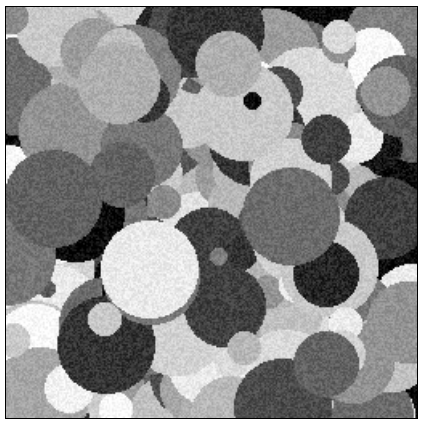
where E_{ID} and E_{AC} represent the number of ideal and actual edge pixels, α is a scaling parameter which penalizes offset edges, and d is the distance from the actual edge to its correct location. Some experimental results are shown in Figure 8 and Table 1. The minimum edge amplitude is 10, and the signal-to-noise ratios in dB are computed with respect to this value. Notice that images are degraded by up to – 6 dB white noises. The results shown that the FFNN edge detector in this paper can produce single-line, continuous, and smooth edge chains under a great range of noise level. Thanking to the structural boundary extraction network in Section 3, the figure of merit only changes a little when the noise level increases a lot.

Table 1: Figure of merit versus SNR

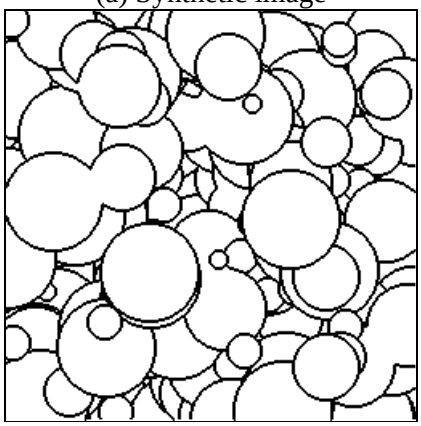
S/N	noise free	20 dB	14 dB	6 dB	0 dB	– 3.5 dB	– 6 dB
FoM	0.8595	0.6886	0.6808	0.6765	0.6876	0.6743	0.6361



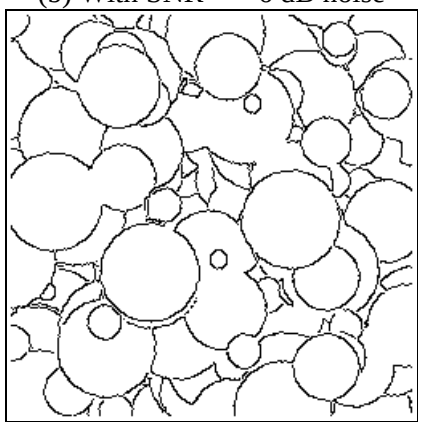
(a) Synthetic image



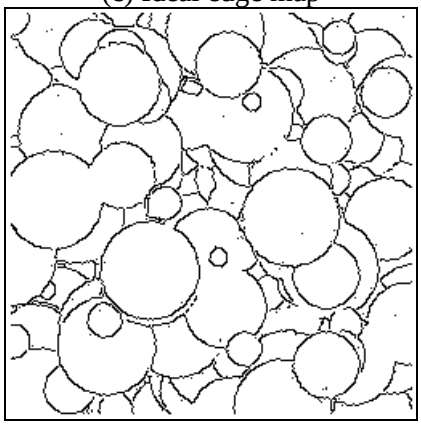
(b) With SNR = – 6 dB noise



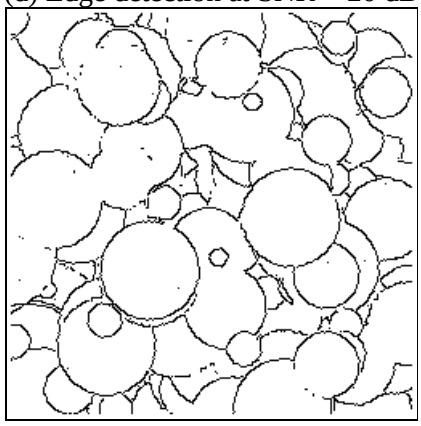
(c) Ideal edge map



(d) Edge detection at SNR = 20 dB



(e) Edge detection at SNR = 0 dB



(f) Edge detection at SNR = – 6 dB

Figure 8: Synthetic image edge detection

The rest experiments are based on natural image. Figure 9 is the *Girl* image. The edge maps produced by the FFNN edge detector, the Canny operator [8], and the optimal edge detector in [9] are compared. Figure 10-12 illustrates more experiments by the FFNN edge detector on natural images. Note that Figure 12 is a picture of very poor quality, but the distorted contours are extracted very well. In these images fine details and sharp corners have also been successfully extracted.

5 Conclusion

The research in this paper comprises two phases, namely, edge enhancement by a traditional FFNNs and boundary extraction by a manually built FFNN architecture without training. Training, the most difficult problem in feed-forward networks, is avoided. If a certain FFNN hardware is available in due course, The FFNN edge detector can also be very fast. The synthetic and natural image experiments show that the FFNN detector can obtain precise boundaries, recover missing edges, and eliminate false edges caused by noise. The produced edge chains are continuous and smooth.

Acknowledgements

The authors would like to thank Professor Armando Pinho, The University of Aveiro, Portugal, for his valuable comments on this paper.

