Skeletonization by a topology-adaptive self-organizing neural network

Amitava Datta[a,*], S.K. Parui[b], B.B. Chaudhuri[b]

[a]*Computer and Statistical Service Centre, Indian Statistical Institute, 203 B.T. Road, Calcutta 700035, India*
[b]*Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700035, India*

Received 5 May 1999; received in revised form 3 December 1999; accepted 20 December 1999

Abstract

A self-organizing neural network model is proposed to generate the skeleton of a pattern. The proposed neural net is topology-adaptive and has a few advantages over other self-organizing models. The model is dynamic in the sense that it grows in size over time. The model is especially designed to produce a vector skeleton of a pattern. It works on binary patterns, dot patterns and also on gray-level patterns. Thus it provides a unified approach to skeletonization. The proposed model is highly robust to noise (boundary and interior noise) as compared to existing conventional skeletonization algorithms and is invariant under arbitrary rotation. It is also efficient in medial axis representation and in data reduction.

Keywords: Skeleton; Medial axis; Neural networks; Self-organization; Adaptive topology; Robustness

1. Introduction

Skeletonization is the process of transforming a several pixel wide object to a single pixel wide object so that the topological properties of the object are more or less preserved. The resulting object is called a *skeleton*. Such skeletons are useful in the recognition of elongated shaped objects, for example, character patterns, chromosome patterns, etc. The skeleton provides an abstraction of geometrical and topological features of the object. Since it stores the essential structural information of the pattern, skeletonization can be viewed as data compression as well.

The concept of a skeleton (in continuous case, it is called medial axis transformation or MAT) was introduced by Blum [1]. Formally, the medial axis can be defined as follows. Consider the circles that totally lie 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. Extensive researches have been carried out in this area (in digital domain) and as a result, a large number of skeletonization or thinning techniques have been developed in the last few decades. These techniques differ from each other in performance and in implementation. But the main objective is to achieve a close approximation of the MAT of the object. A large number of skeletonization algorithms remove outer layer pixels iteratively until a one-pixel thick skeleton is achieved. Another popular approach uses the distance transform (DT). Several other techniques based on polygon approximation, run-length encoding, contour following, etc., are also available. These techniques are surveyed in a classified manner by Smith [2] and Lam et al. [3]. The output skeleton, produced by some of these algorithms, may not be a raster representation. Rather, it may be in the form of a planar graph providing a line-segment approximation of the input pattern. In this paper, we call such representation as *vector skeleton* as opposed to *raster skeleton* produced by other techniques.

This paper deals with generation of the vector skeleton of an object. For this, we propose a self-organizing neural network model which is topology-adaptive. The neural net grows in size with iterations according to the local topology of the input pattern. When the net converges, the topology of the processors defines the vector skeleton.

Our study shows that the proposed neural algorithm has quite a few advantages over the conventional ones. The most important advantage is high robustness with respect to boundary and interior noise. Second, the output here is rotation invariant to arbitrary angles which is not so for many conventional algorithms. The medial axis representation efficiency is found to be satisfactory. The algorithm is also capable of higher data reduction. Finally, the proposed algorithm can be seen as a unified approach to skeletonization because it is applicable to all the three types of input patterns, namely, binary images, dot patterns and gray-level images.

We describe our neural network model for skeletonization in Sections 2 and 3. Section 4 presents comparative results between the proposed neural algorithm and some conventional thinning algorithms. Conclusions are given in Section 5.

2. The topology-adaptive self-organizing neural network (TASONN) model

Kohonen's self-organizing neural network (SONN) model [4] uses a network of fixed (either linear or planar) topology. The neighbourhood topology in the network is fixed. Such a net of fixed neighbourhood topology does not work well in some situations [5,6]. It is so because during weight-updation process the weight vectors lying in zero-density areas may be affected by input vectors from the surrounding parts of the nonzero distribution. As the neighbourhoods are shrunk, the fluctuation vanishes that makes some processors remain outlier due to the residual effect. Moreover, because of rigid topology of the net, the topology of the input pattern cannot be completely adapted. These pose problems, as discussed in the next section, because the resulting network does not give the required vector skeleton.

