Edge Detection Filters Based on Artificial Neural Networks

Armando J. Pinho[1] and Luís B. Almeida[2]

[1] Dep. Electrónica e Telecomunicações / INESC
Universidade de Aveiro, 3800 Aveiro, Portugal
(Fax +351-34-370545, Email ap@inesca.pt)
[2] INESC / Inst. Superior Técnico
R. Alves Redol, 9, 1000 Lisboa, Portugal
(Fax +351-1-525843, Email lba@inesc.pt)

Abstract. This paper presents quantitative results on the problem of edge detection using neural network filters. These results are compared with the results provided by the derivative of the Gaussian edge detection filter. A new figure of merit for edge quality, based on Pratt's figure of merit, is introduced. The results displayed in this paper give evidence that neural network edge detection filters can perform better than the linear "optimal" filters.

1 Introduction

In [1] some qualitative results were presented on the subject of edge enhancement using artificial neural networks. It was shown that this type of non-linear filters can provide good localization and low distortion of edges and also a good response to corners. In this paper we further extend that work and we present a more systematic and quantitative evaluation of the properties of these filters.

As in [1] the representation adopted for the edge elements is based on the inter-pixel *cracks* [2]. This type of edge representation offers an unambiguous placement of edge elements, an effective region separation and also a potentially greater capacity to retain information.

The evaluation of the quality of edges is often a subjective operation due to the lack of knowledge of the correct answer. The work reported here is based on synthetic images which allow us to access the respective edge maps. Therefore, in this case we can use an objective measure to evaluate the quality of the edges generated by the detectors. Pratt proposed a figure of merit [3] defined as

$$F_{\text{Pratt}} = \frac{1}{\max(E_{\text{ID}}, E_{\text{AC}})} \sum_{i=1}^{E_{\text{AC}}} \frac{1}{1 + \alpha d^2} \tag{1}$$

where E_{ID} and E_{AC} represent the number of ideal and actual edge points, α is a scaling parameter which penalizes offset edges, and d is the distance from the actual edge to its correct location. This figure of merit although useful does not correctly penalize spurious responses, due for example to noise or blur. As stated in (1) every actual edge is matched with one ideal edge. This means that

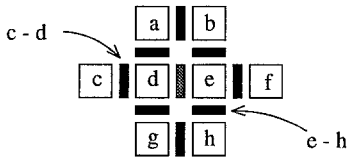


Fig. 1. The neural network input values are the first differences of the context pixels. As can be seen, the eight context pixels (represented as squares) generate nine differences (the rectangles), which form the input of the neural network used to process the central crack (the gray rectangle). The horizontal cracks are processed in a similar way, due to symmetry properties.

if E_{AC} is greater than E_{ID} there are some ideal edge elements that have to match more than one actual edge element. This results in a lack of explicit and effective accounting of false edges.

To improve Pratt's figure of merit we propose the following formulation:

$$F = \left(\frac{1}{E_{\mathrm{ID}}} \sum_{i=1}^{E_{\mathrm{ID}}} \frac{1}{1+\alpha d^2} \right) \times \left(\frac{1}{1+\beta \frac{E_{\mathrm{FA}}}{E_{\mathrm{ID}}}} \right) \tag{2}$$

The first term in parentheses is a modified version of Pratt's figure of merit, differing only in the direction of matching. While in (1) each actual edge element is matched with one ideal edge element, in the modified version each ideal edge looks for one, and only one, actual edge element. Also, an actual edge element can, at most, be allocated to an ideal edge element. All un-allocated actual edges are considered false edge elements and are handled by the second term in parentheses of (2). E_{FA} denotes the number of false (unmatched) edge elements.

The two parameters included in (2) (α and β) are responsible for a good balance between the missing and misplaced edge error types (α) and false edge error type (β). For the first one (α) we adopted the value $1/9$, suggested by Pratt. For the second parameter (β) we used the value 1, which seems to be a reasonable choice. This means that if the number of false edge elements is equal to the number of ideal edges the second term in parentheses will be 0.5.

2 Neural Network Filters for Edge Detection

In this work we used the same type of neural network topology as described in [1], i.e. feed-forward multi-layer perceptrons trained with back-propagation [4], improved with the acceleration technique proposed by Silva and Almeida [5].

