Contextual Edge Detection Using a Recurrent Neural Network

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

[1] Dep. Electrónica e Telecomunicações / INESC
Universidade de Aveiro, 3810 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. If we consider edge detection as a classification problem, then it seems reasonable that context should play an important role in its study. In fact, it is frequent that neighboring pixels exhibit a strong inter-dependence. In this paper we propose a recurrent neural network for edge detection, which uses a special architecture intended to incorporate contextual information during operation. Some experimental results are presented, showing its effectiveness.

1 Introduction

Probably, the most traditional approach to edge detection is based on a filtering and differentiation paradigm. It is well known that edges are characterized by abrupt changes in the intensity profiles of images and, therefore, spatial derivatives of the light intensity can be helpful in detecting and localizing them. However, it is also known that the differentiation of digital images is an ill-conditioned problem, requiring regularization [1].

A common approach to regularization is to apply low-pass filtering to the data. In general, this implies that the high frequency components of noise are attenuated, allowing a more convenient differentiation. Most of the edge detection filters can be analyzed under this perspective, i.e., composed of low-pass filtering followed by a differentiation operation.

There are two criticisms that can be made to the linear filtering approach to edge detection. First, although it is true that low-pass filtering reduces high frequency noise, it is also true that it attenuates edges. Second, these filters were designed for specific types of edge profiles (typically, step edges), which may not be appropriate for some classes of images.

In [2,3] we presented some results on edge detection using neural networks. Since these systems are simultaneously non-linear and adaptable, they are in a good position to attack both problems that we mentioned. In this paper we extend that work. However, instead of regarding the neural networks as non-linear filters, as in [2,3], we study them as pattern recognition devices. An important advantage of this approach is that contextual information can be handled in a more convenient way.

Next, we introduce some notation and definitions related to images and edge maps. Also, we discuss some data pre-processing issues and the topology used for the input supports of the neural network edge detectors.

Definition 1 (Image) *We define an image, g, as a mapping $\mathcal{C}\times\mathcal{R} \xrightarrow{g} \mathcal{I}$, with $\mathcal{C} \stackrel{\text{def}}{=} \{0,1,\ldots,N_{\mathcal{C}}-1\}$, $\mathcal{R} \stackrel{\text{def}}{=} \{0,1,\ldots,N_{\mathcal{R}}-1\}$, $\mathcal{I} \stackrel{\text{def}}{=} \{0,1,\ldots,N_{\mathcal{I}}-1\}$, and $N_{\mathcal{C}}, N_{\mathcal{R}}, N_{\mathcal{I}} \in \mathbb{N}$. The $(c,r) \in \mathcal{C}\times\mathcal{R}$ are coordinate pairs in a Cartesian system, where c denotes the column and r denotes the row.*

We use a representation of edge maps similar to that of [2, 3], i.e., based on the inter-pixel sites (*cracks*). The next definition describes the set of cracks (interstitial mesh) associated with an image.

Definition 2 (Interstitial mesh) *We define the interstitial mesh, $\mathcal{M}$, associated with image g, as*

$$\mathcal{M} \stackrel{\text{def}}{=} \Big\{(\theta,c,r) : (\theta,c,r) \in \{-,\,|\} \times \{0,\ldots,N_{\mathcal{C}}-2\} \times \{0,\ldots,N_{\mathcal{R}}-2\} \cup \{-\}\times\{N_{\mathcal{C}}-1\}\times\{0,\ldots,N_{\mathcal{R}}-2\} \cup \{\,|\,\}\times\{0,\ldots,N_{\mathcal{C}}-2\}\times\{N_{\mathcal{R}}-1\}\Big\}$$

where (c,r) denotes the image element that refers to the interstitial mesh element (θ,c,r), and θ denotes the orientation of that mesh element: $\theta = -$ denotes horizontal orientation and $\theta = |$ vertical orientation.

Definition 3 (Edge map) *Considering g an image and $\mathcal{M}$ its interstitial mesh, we define an edge map, Ω, as a mapping $\mathcal{M} \xrightarrow{\Omega} [-1,1]$.*

Definition 4 (Ideal edge map) *We define an ideal edge map, Ψ, as an edge map for which*

$$\Psi(\theta,c,r) \in \{-1,0,1\},\ \forall(\theta,c,r) \in \mathcal{M}$$

where -1 and 1 represent active edge elements (the sign encodes edge orientation) and 0 represents inactive edge elements.

