Image Thinning by Neural Networks

A. Datta[1], S. Pal[2] and S. Chakraborti[3]

[1]Computer and Statistical Service Centre, [2]Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta, India; [3]Computer Science Department, University of Kalyani, Kalyani, India

An image thinning technique using a neural network is proposed. Using different activation functions at different layers, the proposed neural network removes the boundary pixels from four directions in such a manner that the general configuration of the input pattern is unaltered and the connectivity is preserved. The resulting object, called a skeleton, provides an abstraction of the global shape of the object. The skeleton is often useful for geometrical and structural analysis of the object. The output skeleton here satisfies the basic properties of a skeleton, namely connectivity and unit thickness. The proposed method is experimentally found to be more efficient in terms of better medial axis representation and robustness to boundary noise over a few existing algorithms.

Keywords: Binary image; Neural network; Shape; Skeleton; Thinning

1. Introduction

In image processing, *thinning* is the process of reducing the width of a digitised pattern to just a single pixel so that the topological properties are preserved. Thinning has been of continuous interest in object representation for the last few decades. The output of thinning is called a *skeleton*. The purpose of thinning is to obtain an approximation of the *medial axis* of an object that can be formally defined (in a continuous case) as follows. Consider the circles that lie totally within the object and touch the object boundary at two or more points. The centres of all such circles form the medial axis of the object [1].

Skeletonisation, the process of finding the skeleton (in the continuous case, it is called Medial Axis Transformation, MAT), was introduced by Blum [2]. His procedure for finding the medial axis was to set up 'grassfire', which can be described as follows. Assume that the object region is filled with dry grass and fire is set everywhere on the boundary simultaneously. As the fire burns the grass, the object becomes thinner. Suppose that the fire line propagates with constant speed from the boundary of the object towards its inside. Then all the points lying in positions where at least two wavefronts of the fire line meet during the propagation (quench points) will form the medial axis of the object.

In this paper, we deal with thinning of *binary images*. Binary images (two-level images) are obtained from *grey-level images* by suitable segmentation (background-object separation). Many applications need the segmentation to be done into two levels only – one for the object and the other for the background. For example, in character recognition problems, the character patterns are to be separated from the background; in fingerprint analysis only the fingerprint ridges are required.

Skeletons are often useful for topological analysis and classification of the shape of patterns containing elongated or line like parts [3] (e.g. printed or handwritten characters, chromosomes, printed circuit board, etc.). A skeleton requires much less storage space compared to the original pattern and, at the same time, it contains the essential structural information of the pattern. Thus, skeletons are also useful in data reduction. For digital binary objects a number of thinning techniques have been developed.

The most common technique is the removal of outer layers of object pixels iteratively so that the general configuration and connectivity are not disturbed. These iterative algorithms can be either sequential or parallel. The sequential algorithms operate on a single pixel at a time, and the operation depends upon the preceding processing results. Parallel algorithms operate on all or a subset of all pixels simultaneously. Another thinning technique is based on Distance Transform (DT). A few other techniques have been developed based on run-length encoding, polygon approximation, contour tracing, etc. A survey on thinning algorithms is available in Lam et al. [3].

In the last few years, there has been a great interest in applying neural networks technology in various fields of conventional computing [4,5], and this trend, in turn, has been enriching the neurocomputing technology. This is due to the facts that neurocomputing provides parallelism, it is adaptive, and it sometimes simplifies a problem. A number of processors, each performing simple computation, when working collectively, can solve a complex problem. Neurocomputing provides a new form of parallel computation. Although a good number of binary thinning algorithms exist that use various conventional techniques, namely template matching, distance transform, run-length encoding, polygon approximation, contour tracing, etc. (for details, see Lam et al. [3]), very few algorithms have been developed using the neural network technique. This motivates the authors to develop the present neural network-based thinning algorithm for binary images.

In this paper, we propose a thinning technique based on a connectionist model. Connectionist models, also called neural network models, are massively parallel interconnections of simple computational elements (processors or nodes) that work as a collective system [6]. Connectionist models are capable of computing simultaneously using networks composed of a number of processors connected by links in an adaptive manner. For every connection, connection weights are assigned which can be learned. In a connectionist model, a complex problem can be solved by a number of processors working collectively, while each processor needs to do much simpler computation. The characteristics of the processors (e.g. the activation function), the links, the network topology and the learning mechanism characterise a connectionist model.