From the above issues it is felt that a dynamic network with an adaptive local neighbourhood is required for skeletonization. The model proposed here meets this requirement. In the present model, the initial list of processors is empty and the resulting topology is completely data driven. That is, initially there is no processor and no neighbourhood is defined. The net grows in size by means of a certain processor evolution mechanism. During the learning process the processors create/adapt their neighbourhoods dynamically, by means of connection building, on the basis of the input. The neighbourhood of a processor in the net is totally input driven which gets dynamically defined. The degrees of different processors (number of neighbours) may vary and may increase or decrease over time and hence the neighbourhood topology of the net is not fixed. The topology is adapted on the basis of the local input vectors. The model enables the network to learn the weight vectors as well as its topology from the input vectors in an unsupervised manner. Thus the present model is a *topology-adaptive self-organizing neural network* (TASONN).

A different topology adaptive network was proposed by Martinetz and Schulten [7] but, as we shall see later on, this network is not suitable for generating the vector skeleton of a pattern.

We shall first describe our model in a general case and then demonstrate its applicability to skeletonization. Let the input vectors X come from a manifold defined in the m-dimensional space. Denote the array of processors by $\{\pi_1, \pi_2, \ldots, \pi_n\}$ and the neighbourhood N_i of a processor π_i by $\{\pi_p | \pi_p$ is connected to $\pi_i\}$ which excludes π_i. The processor π_i stores an m-dimensional weight vector W_i which can be considered to be the location of the processor in R^m.

Definition 1. By *sensitive region* of a processor we mean an m-dimensional ball of a given radius centred at the weight vector of the processor. The radius is called the *sensitive level*.

Definition 2. A processor is said to *respond* to an input signal if the input vector falls inside the sensitive region of the processor.

Definition 3. A processor is said to *win* if it is the nearest to the input vector among all the processors. The processor is called the *winner* processor. The processor that is second nearest to the input vector is called the *second winner*.

In the present model, the entire network is evolved. Initially, there is no processor, that is, the set of processors is null. New processors can be added to the network in two manners (which will be discussed in a shortwhile) and we use two terms *create* and *insert* to distinguish between them. Both creation and insertion are performed according to the demand by the input. The former is to take care of new input regions while the latter is to take care of highly dense regions. If some input region is unattended, that is, if no other processor is found around, a new processor is created there. Again, if any input region is highly dense, new processors are inserted there.

An input vector is received through the input lines. The first processor is created at the location of the first input vector. A sensitive level is set and a sensitive region is assigned to the processor. From the second input onwards the process continues as follows. If the presented input falls outside the sensitive regions of all the existing processors (that is, no processor responds) then a processor is created at the location of the input. If at least one processor responds, then (1) adjust the weights of the winner and the second winner; and (2) strengthen the strength variable of the edge connecting the winner and the second winner and weaken that of all other edges.

Several iterations (presentations of input) make one *phase*. One phase is completed when the weight vectors of the current set of processors converge, that is, when the network becomes stable. After every phase, a new processor is inserted at the middle of the link with the maximum strength value. The links for the new processor are rebuilt after removing the old link and the strengths are readjusted (strength of the old link is distributed equally to the two new links). The next input vector is presented and the whole process is continued until certain convergence criterion is met. Formally, the above process can be described as follows.

Processor creation: Suppose the sensitive level is τ. A new processor is created when the input vector $X(t)$, at time t, comes from a new region. A new input region is detected if no processor is found within a distance of τ from $X(t)$. In other words, on presentation of an input, if no processor responds (or if the processor set is empty) with the given sensitive level τ then create a new processor at the location of $X(t)$.