To ensure invariance to changes in the average intensity of the images, we feed the neural network with the first differences calculated using adjacent pixels, instead of the intensity values:

Definition 5 (Initial edge map) *Considering g an image, we define an initial edge map, Φ, as an edge map for which*

$$\Phi(\theta,c,r) = \begin{cases} \dfrac{g(c,r)-g(c+1,r)}{2^b-1} & \text{for } \theta = | \\ \dfrac{g(c,r)-g(c,r+1)}{2^b-1} & \text{for } \theta = - \end{cases}$$

where b denotes the number of bits used in the representation of the intensity values, i.e., $\mathcal{I} = \{0,1,2,\ldots,2^b-1\}$.

Figure 1 shows the topology of the input supports of order 0, 1 and 2 of the neural network edge detectors addressed in this paper. Note that the support of order 1 is the same as the one used in [2,3]. Also note that the construction of these supports obey to a quite simple and regular rule, and that the number of cracks needed to build a support of order $n \geq 0$ is given by $N_\Omega(n) = (2n+1)^2$.

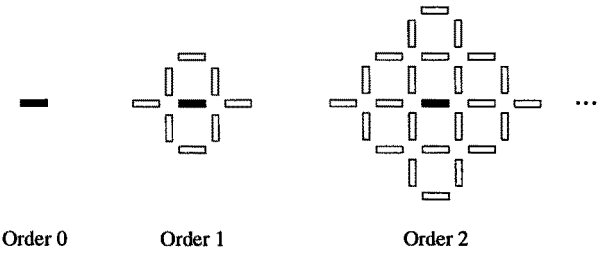

Fig. 1. Topology of the input supports of order 0, 1 and 2, used by the neural network edge detectors addressed in this paper.

In what follows, we present a recurrent neural network topology for edge detection, intended to incorporate contextual information. Its operation vaguely resembles another context incorporation paradigm, known as "probabilistic relaxation". However, although not discussed here, we argue that the method proposed in this paper avoids some important drawbacks of probabilistic relaxation, and also offers some additional advantages.

2 A recurrent neural network for edge detection

A possible way to address edge detection in a framework of pattern recognition is the following. Let $\mathbf{x}$ be a feature vector containing relevant information related to the problem of finding if a contour crosses the image area from where $\mathbf{x}$ was obtained. We represent by ω_c the presence of an active edge element[3], and by $\bar{\omega}_c$ the absence of an active edge element (or presence of an inactive edge element). The *a posteriori* probability that vector $\mathbf{x}$ belongs to class ω_c can be given, for example, by Bayes formula [4].

One of the most recent techniques that is able to provide direct estimates of the *a posteriori* probabilities is based on a relatively broad class of artificial neural networks. Several authors provided proves showing that, under certain conditions, the output values of a neural network are approximations of *a posteriori* probabilities or some combination of them (see, for example, [5–9]).

We denote by Π the neural network, and by $\hat{y} = \Pi(\mathbf{x})$ the estimate of the *a posteriori* probability that $\mathbf{x}$ belongs to class ω_c. In this paper we will not go further into details of how the neural network edge detector was implemented (see [10]). Instead, we will focus on the issue of context incorporation, which is more related to the problem of finding an appropriate feature vector.

[3] An active edge element is an edge element belonging to a contour.

Our objective is to increase the amount of information used for the classification of the edge elements, avoiding, at the same time, increasing the size of the support regions and, consequently, increasing the number of parameters of the neural network. Let us consider the neural classifier as being made, internally, of a pyramid of sub-systems, as displayed in Fig. 2.

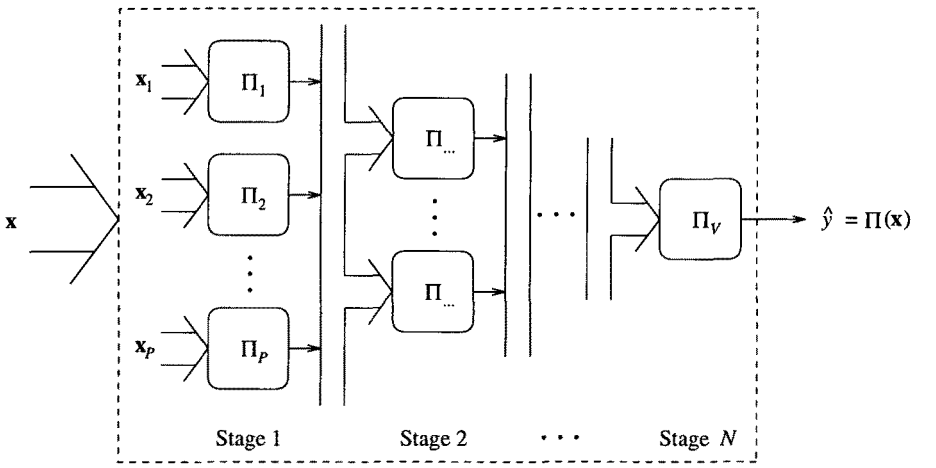

Fig. 2. Schematic representation of a neural network classifier built of a pyramid of sub-systems.