The input of the neural networks is formed by the nine first differences calculated using adjacent pixels, as shown in Fig. 1. This figure also displays the topology of the eight pixels used as the input context (i.e. the support of the filter), which was inspired on early work of Hanson and Riseman on image segmentation based on relaxation labeling [6].

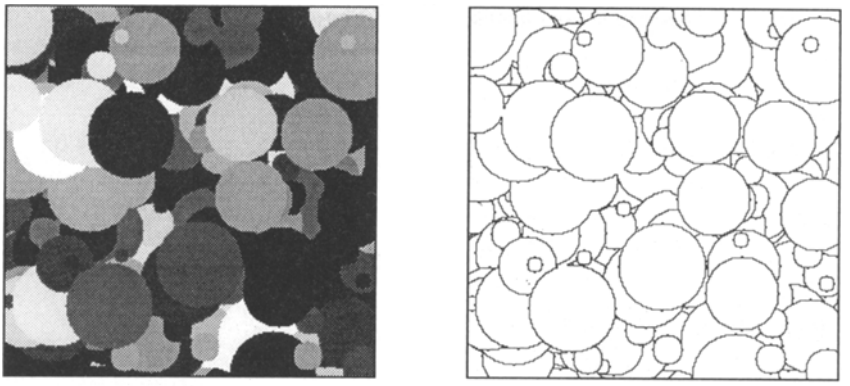


Fig. 2. a) Synthetic image used to train the neural networks and b) its desired contours.

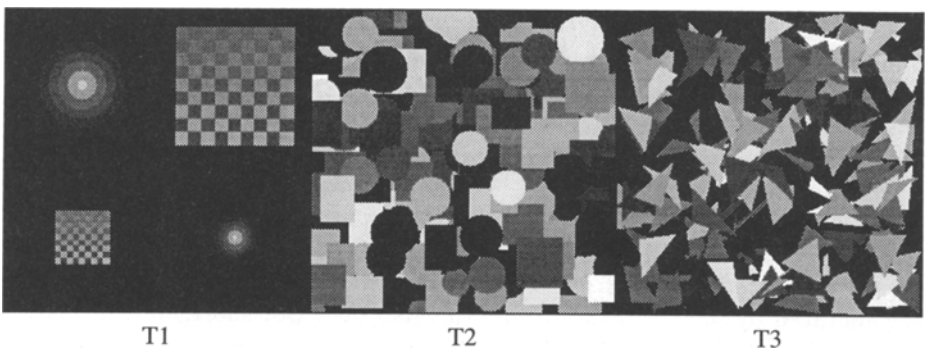


Fig. 3. Synthetic test images (T1, T2, T3).

The neural networks were trained and tested using synthetic images. On one hand this provides a better control of the training and test data and also an easy access to the desired edge maps. On the other hand the variability of the synthetic data can be considered poor when compared to data obtained from real images.

Figure 2a shows an image of 256×256 pixels, with 8 bits per pixel, composed of disks of diameters randomly drawn from the set $\{10, 20, \cdots, 60\}$. The gray level inside each disk is constant and was obtained randomly from the set $\{0, 10, 20, \cdots, 250\}$. Figure 2b depicts the desired contours, obtained by construction. To test the behavior of the neural network filters we used three synthetic images which are displayed in Fig. 3.

All images (training and test) were blurred using the 3×3 low pass mask

$$\begin{bmatrix} a & a & a \\ a & b & a \\ a & a & a \end{bmatrix}$$

with $a = 0.075$ and $b = 0.4$. This provides more realistic data since edges are

rarely step like. Also, four levels of additive Gaussian noise were used to degrade the images: $\sigma = 3$, $\sigma = 5$, $\sigma = 10$ and $\sigma = 20$. The objective is to obtain filters robust to noise (from the training point of view) and also to test their noise rejection capabilities.

To provide a simple way of referencing the images (training and test) subjected to the different levels of noise we adopt the following notation: `nameSnn`, where `name` is the name of the image (`Tr`, `T1`, `T2` and `T3`) and `nn` refers to the standard deviation of the noise (03, 05, 10 and 20).

3 Simulation Results

Four neural network configurations were tested. We denote them as H0, H2, H4 and H8, which stands for 0, 2, 4, and 8 units in the hidden layer. Note that the H0 network degenerates into a single layer neural network, which can be viewed as a linear filter followed by a non-linearity.