Several neural network models have been proposed so far [6]. Neural network models are specified by their network topology, node characteristics and training/learning rules. Training is accomplished by applying external input signals (usually vector valued). The rules specify how the weight vectors are updated against the presentation of each input vector. After a finite number of updates, the weight vectors converge and then the network is said to be trained/converged. Based on the training/learning technique, the neural network models can be broadly classified into two categories: (1) *supervised* and (2) *unsupervised*.

There have been attempts at neural network-based thinning in the past. For example, Krishnapuram and Chen [7] investigated the use of recurrent neural networks for thinning. Their basic algorithm was based on iterative pixel removal [8]. The authors used a backpropagation learning rule to train a three-layer network to learn a set of connection weights so that the network can classify the input patterns into deletable and undeletable pixels. In this paper, we formulate a thinning algorithm, which is based on a template-based four-pass algorithm [9], into a multi-layer unsupervised neural network. Using different activation functions at different layers, the proposed neural network removes the boundary pixels from four directions in such a manner that the general configuration of the input pattern is unaltered and the connectivity is preserved.

2. Preliminaries

We shall now state the definitions that will be used in different contexts of our discussion. The input here is a binary image, i.e. each pixel is either 1 or 0. The pixel 1 represents the object pixel, and the pixel 0 represents the background pixel.

Definition 1. For any object pixel p there are four four-neighbouring pixel β (four-neighbours), namely p_2, p_4, p_5, and p_7; and eight eight-neighbouring pixels (eight-neighbours), namely p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8 (Fig. 1).

Definition 2. $\pi(p,q)$ is said to be an eight-path (resp. four-path) of length n from an object pixel p to another object pixel q if there exists a sequence of object pixels ($p = a_0, a_1, a_2, \ldots, a_n = q$) such that a_i is an eight-neighbour (resp. four-neighbour) of a_{i-1}, $1 \leq i \leq n$.

p_1	p_2	p_3
p_4	p	p_5
p_6	p_7	p_8

Fig. 1. Four- and eight-neighbours of a pixel p.

Definition 3. A subset C of the object pixels is called an eight-connected (resp. four-connected) subset if, for any two distinct pixels $p,q \in C$, there is an eight-path (resp. four-path) in C from p to q.

Definition 4. The maximal eight-connected (resp. four-connected) subsets of the object pixels are called eight-connected (resp. four-connected) components.

In our paper, we have considered the object (denoted by 1) to be eight-connected, and hence the background or holes (denoted by 0) are four-connected.

Definition 5. The boundary of an object is defined by the set of the object pixels, each having at least one non-object pixel as its four-neighbour.

Definition 6. The four-connected component of non-object pixels which contains the top and bottom rows, and right-most and left-most columns of the image, is the background; and any other four-component of non-object pixels is a hole.

Definition 7. A pixel p is called 'critical' if its removal causes any disconnectivity or formation of a hole.

Definition 8. A skeletal leg is a limb of thickness one, with one end not connected to anything and this pixel not connected to anything, is called an end pixel. In other words, the end pixel has exactly one eight-neighbour.

3. The Proposed Model

The input binary image is a 2D array of 0s and 1s. Among various techniques of binary image thinning, the most common technique is the removal of boundary object pixels iteratively. A class of iterative thinning algorithms has used 3×3 thinning templates for the removal of boundary object pixels. The algorithms match (with respect to the centre) these templates against each pixel, and decide whether object pixels are to be removed. Some of these algorithms use multiple passes per iteration, while a few other thinning algorithms use only one pass. One-pass algorithms are faster than multi-pass algorithms, but the latter usually produce better results than the former. Moreover, one-pass algorithms require additional restoring templates to preserve the connectivity.