Although externally the feature vector is $\mathbf{x}$, from an internal point of view each sub-system $\Pi_i, i = 1, \ldots, P$ only "sees" part of it. Therefore, the dimensionality of the classifier is reduced according to the dimensionality of the sub-systems, maintaining, simultaneously, a large input context.

There are some comments that can be made about the structure depicted in Fig. 2. First, its pyramidal shape suggests that the information conveyed by $\mathbf{x}$ is integrated, stage after stage, until the final decision is performed during the final stage. This can be seen, also, as an iterative process, where each stage corresponds to an iteration done towards a final classification. Although, in principle, we can have arbitrary sub-systems, the case where they are all equal, i.e. $\Pi_i = \Pi_j, \forall i, j$, is particularly interesting. In that case, the classification process is simplified to the iterative use of the same sub-system, as many times as the number of stages (N).

We would like to note that, by imposing restrictions to the architecture of the neural network we may also limit the overall capacity of the classifying system. Therefore, there is a trade-off between the minimum dimension of the sub-systems and the potential capacity of the structured system to implement the classifier. Unfortunately, that compromise is not easy to determine.

Figure 3 shows a more detailed example (although quite simple) of the type of system that we used for edge detection. As can be seen, the sub-system Π_{Sub} has two different kinds of input information. On one hand, it receives information

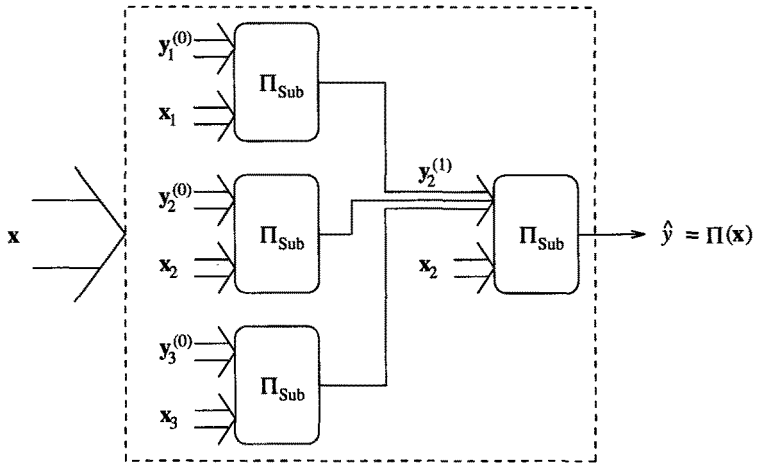

Fig. 3. Schematic representation of an example of a neural classifier based on two iterations of the sub-system Π_{Sub}.

from (part of) the feature vector, $\mathbf{x}$, that we will refer to as *evidence*. On the other hand, the sub-system has access to information related with *context*, given by the previous stage (iteration), i.e., $\mathbf{y}^{(t)}$, where t denotes the number of the stage (iteration). By convenience, we define $\mathbf{y}^{(0)} \stackrel{\text{def}}{=} \mathbf{x}$, i.e., the initial contextual information coincides with the evidence itself.

Let us analyze more in detail the example of Fig. 3, assuming that it corresponds to the problem of classifying edge elements. The initial aim is to classify the edge element characterized by feature vector $\mathbf{x}_2$. Observing the output stage, and ignoring the presence of the input $\mathbf{y}_2^{(1)}$, then Π_{Sub} could be used to perform that classification. On the other hand, we want to use more available evidence without compromising, at the same time, the dimensionality of the problem. Let us admit that $\mathbf{x}_1$ and $\mathbf{x}_3$ are feature vectors of two spatially neighboring edge elements, in relation to the one we are classifying. Then, assuming that the output of the first stage (i.e., $\mathbf{y}_2^{(1)}$) is a rough classification of those three elements, we can regard the second (and last stage) as the classification of the central edge element, improved by the previous provisional assignment of classes. Therefore, although indirectly, the whole feature vector $\mathbf{x}$ is used in the classification.

The amount of context that is introduced can be easily controlled by the number of stages (iterations). Since the system presents a pyramidal structure, each stage that is added implies the enlargement of the context, i.e., the increase of the dimension of $\mathbf{x}$. The advantage of this solution is that the dimensionality of the classifier, which is determined by the dimension of the vectors $\mathbf{x}_i$ and $\mathbf{y}_i^{(t)}$, is, generally, small.