All networks were trained using 10000 examples, randomly extracted from the training image. Each training was performed during 2000 epochs[3], and was repeated with three different random initializations. The training process was controlled periodically (every 100 epochs) through the processing and evaluation of the training image. This procedure can be considered a kind of cross-validation[4], which is used to ensure that the optimal training point is not missed. The best of the three neural networks (which resulted from the three random initializations) was then retrained until the optimal point of the cross-validation curve.

We compared the results obtained with the neural network filters with the results given by the derivative of the Gaussian filter (for short, we will refer to it as the Ln filter), which is considered a good approximation of some optimally derived linear filters for step edges [7] [8]. The Ln filter is defined as

$$\mathrm{Ln}(x) = -\frac{x}{\sigma_f^2} \exp\left(-\frac{x^2}{2\sigma_f^2}\right),$$

where σ_f is the scale of the filter. For each image we determined the σ_f that provided the largest figure of merit. Also, since we need binary edge maps to compute the figure of merit, we always used the best possible threshold, both for the Ln and neural network filters.

Table 1 shows the figure of merit of the test images, after processing by the neural networks obtained from the various network topology and training data combinations. In addition, it also shows the results obtained with the Ln filter, using the best σ_f for each image. Note that the test images were never used during the process of neural network training, including the cross-validation operation. Only the training image was used for that purpose.

Figure 4 shows image `T3S05` processed by the Ln filter ($F = 0.53$) and also processed by the H4 neural network trained with image `TrS05` ($F = 0.67$).

[3] One epoch is a complete pass through the training set.

[4] Note that only a small part of the training image is actually used for training.

Table 1. Comparison of the figures of merit calculated for T1, T2 and T3 after processing by the Ln filter and the filters resulting from the several combinations of neural network topology and training sets. The σ_f line displays the scale of the Ln filters that offered the best result (maximum F) for each image.

	T1S03	T1S05	T1S10	T1S20	T2S03	T2S05	T2S10	T2S20	T3S03	T3S05	T3S10	T3S20
Ln	0.43	0.39	0.33	0.29	0.47	0.47	0.44	0.38	0.54	0.53	0.50	0.42
σ_f	0.35	0.55	0.65	0.70	0.10	0.20	0.30	0.60	0.05	0.05	0.30	0.55
Neural networks trained with image TrS03												
H0	0.44	0.36	0.27	0.19	0.55	0.53	0.41	0.25	0.64	0.61	0.49	0.33
H2	0.49	0.41	0.26	0.17	0.66	0.59	0.38	0.20	0.71	0.65	0.47	0.29
H4	0.63	0.46	0.28	0.19	0.70	0.60	0.40	0.22	0.74	0.68	0.49	0.31
H8	0.63	0.45	0.27	0.19	0.69	0.59	0.38	0.21	0.74	0.67	0.48	0.31
Neural networks trained with image TrS05												
H0	0.43	0.38	0.30	0.23	0.47	0.47	0.44	0.31	0.54	0.53	0.50	0.39
H2	0.41	0.40	0.29	0.18	0.64	0.59	0.45	0.22	0.67	0.63	0.51	0.31
H4	0.57	0.49	0.34	0.24	0.64	0.62	0.48	0.29	0.70	0.67	0.56	0.38
H8	0.57	0.50	0.35	0.24	0.66	0.63	0.51	0.35	0.71	0.67	0.57	0.42
Neural networks trained with image TrS10												
H0	0.37	0.37	0.32	0.27	0.39	0.39	0.39	0.37	0.42	0.41	0.42	0.41
H2	0.36	0.36	0.33	0.25	0.49	0.48	0.45	0.34	0.51	0.50	0.47	0.40
H4	0.52	0.46	0.37	0.25	0.59	0.58	0.50	0.33	0.63	0.61	0.56	0.42
H8	0.46	0.42	0.36	0.26	0.61	0.59	0.52	0.37	0.65	0.63	0.57	0.44