In the present work, we apply neural networks to perform similar transformations as done by the above algorithms for thinning a binary pattern. We have designed a network that consists of five layers (see Fig. 2). The bottom layer is the input layer. Layers 1, 2, 3 and 4 perform the boundary removal operation. Different layers remove boundary pixels from different directions, namely East, North, West and South. Depending on these directions, we can have four types of boundary pixels. For example, a pixel matching the pattern '110' (with respect to the middle position) is an East-boundary, and so on.

The thinning algorithm proposed here removes the boundary pixels in multiple layers. In each layer it removes a subset of the boundary pixels in parallel, and the output is passed to the next upper layer. A boundary pixel is not removed if (i) its removal alters the connectivity, or (ii) its removal shortens any leg of the skeleton. A pixel that satisfies condition (i) is termed a 'critical pixel', and a pixel satisfying condition (ii) is termed an 'end pixel'. The algorithm used a 3×3 neighbourhood window to prevent the deletion of a pixel which satisfies conditions (i) or (ii). Necessary and sufficient conditions for a boundary pixel to be a critical or an end pixel are established, on the basis of which the algorithm preserves the connectivity and general configuration of the skeleton.

Theorem 1. If p is an East- or North- or West- or South-boundary pixel, then the following are true, respectively:

(a) p is critical if and only if ($p_2=0$ and $p_3=1$) or ($p_7=0$ and $p_8=1$);
(b) p is critical if and only if ($p_4=0$ and $p_1=1$) or ($p_5=0$ and $p_3=1$);
(c) p is critical if and only if ($p_2=0$ and $p_1=1$) or ($p_7=0$ and $p_6=1$);

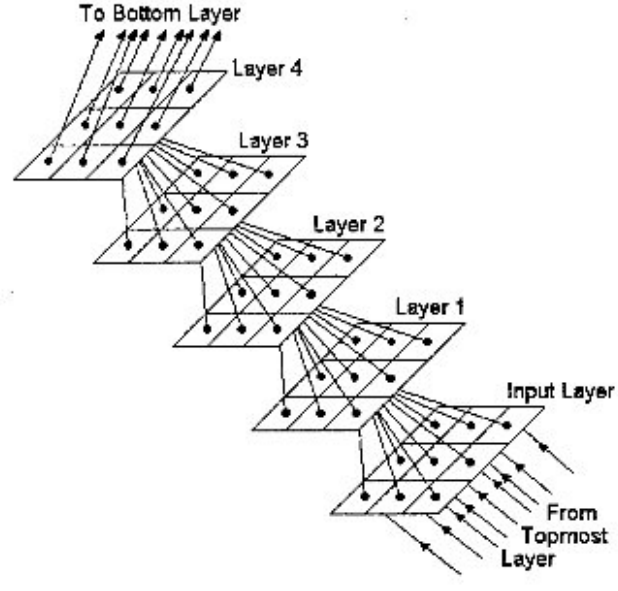

Fig. 2. The proposed network architecture.

(d) p is critical if and only if (p_4=0 and p_6=1) or (p_5=0 and p_8=1).

Proof. Suppose p is an East-boundary pixel. That is, $p_4 = 1$, $p = 1$ and p_5=0. Since p is a boundary pixel, its removal cannot form a hole. Hence, in the present situation, p is critical if and only if its removal causes disconnectivity.

'if' part: suppose p_2=0 and p_3=1. The two object pixels p_4 and p_3 become disconnected when p is removed. Hence p is critical. Similarly, p is critical if p_7=0 and p_8=1.

'only if' part: suppose p is critical. That is, removal of p causes disconnectivity. So, there is a pair of object pixels (p_i, p_j) ($i \neq j$ and $i, j \in \{1, 2, 3, 4, 6, 7, 8\}$) such that p_i and p_j are connected (disconnected) before (after) removal of p. Now note that no such pair exists if $p_2 = p_7 = 1$. Thus, either $p_2 = 0$ or $p_7 = 0$. If $p_2 = 0$, then one of the object pixels p_i and p_j has to be p_3, the other being from $\{p_1, p_4, p_6, p_7, p_8\}$. On the other hand, if $p_7 = 0$, then one of the object pixels p_i and p_j has to be p_8, the other being from $\{p_1, p_2, p_3, p_4, p_6\}$.