Link construction: Establish a link between two nodes π_u and π_v $(u \neq v)$ if X is an input from $p(X)$ such that

$$\max(d(X, W_u), d(X, W_v)) \leqslant d(X, W_k) \quad \forall k \ (k \neq u, k \neq v),$$

where $d(\cdot)$ stands for distance. The above criterion means, in the iterative process, if some input vector arises which has W_u and W_v as the two nearest weight vectors (one is the nearest and the other is the second nearest) then the nodes π_u and π_v are joined by a link, say, L_{uv}. A strength β_{ij} $(0 \leqslant \beta_{ij} \leqslant 1)$ is associated with every such link L_{ij}. These strengths are updated during learning. Initially, all β_{ij}'s are set to 0. Suppose, the input vector at iteration t is $X(t)$. Let π_u and π_v $(u \neq v)$ be, respectively, the first and second nearest processors of $X(t)$, that is,

$$\max(d(X(t), W_u), d(X(t), W_v)) \leqslant d(X(t), W_k)$$
$$\forall k \ (k \neq u, k \neq v). \tag{1}$$

Then the strengths are updated as follows:

$$\beta_{uv}(t+1) = \beta_{uv}(t) + \frac{1}{t+1}\,(1 - \beta_{uv}(t)). \tag{2}$$

For all other links,

$$\beta_{ij}(t+1) = \beta_{ij}(t) + \frac{1}{t+1}\,(0 - \beta_{ij}(t)) = \frac{t}{t+1}\,\beta_{ij}(t). \tag{3}$$

This is as if $X(t)$ pulls only β_{uv} towards 1 and pulls all other β's towards 0.

Weight updation: After presentation of input $X(t)$ at tth iteration the links are established by condition (1) and the weights are updated by

$$W_u(t+1) = W_u(t) + \alpha_1(t)[X(t) - W_u(t)], \tag{4}$$

$$W_v(t+1) = W_v(t) + \alpha_2(t)[X(t) - W_v(t)], \tag{5}$$

where $\alpha_1(t)$ and $\alpha_2(t)$ $(0 < \alpha_2(t) \leqslant \alpha_1(t) < 1)$ are chosen as in the SONN model.

The asymptotic relation between β_{ij}'s and W_i's will be as follows. Suppose, the asymptotic values of the weight vectors are W_i^*. Consider the Voronoi tiles [7] of the Voronoi diagram of order 2 of the set of weight vectors W_i^*:

$$V_{ij} = \{X \in R^m\colon \max(d(X, W_i^*), d(X, W_j^*))$$
$$\leqslant d(X, W_k^*) \quad \forall k \ (k \neq i, k \neq j)\}.$$

The present model considers pattern distribution $p(X)$ that has support only on a submanifold, not on the entire embedding space R^m. It can be noted that the asymptotic value of the strength β_{ij} is $\int_{V_{ij}} p(X)\,\mathrm{d}X$. For some $V_{ij} \neq \phi$, the integral $\int_{V_{ij}} p(X)\,\mathrm{d}X$ may vanish as a result of which the respective β_{ij} values will tend to zero. Again, it is possible that some β_{ij} can have a positive value at a certain stage of learning although V_{ij} is in fact empty. In both cases, a link introduced during learning will vanish in the limit. By similar arguments given by Martinetz and Schulten [7], the links with asymptotically non-zero β_{ij}'s will form a subgraph of the Delaunay triangulation of W_i^* $(i = 1, 2, \ldots, n)$.

It is to be noted that the asymptotic links alongwith their strengths contain information regarding $p(X)$. The β value, against each link, indicates the degree of existence of the link. For example, for an 'L'-shaped pattern (Fig. 1(a)), there are 13 processors (after convergence) and a link between two processors is shown whenever the corresponding β value is greater than zero. Here $p(X)$ is the uniform distribution over the 'L'-shaped area. The value of β for the diagonal link is 0.015 while the β values for the other links are between 0.070 and 0.080 (for the end processors they are 0.110). Removal of a link with a significantly low (in comparison to others) β value can

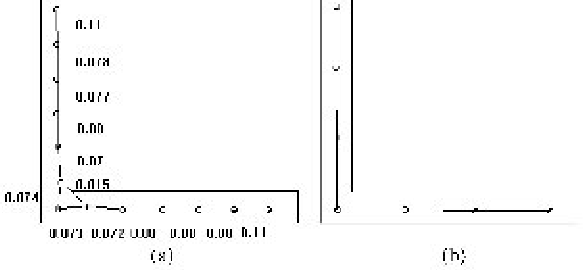

Fig. 1. For 'L'-shaped pattern, the output net achieved by TASONN after convergence: (a) $\delta = 25$, (b) $\delta = 50$. The nonzero β values are shown against each link.

thus result in a network that represents the skeletal shape of the input pattern more accurately. It is to be noted that such undesirable spurious triangles may occur at the junctions of a pattern. They can be avoided by increasing the gaps between two processors (Fig. 1(b)). Alternatively, such spurious triangles can be removed by deletion of links with significantly low β.