For the experimental results that we present in this paper, we chose a subsystem characterized by an input evidence with the topology of a support of order 1, and also the same configuration for the contextual input. This means that the

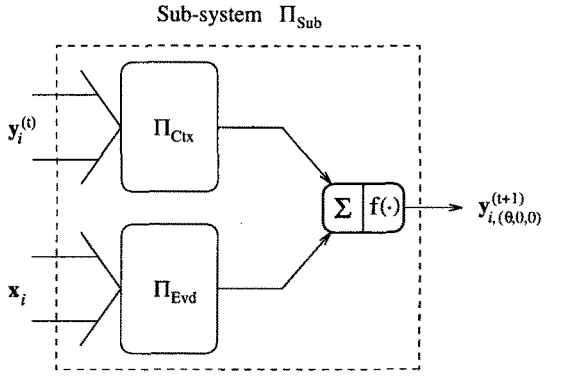

Fig. 4. Architecture of the sub-system that we used, which contains a block (Π_{Ctx}) that processes contextual information, and a block (Π_{Evd}) that processes information obtained from evidence.

neural network sub-system has 18 inputs and one output, and an architecture identical to that shown in Fig. 4. Blocks Π_{Ctx} and Π_{Evd} are two arbitrary neural networks, connected by one output unit. Note that, by separating the two blocks, we are able to better control their individual capacity.

Both $\mathbf{x}_i$ and $\mathbf{y}_i^{(t)}$ have the structure of a support region of order 1, obtained from the initial edge map, Φ, and the edge map under construction, $\widehat{\Psi}^{(t)}$, respectively. The output value of the sub-system Π_{Sub} corresponds to the element of the region of support associated with $\mathbf{y}_i^{(t+1)}$, having coordinates $(\theta, 0, 0)$, i.e., the central element.

Figure 4 displays a recurrent system characterized by a discrete and synchronous temporal evolution. After appropriate training, it is used to process an image in the same way as a convolution filter, as many times as the number of iterations for which it was designed. It is not difficult to see that a convenient algorithm to perform the training is back-propagation through time [11, 12]. In fact, the unfolding of the recurrent system that we used in Figs. 2 and 3, to explain how it works, is also the principle used by the back-propagation through time training algorithm.

3 Experimental results and conclusions

In this section we present some results of edge detection obtained with the system that we just described. In this paper, we are most interested in showing that the iterative method proposed improves the classification, when compared to a non-iterative approach. In other words, our aim is to show that, indeed, it is able to incorporate context.

Training was performed using 25 000 randomly extracted examples from the image displayed in Fig. 5(a). This (256 × 256 pixels, 8 bits per pixel) image is built of circles and triangles with gray levels obtained from sampling of sloped

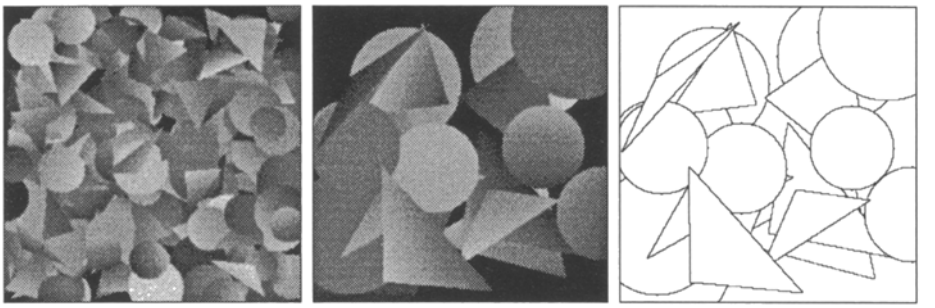

Fig. 5. (a) Synthetic image used for training. (b) Synthetic image used for testing the neural network and (c) its ideal edge map.

planes. Also, some Gaussian blurring and random noise were added. Another image with similar characteristics was used for testing (see Fig. 5).

We trained recurrent neural networks for one, two and three iterations. The one-iteration network is, in fact, non-recurrent, and it serves as comparison. For all the sub-systems (see Fig. 4) we used neural networks with four units in the hidden layer to implement the Π_{Evd} module, and with two units in the hidden layer to implement the Π_{Ctx} module.

Figure 6 shows edge maps obtained with the neural network edge detectors. It is not difficult to observe from those maps that recurrence improves, in fact, the quality of the edges detected. Going from the non-recurrent (one iteration) network to the neural network designed to perform three iterations we can observe that, in general, multiple responses are reduced (i.e., the edges are thinner) and some gaps are filled (i.e., connectivity improves).