Similar arguments hold for the other types of boundary pixels. □

Theorem 2. If p is an East-boundary pixel, then p is end pixel if and only if p_1, p_2, p_3, p_6, p_7, and p_8 are 0 (equivalently, $p_1 + p_2 + p_3 + p_6 + p_7 + p_8 = 0$). Similar conditions hold for the other three types of boundary pixels.

Every layer in the proposed network is constituted by a rectangular array of processors or nodes where each such processor is capable of storing a binary value. Each pixel in the input image is assigned with one processor in every layer. At the beginning, the input image is fed to the input layer. That is, the pixel values (0 or 1) are stored into the respective processors. The layers are connected by massive interconnections. The node in the ith row and the jth column in a layer (except for the input layer) is connected via nine input connections to the node in the ith row and the jth column, and all the nodes that are eight-neighbours (see Definition 1) to it in the immediate lower layer (Fig. 2). As each layer removes the boundary pixels from one of the four directions, the activation functions for all the layers are different. A processor in Layers 1, 2, 3 and 4 computes its output from the nine input values of the immediate lower layer by the activation functions f_a, f_b, f_c and f_d, respectively:

$$f_a(p, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8) \quad (1)$$
$$= pp_4p_5(p_2p_3 + p_7p_8 + p_1p_2p_3p_6p_7p_8)$$

$$f_b(p, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8) = pp_7\bar{p}_2(\bar{p}_4p_1 + p_5p_3 + p_1p_3p_4p_5p_6p_8)$$

$$f_c(p,p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8) = pp_5\bar{p}_4(\bar{p}_2p_1 + \bar{p}_7p_6 + \bar{p}_1\bar{p}_2\bar{p}_3\bar{p}_6\bar{p}_7\bar{p}_8)$$

$$f_d(p,p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8) = pp_2\bar{p}_7(\bar{p}_4p_6 + \bar{p}_5p_8 + \bar{p}_1\bar{p}_3\bar{p}_4\bar{p}_5\bar{p}_6\bar{p}_8)$$

The output values of the topmost layer are fed back to the input layer for the next iteration. The above activation functions can easily be implemented by simple circuits. For example, the circuit in Fig. 3 can be used for activation function f_a.

The Algorithm

Step 1. Assign the given binary image to the array of nodes in the input layer.
Step 2. For all the nodes in Layer 1 compute f_a.
Step 3. For all the nodes in Layer 2 compute f_b.
Step 4. For all the nodes in Layer 3 compute f_c.
Step 5. For all the nodes in Layer 4 compute f_d.
Step 6. Go on repeating Steps 2–5 until no further changes occur in the image pattern, i.e. no further pixels are removed.

3.1. Connectivity

A basic property of a skeleton is preserving the connectivity of the original input pattern. We shall now prove that this property is satisfied by the proposed neural network-based model.

Proposition 1. The output skeleton preserves the connectivity of the input pattern.

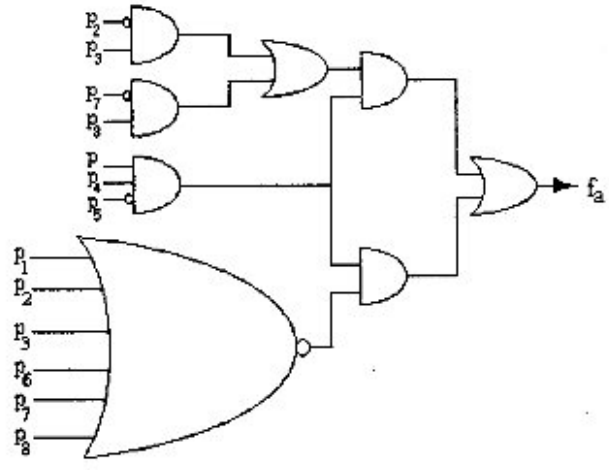

Fig. 3. The circuit for activation function f_a.