Processor insertion: The processors are inserted only after a phase is completed. Suppose, at the end of the sth phase, the weight vectors of the processors are $W_1(t_s), \ldots, W_{n(s)}(t_s)$ where $n(s)$ is the number of processors during the sth phase and t_s is the total number of iterations needed to reach the end of the sth phase. Now, the stopping criterion can be set depending upon the application in hand (see the next section). If the stopping criterion is not met then a new processor π_I is inserted between π_k and $\pi_{k'}$, where

$$\beta_{kk'} = \max_{i=1,\ldots,n(s)} \max_{\pi_{i'} \in N_i} \beta_{ii'}. \qquad (6)$$

Set $\beta_{Ik} = \beta_{Ik'} = \frac{1}{2}\beta_{kk'}$, new $\beta_{kk'} = 0$ and the weight of $\pi_I = \frac{1}{2}[W_k(t_s) + W_{k'}(t_s)]$.

By this process a processor is inserted at a location where the demand is maximum. The process continues until, at the end of a phase, the stopping criterion is met.

The TASONN Algorithm

Step 1: Initialize $t = 0$.

Step 2: Present the input pattern $X(t)$.

Step 3: If some processor responds to $X(t)$ then go to Step 5.

Step 4: Create a processor at the location of $X(t)$.

Step 5: If the second winner exists then update the strengths β_{ij}'s according to Eqs. (2) and (3).

Step 6: Update the weights of the winner and the second winner, according to Eqs. (4) and (5).

Step 7: If the net is not stable, set $t = t + 1$ and goto Step 2.

Step 8: If the stopping criterion is met, then go to Step 10.

Step 9: Insert a processor according to condition (6), set $t = t + 1$. Goto Step 2.

Step 10: Stop.

We shall now see how the above model can be used in skeletal shape extraction. The main feature of the TASONN model, useful in skeletonization, is the dynamic topology adaptivity and its dynamic growth in size.

3. Skeletonization by TASONN model

To make the propopsed TASONN model applicable to skeletonization problem, a suitable stopping criterion is used. A parameter δ is introduced as an upper bound of the distances between two neighbouring processors. The problem of skeletal shape extraction of two-dimensional visual patterns, images and dot patterns, are considered together here. Here, the set of input vectors is $S = \{P_1, P_2, \ldots, P_N\}$ where P_j represents the positional co-ordinates of an object point. One presentation of all the points in S makes one *sweep* consisting of N iterations. After one sweep is completed, the iterative process for the next sweep starts again from P_1 through P_N. With the present stopping criterion, the processor insertion step (discussed in the previous section) is set as follows.

Processor insertion: Processors are inserted after each phase. A phase is completed when the network with the current set of processors converges, that is, when

$$|W_i(t) - W_i(t')| < \varepsilon \quad \forall i,$$

where t and t' are the iteration numbers at the end of two consecutive sweeps and ε is a predetermined small positive quantity. If

$$|W_k(t_s) - W_{k'}(t_s)| = \max_{i=1,\ldots,n(s)} \max_{\pi_{i'} \in N_i} |W_i(t_s) - W_{i'}(t_s)| > \delta$$

then one processor is inserted between π_k and $\pi_{k'}$. The weight of the new processor and the new β values are set as before. After the insertion of a processor, the next phase starts with the new set of processors. The process continues until, at the end of a phase,

$$\text{for all } i, |W_i(t_s) - W_{i'}(t_s)| \leqslant \delta, \quad \forall \pi_{i'} \in N_i$$

The output network of the algorithm gives an approximate global shape of the input pattern. The final network obtained by the above algorithm gives a vector skeleton for the given input pattern. The raster skeleton can easily be derived from the network as follows. It should be mentioned that the output net depends on the parameter δ, and with a proper choice of δ, one can get a satisfactory vector skeleton. This issue is discussed in Section 3.1.