References

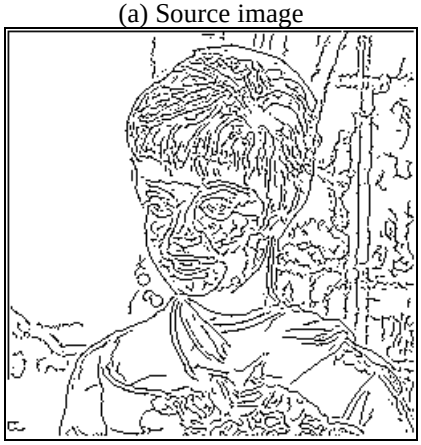
[1] Grossberg, S. & Mingolla, E., Neural dynamics of surface perception: boundary webs, illuminants, and shape-from-shading, *CVGIP*, **37**, pp. 116-165, 1987.

[2] Pinho, A.J. & Almeida, L.B., Edge detection filters based on artificial neural networks, *Proc. 8th Int. Conf. on Image Analysis and Processing*, Sanremo, Italy, pp. 159-164, 1995.

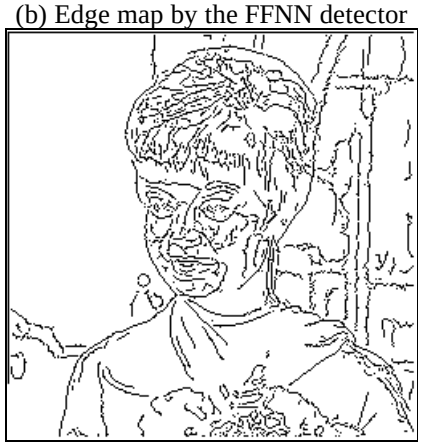
[3] Intajag, S. & Paithoonwatanakij, K., Edge detection by fuzzy neural network, *Proc. 11th Int. Conf. on Artificial Intelligence in Engineering*, eds. R.A. Adey, G. Rzevski & A.K. Sunol, pp. 248-257, 1996.

(a) Source image

(b) Edge map by the FFNN detector



(c) Edge map by the Canny detector



(d) Edge map by the optimal detector

Figure 9: *Girl* image

Figure 10: Edge detection based on FFNNs: *Lena* image

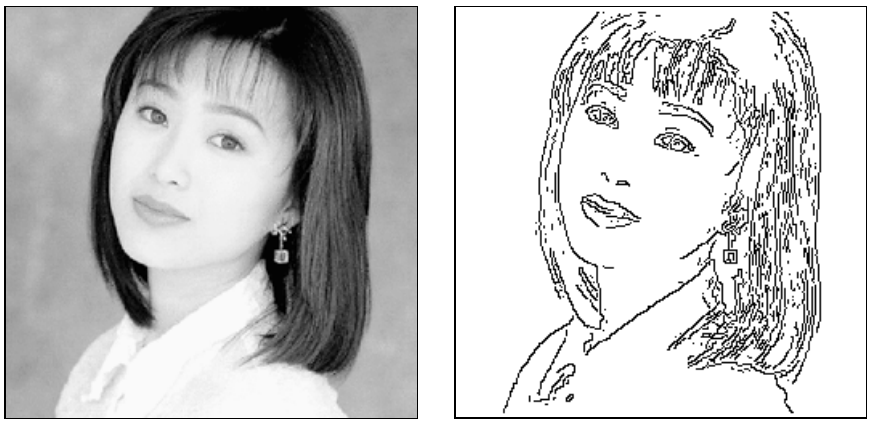


Figure 11: Edge detection based on FFNNs: *Noricosakai* image

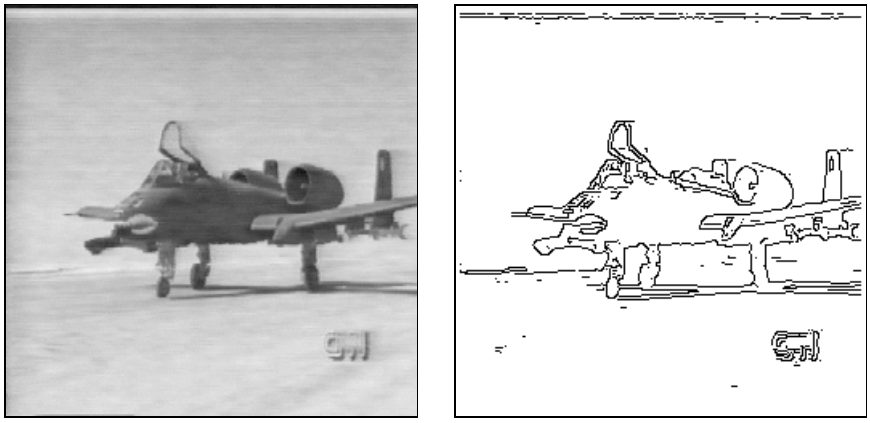


Figure 12: Edge detection based on FFNNs: *Airplane* image

[4] Lu, S.W. & Shen, J., Artificial neural networks for boundary extraction, *Proc. IEEE Int. Conf. on Systems, Man and Cybernetics*, **3**, Beijing, China, pp. 2270-2275, 1996.

[5] Vrabel, M.J., Edge detection with a recurrent neural network, *Proc. SPIE, Applications and Science of Artificial Neural Networks II*, **2760**, Orlando, FL, pp. 365-371, 1996.

[6] Kuffler, S.W., Nicholls, J.G. & Martin, A.R., *From Neuron to Brain*, 2nd Ed., Sinauer Associates Inc., 1984.

[7] Pratt, W.K., *Digital Image Processing*, John Wiley & Sons, Inc.,

1978.

[8] Canny, J., A computational approach to edge detection, *IEEE Trans. Pattern Analysis & Machine Intelligence*, **PAMI-8**, pp. 679-698, 1986.

[9] Zhou, L.X. & Gu, W.K., Adaptive edge detection: a new criterion and its recursive implementation. *Proc. 3rd Int. Conf. on Signal Processing*, Beijing, China, pp. 1130-1133, 1996.