Proof. Consider the activation function f_a in Layer 1. The first factor of f_a, that is, $pp_4\bar{p}_5$, is for checking boundary object pixels. The sum of the first and second terms of the second factor, i.e. $p_2p_3 + \bar{p}_7p_8$, is for checking critical pixels; and $p_1\bar{p}_2\bar{p}_3\bar{p}_6\bar{p}_7\bar{p}_8$ is for checking end pixels. Thus the activation functions in Eq. (1) remove (output 0) or retain (output 1) a boundary pixel according to the conditions in Theorems 1 and 2. Hence, the removal process of the boundary pixels in the Layers 1, 2, 3 and 4 posseses the homotopy property (the property that preserves the topology of the input pattern) and does not shorten the skeletal legs. □

3.2. Convergence

We shall now show that the proposed algorithm converges in a finite number of iterations and a skeleton is really achieved.

Proposition 2. The algorithm converges in a finite number of iterations and produces a non-empty skeleton.

Proof. It is easy to see that the activation functions f_a, f_b, f_c and f_d are equivalent to the deletion of boundary pixels. Moreover, only the pixels which are neither critical nor end pixels are deleted. Thus, after a finite number of iterations, only the critical and end pixels will remain, which will constitute the required skeleton. □

3.3. Unit Thickness

In the previous subsections we have established the convergence of the proposed algorithm, and have observed that the basic property of a skeleton, i.e. connectivity, is preserved. Now we shall see that the resulting skeleton is of one pixel thickness everywhere.

Proposition 3. The output skeleton is one pixel thick everywhere.

Proof. It suffices to show that at any pixel of the final skeleton there does not exist any of the 2×2 square patterns, as shown in Fig. 4, unless the pixel is critical. This is obvious, since some of the activation functions will detect these 2×2 patterns and remove the origin pixels unless it is a critical one. □

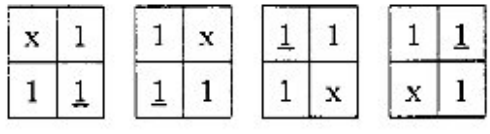

Fig. 4. 2×2 squares (origin is underlined, × means don't care).

4. Experimental Results and Discussion

The proposed technique is tested on several binary patterns. Some results are shown in Fig. 5. In addition to satisfying the basic properties of a skeleton, namely unique thickness and connectivity, the proposed model is found to be more efficient. Experiments have shown that the present model produces results with better medial axis representation compared to quite a few existing conventional thinning algorithms. The following measure η is computed for comparison of the goodness of medial axis representation with a few other thinning algorithms.

$$\eta = \frac{Area\ [S'']}{Area\ [S]} \tag{2}$$

where S = set of all object pixels in the input pattern, and S'' = union of the maximal digital disks (included in S) centred at all the skeletal pixels. $Area[.]$ is an operator that counts the number of pixels. Clearly, η ranges from 0 to 1, and the derived skeleton is identical to the ideal medial axis if η is 1. The measure η measures the closeness of the extracted skeleton to the true medial axis, and hence can be used as an index of medial axis representation. For the proposed algorithm, the average η value is found to be 0.931. Average values of η are computed for a number of conventional algorithms for the same set of test patterns (English alphabets). The results are summarised in Table 1.

The proposed algorithm is also found to be robust to boundary noise. The boundary noise is distributed on the immediate neighbourhood of the boundary of the object. If the original object contains noise, the skeleton should not deviate much from the skeleton of the same object without noise. A serious problem with some of the existing algorithms is that they sometimes produce noisy skeletons if the input patterns contain boundary noise [10,11]. We add boundary noise pixels and study their effect on the skeleton. The robustness of our algorithm to boundary noise is demonstrated in Fig. 5(d). An existing thinning algorithm (for example, the algorithm by Jang and Chin [10]) is found to be noise sensitive (Fig. 5(e)) and generates noise spurs, while our proposed algorithm is insensitive to it (Fig. 5(d)).