Procedure raster-skel: For each link, the line segment connecting the weight vectors of the two corresponding processors, is considered. The set of all object pixels intersecting such a line segment gives the raster skeleton.

Experimental results on various input patterns have shown that the above algorithm converges and the resulting network gives an approximation of the skeleton of the input pattern. The following discussions would help us to see how the resulting net gives an approximation of the skeletal shape of the pattern.

For arc patterns, it can easily be seen that the above node joining criteria would join the processors by single links (other link strengths are zero) so that the topology of the input is preserved (for example, see Fig. 2(a)). Here two processors corresponding to the regions S_2 and S_3 are joined since they are the two nearest processors from the input vectors lying in the shaded region. Similarly, other links are established. The strengths of all other possible links are zero since no input vector contributes to them. For a node representing a fork

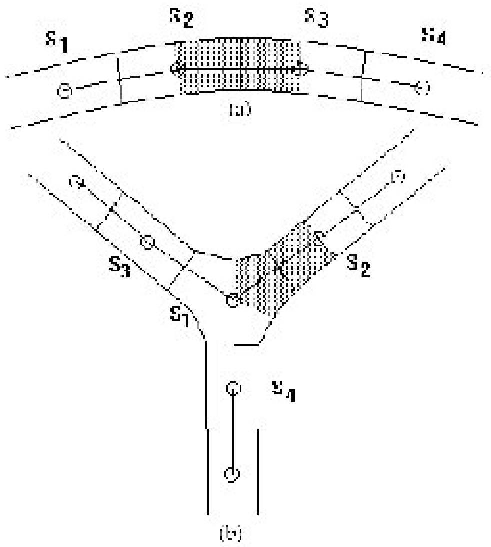

Fig. 2. Link establishments in (a) arc pattern, (b) fork pattern.

junction region S_1 (Fig. 2(b)) the node pairs corresponding to the region pairs (S_1, S_2), (S_1, S_3) and (S_1, S_4) are joined for the same reason.

3.1. Role of parameter δ

The parameter δ controls how densely the processors are to be located in the network. It controls the gap between two neighbouring processors. A desirable skeleton may not be achieved if δ is too high or too low. If δ is very high then the skeleton, although may preserve the essential topology of the pattern, does not properly represent the medial axis (Figs. 3(a) and (b)). A portion of the output skeleton may lie outside the object. On the other hand, very low values of δ might produce spurious triangles in the network (Figs. 3(e) and (f)) which does not represent the true skeletal shape of the pattern. Hence we require some adaptive mechanism to avoid this situation so that a satisfactory medial axis representation can be obtained. One such mechanism is to introduce *activation level* in the weight updating process which is described below.

We specify an *activation region* of a processor so that if an input vector falls within the region then only it activates the processor. The activation region decreases over time. In the present problem it is defined as follows.

Definition 4. The activation level $a_i(s)$ of ith processor, for $i = 1, \ldots, n(s)$, at the end of the sth phase, is defined as

$$a_i(s) = \frac{1}{c_i} \sum_{\pi_r \in N_i} |W_i(t_s) - W_r(t_s)|$$

where c_i is the number of neighbours of π_i excluding π_i.