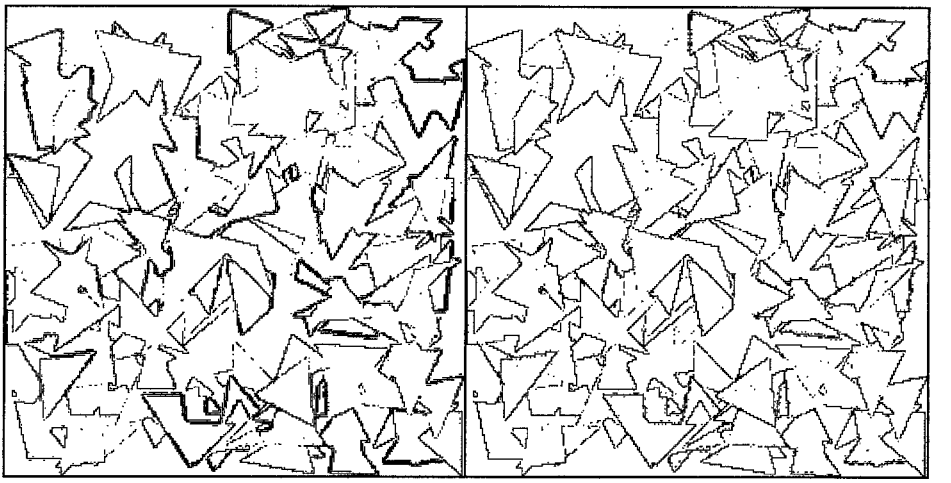
4 Discussion and Conclusions

Analyzing globally the results of Table 1, it is evident that neural network edge detection filters can outperform linear filters. Some other important observations can also be drawn from that table.

As a first observation we note the degradation of performance of the neural networks for images of high level of noise. This should be expected since the input support of these filters is small (only 4 pixels in the direction orthogonal to the edge) when compared to the support of at least 7 pixels used for scales $\sigma_f \geq 0.5$. Image T3, which needs smaller scales, is the least affected by this behavior (compare the T1S20, T2S20 and T3S20 columns of Table 1).

Another important observation is the variation of performance with the choice of the noise level in the training image. Low noise during training offers a good performance on low noise images, as would be expected, but at the expense of poor performance on medium and high noise images. If the noise level of the training image is increased then we obtain a somewhat lower performance on low noise images but the medium and high noise images are improved.

The variation of performance due to the complexity of the neural networks (i.e., with the number of hidden units) is more or less as expected. It increases, in general, with the increase in network complexity. However, we could not explain why it decreases when we pass from a H0 to a H$n, n \neq 0$ topology, for high noise images (see the T1S20, T2S20 and T3S20 columns of Table 1).



a) b)

Fig. 4. a) Image T3S05 processed by the Ln filter ($F = 0.53$); b) Image T3S05 processed by the H4 neural network trained with image TrS05 ($F = 0.67$).

The study that we present in this paper is focused on small input support neural networks. Using only these results we are not able to extrapolate about the behavior of this kind of filters for large input supports. However, we believe that the improvement in performance that we observed here will also show for large input supports.

References

1. A. J. Pinho and L. B. Almeida. Some results on edge enhancement with neural networks. In *Proc. of the 1st IEEE Int. Conf. on Image Processing (ICIP'94), Austin, Texas, U.S.A.*, 1994.
2. D. H. Ballard and C. M. Brown. *Computer vision.* Prentice-Hall, Inc., 1982.
3. W. K. Pratt. *Digital image processing.* Wiley-Interscience, 1978.
4. D. E. Rumelhart, J. L. McClelland, and PDP Research Group. *Parallel Distributed Processing - Explorations in the Microstructure of Cognition: Foundations.* Volume 1, MIT Press / Bradford Books, 1986.
5. F. M. Silva and L. B. Almeida. Acceleration techniques for the backpropagation algorithm. In L. B. Almeida and C. J. Wellekens, eds., *Neural Networks, Proc. EURASIP Workshop, Sesimbra, Portugal*, Springer-Verlag, 1990.
6. A. R. Hanson and E. M. Riseman. Segmentation of natural scenes. In A. R. Hanson and E. M. Riseman, eds., *Computer Vision Systems*, pp. 129–163, Academic Press, 1978.
7. J. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1986, **8**, pp. 679–698.
8. S. Sarkar and K. L. Boyer. On optimal infinite impulse response edge detection filters. *IEEE Trans. on Pattern Analysis an Machine Intelligence*, 1991, **13**, pp. 1154–1171.