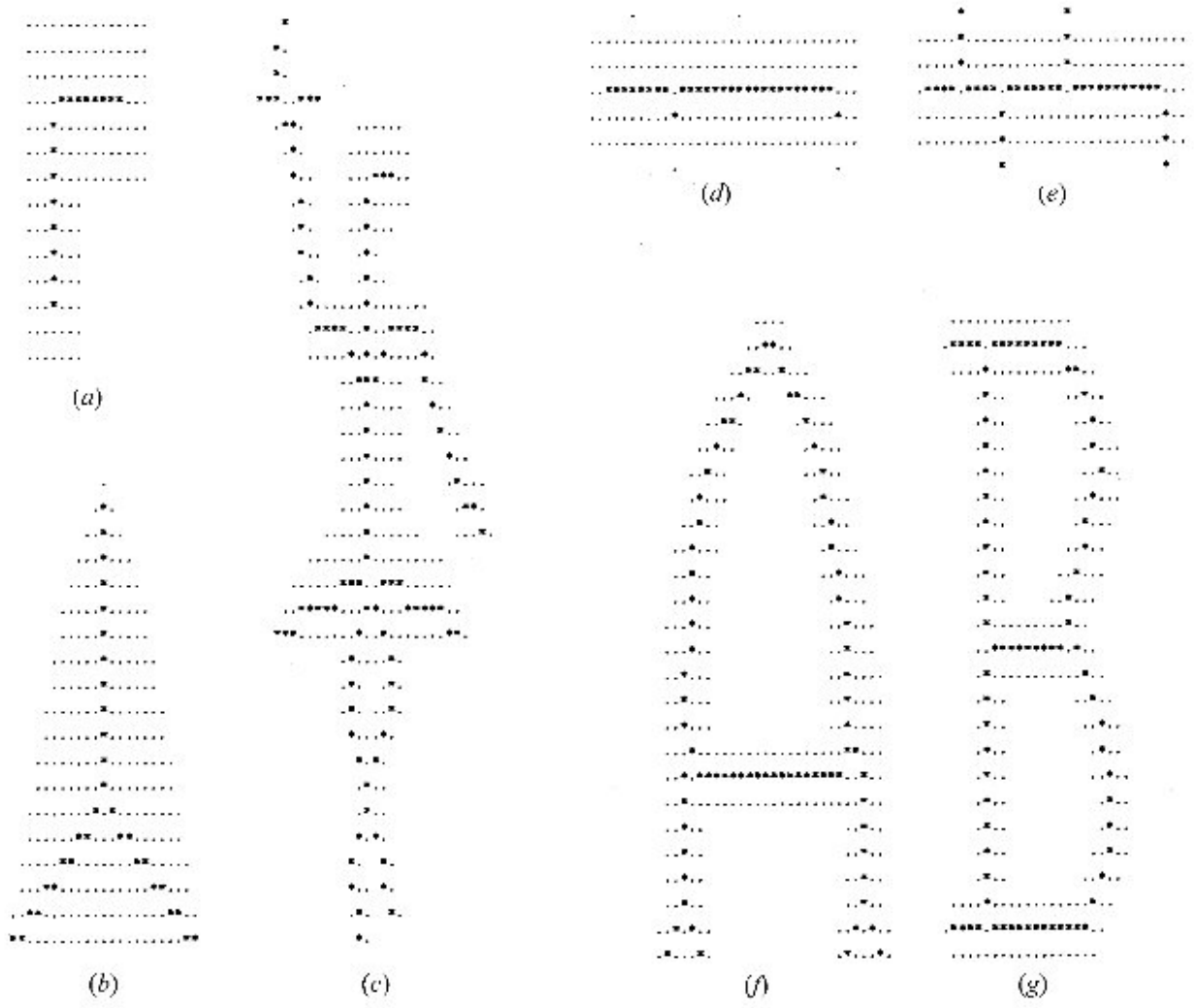

Fig. 5. Results of the proposed algorithm (except (e)). '.' means object and '*' means skeletal pixels; (e) result of Jang and Chin [16].

Table 1. Index of medial axis representation

Algorithms	η
Chin et al. [12]	0.819
Holt et al. [13]	0.891
Hall [14]	0.902
Zhang and Suen [15]	0.886
Lu and Wang [16]	0.898
Arcelli and Baja [17]	0.867
Jang and Chin [10]	0.881
Fan et al. [18]	0.811
Proposed neural algorithm	0.931

5. Conclusions

The present paper proposes an image thinning technique based on a neural network. There are several techniques to thin a binary image, among which the removal of boundary pixels, from different directions, by template matching [9–14] is the most common one. A similar concept is used here in applying neural networks. The network architecture and the activation functions are designed such that neurons at four different layers remove the respective types of boundary pixels. After a finite number of iterations, depending upon the thickness of the object, the network produces the required skeleton of thickness one which preserves the connectivity of the object without shortening any skeletal leg.