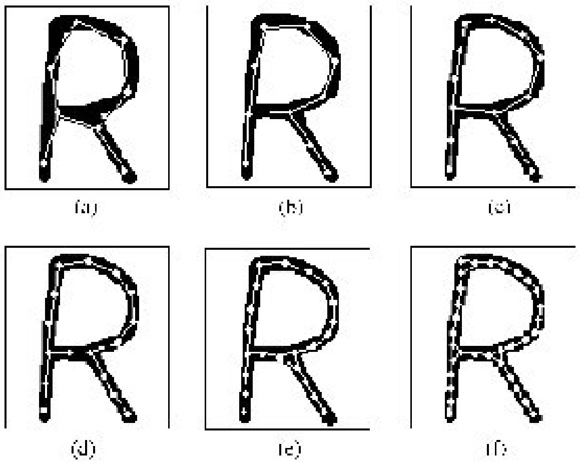

Fig. 3. Output nets for a 'R'-shaped pattern when (a) $\delta = 30$, (b) $\delta = 25$, (c) $\delta = 20$, (d) $\delta = 15$ (e) $\delta = 10$, (f) $\delta = 5$ without activation level.

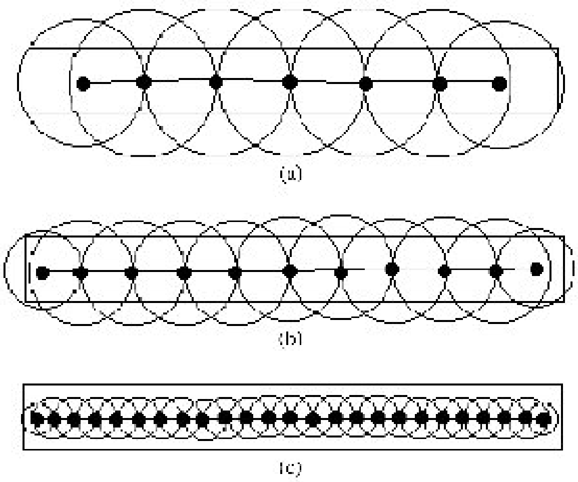

Fig. 4. Activation regions (a) at time t_1, (b) at time t_2, (c) at time t_3. $t_1 < t_2 < t_3$.

Definition 5. The activation region of a processor is a circle with the corresponding weight vector as the centre and the activation level as its radius (see Fig. 4).

It should be mentioned at this stage that the sensitive level and activation level, although similar, are used in different contexts and hence are given different names. The sensitive level remains fixed over time and is used to create the initial set of processors. It is not used any more unless a new input region occurs. The activation level varies over time and may not be the same for all the processors at a point of time. It is used in competition and weight updating.

A processor is called *active* if the presented input vector lies within its activation region. In other words, if an input vector lies outside all the activation regions, it is ignored in the competition and updating process. An input must activate a processor first before entering into the competition. Only active processors are qualified for competition after which the winner processor is selected and the weight vectors are updated accordingly. Thus, the weak signals (here the object pixels near the boundary of the object) gradually become ineffective in the weight updating process. This is so because, as the processors become more and more close to each other, the region of input vectors that activates (influences) a processor is also shrunk and thereby the influence of the outer layers are symmetrically decreased. The object pixels near the medial axis aquires more control over the process (Fig. 4(c)).

It should be noted that the sensitive level is also taken as a parameter in the TASONN model. But, since it is used to detect new regions and to select the initial few processors, its choice is not crucial. Within a wide range of the sensitive level, the model produces identical results.

Although the activation level overcomes the formation of spurious triangles in linear or arc segments of the pattern, it cannot overcome the problem completely at junctions of the pattern. If two such segments meet at a small angle then spurious triangles may occur even after incorporating activation levels. But this situation can be avoided by keeping δ high as seen in Fig. 1.

With these observations, we implement the TASONN model for skeletonization as follows. The whole process is divided into two stages (a) finding an initial skeleton to get the overall topology of the pattern and (b) arriving at a more accurate final skeleton after incorporating the activation level. In the first stage, set δ high and get an initial skeleton that does not contain any spurious triangle but preserves the topology of the input pattern (Figs. 3(a)–(d)). Note that higher the value of δ, lower is the chance of forming a spurious triangle. If there is a spurious triangle then one of the following actions can be taken: (i) remove a link by choosing a threshold on the β values as described in Fig. 1 and (ii) take a higher value of δ and repeat the process to get an initial skeleton.