We can conclude, therefore, that the recurrent system that we propose in this paper successfully uses the context that becomes available, iteration after iteration. It is important to note that we did not need to make any assumptions about the statistics of the data. Apart from the architecture of the neural network, all other information is collected from the training data, during training.

References

1. M. Bertero, T. A. Poggio, and V. Torre. Ill-posed problems in early vision. *Proceedings of the IEEE*, 1988, **76**, pp. 869–889.
2. 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, TX, 1994, pp. 893–897 (vol. III).
3. A. J. Pinho and L. B. Almeida. Edge detection filters based on artificial neural networks. In *Proc. of the 8th Int. Conf. on Image Analysis and Processing, ICIAP'95*, Sanremo, Italy, 1995, pp. 159–164.
4. R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* Wiley-Interscience, 1973.

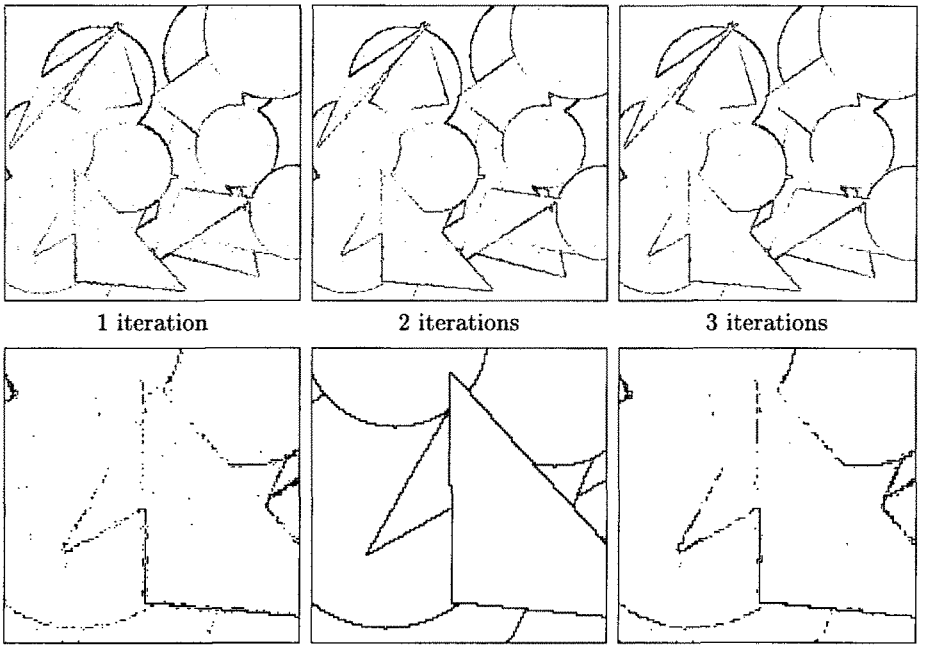

Fig. 6. Edge maps, of the test image, obtained with neural networks designed for 1, 2 and 3 iterations. On the lower half of the figure we display a zooming of part of the maps obtained with neural networks of 1 and 3 iterations, together with the same zone of the ideal map.

5. E. A. Wan. Neural network classification: a Bayesian interpretation. *IEEE Trans. on Neural Networks*, 1990, **1**, pp. 303–305.
6. D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter. The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Trans. on Neural Networks*, 1990, **1**, pp. 296–298.
7. P. A. Shoemaker. A note on least-squares learning procedures and classification by neural network models. *IEEE Trans. on Neural Networks*, 1991, **2**, pp. 158–160.
8. F. Kanaya and S. Miyake. Bayes statistical behavior and valid generalization of pattern classifying neural networks. *IEEE Trans. on Neural Networks*, 1991, **2**, pp. 471–475.
9. M. D. Richard and R. P. Lippmann. Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Computation*, 1991, **3**, pp. 461–483.
10. A. J. Pinho and L. B. Almeida. Neural network classifiers for edge detection. In A. Sanfeliu, J. J. Villanueva, and J. Vitrià, eds., *Pattern Recognition and Image Analysis, Proc. of NSPRIA'97*, Barcelona, Spain, 1997, pp. 371–376.
11. D. E. Rumelhart and J. L. McClelland, eds. *Parallel distributed processing — explorations in the microstructure of cognition: foundations.* Volume 1, MIT Press, Cambridge, MA, 1986.
12. P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 1990, **78**, pp. 1550–1560.