Experimentally, the proposed technique is found to be more efficient in medial axis representation. The present model is also found to be more robust to boundary noise in comparison with a few other thinning algorithms. Skeletons are often used as a preprocessing step in Optical Character Recognition (OCR), medical imaging applications, visual inspection of industrial parts, printed circuit boards, engin-

eering drawings, and in several other application areas [19]. So in such applications, the end results may depend significantly on whether the intermediate output (skeleton) is of poor quality or sensitive to noise. For example, in an OCR application, if the thinning algorithm is sensitive to boundary noise, a separate postprocessing step is required to remove the noise spurs, which is time consuming. Otherwise, noisy skeletons would lead to a poor recognition rate. The present algorithm, as it produces a robust skeleton, will be more useful in such situations.

Hundreds of thinning algorithms have been developed [3] during the last few decades. We have compared our results with only a few of them, out of which the results poorer than that of the proposed algorithm are reported here. However, there are algorithms [11] which use templates of a bigger size (5×5) and produce better results than that proposed here, in some respect (for example, noise immunity).

References

1. Rosenfeld A, Kak AC (1982) Digital Picture Processing, Academic Press
2. Blum H (1964) A transformation for extracting new descriptions of shape. IEEE Proc Symp on Models for the Speech and Vision Form, Boston 362–380
3. Lam L, Lee SW, Suen CY (1992) Thinning methodologies – a comprehensive survey. IEEE Trans Patt Anal Machine Intell 14: 869–885
4. Lippmann RP (1987) An introduction to computing with neural nets. IEEE ASSP Magazine 4–22
5. Carpenter GA, Grossberg S (eds) (1992) Neural Networks for Vision and Image Processing. MIT Press
6. Mehrotra K, Mohan CK, Ranka S (1997) *Elements of Artificial Neural Networks*. MIT Press
7. Krishnapuram R. Chen LF (1993) Implementation of parallel thinning algorithms using recurrent neural networks. IEEE Trans Neural Networks 4: 142–147
8. Wang PSP, Zhang Y (1989) A fast and flexible thinning algorithm. IEEE Trans Comput 38: 741–745
9. Datta A, Parui SK (1994) A robust parallel thinning algorithm for binary images. Pattern Recognition 27: 1181–1192
10. Jang BK, Chin RT (1990) Analysis of thinning algorithms using mathematical morphology. IEEE Trans Patt Anal Machine Intell 12: 541–551
11. Jang BK, Chin RT (1992) One-pass parallel thinning: analysis, properties and quantitative evaluation. IEEE Trans Patt Anal Machine Intell 14: 1129–1140
12. Chin RT, Wan HK, Stover DL, Iverson RD (1987) A one-pass thinning algorithm and its parallel implementation. Computer Vision, Graphics, Image Process 40: 30–40
13. Holt CM, Stewart A, Clint M, Perrott RH (1987) An improved parallel thinning algorithm. Comm ACM 30: 156–160
14. Hall RW (1989) Fast parallel thinning algorithms: Parallel speed and connectivity preservation. Comm ACM 32: 124–131
15. Zhang TY, Suen CY (1984) A fast parallel algorithm for thinning digital patterns. Comm ACM 27: 236–239
16. Lu HE and Wang PS (1986) A comment on a fast parallel algorithm for thinning digital patterns. Comm ACM 29: 239–242
17. Arcelli C, Sanniti di Baja G (1985) A width-independent fast thinning algorithm. IEEE Trans Patt Anal Machine Intell 7: 463–474
18. Fan KC, Chen DF, Wen MG (1998) Skeletonisation of binary images with nonuniform width via block decomposition and contour vector matching. Pattern Recognition 31: 823–838
19. Suen CY, Wang PSP (eds) (1994) Thinning Methods and Applications to Pattern Recognitions. World Scientific