By the above method, we get an initial skeleton free of spurious triangles that reflects the overall topology of the pattern but may not be close to the medial axis. Now, in the second stage, to get a close medial axis approximation, a lower value is assigned to δ and the weight updating process is continued. As the topology of the input pattern is already learned, the strengths (β) are ignored at this stage. That is, we do not develop any more connections or links in the second stage. We only update the weight vectors in an effort to position them more accurately along the medial axis. New processor insertion is continued as before. For example, the output networks for the 'R'-shaped pattern, using activation level, are shown for different δ values in Fig. 5. It can be

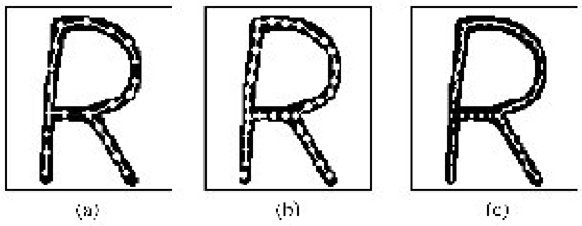

Fig. 5. Output nets for a 'R'-shaped pattern when (a) $\delta = 10$, (b) $\delta = 5$, (c) $\delta = 3$ with activation level.

observed that these results give satisfactory skeletal shape of the pattern even for small δ's (compare with Fig. 3 where activation level is not used).

An important question is: what value of δ should we choose in stage (a)? Experiments have shown that this choice can be made from a considerablly wide range and hence one need not guess an exact value. For example, in Fig. 3, for an 'R'-shaped pattern, the essential topology could be achieved for $15 \leqslant \delta \leqslant 30$. Within this range, the initial skeletons have the same topology. In many situations, the initial skeleton itself (for example, Figs. 3(a)–(d)) serves the purpose. For example, skeletonization is applied as a preprocessing step in character recognition problems. In character recognition, a vector skeleton of the character pattern makes the recognition task easier. This vector skeleton need not be very close to the medial axis. Crude vector skeletons similar to that shown in Figs. 3(a)–(d) are good enough for structural analysis and recognition of the input pattern.

Moreover, for printed and hand-printed character recognition (for a given font, fixed pen thickness and scanner resolution) we can learn, by trial and error method, an estimate of δ from a number of training characters so that the initial skeleton itself is satisfactory (in other words, the initial skeleton provides the essential topology of the input pattern). This estimate can be subsequently used in stage (a). However, if one wants to get more accurate skeleton, he can go for the second stage mentioned above.

As mentioned earlier, a topology-adaptive model, "neural gas network" (NGN) model, was proposed by Martinetz and Schulten [7]. Fritzke [8] modified the NGN model and suggested a scheme to insert processors in the netwok. Thus in the "growing neural gas network" (GNGN) model of Fritzke, the number of weight vectors to start with need not be known a priori. The models NGN and GNGN are found to be effective methods for topology learning. They are advantageous over the SONN model because the SONN model assumes a predefined topology of the network that remains fixed throughout learning. However, the NGN and GNGN models are not readily useable in the extraction of a vector skeleton. Experiments show that in extraction of the skeleton of a pattern, the TASONN model produces much better results than the SONN, NGN and GNGN models. This is demonstrated in Fig. 6. In Fig. 6, it can be seen that the output nets may form mesh-like structures rather than a vector skeleton in SONN and NGN models. However, in GNGN model, this problem can be avoided by keeping the number of processors small (for which one needs to know the optimum number a priori). For example, the GNGN model may produce a skeleton as good as that of the TASONN model if the number of processors is optimally chosen (here 10). The proposed TASONN model gives an adaptive way (using activation region) to avoid this type of situations.

Second, in NGN model, weights are updated for the r nearest processors. Initially, the value of r is set high so that the chance of getting stuck at a local minimum is reduced. Ordering of the r distances, for each presentation of the input, makes the model computationally expensive.

Third, in NGN and GNGN models, the edge-destruction mechanism is based on an aging scheme. Connections made at an early stage of the adaptation may not be valid anymore at an advanced stage. The authors develop an age counter for every link after the link is constructed. If the age of a link exceeds a predefined "lifetime" then the link is destroyed. The lifetime is chosen subjectively here. This technique may pose problems in certain situations. In the proposed TASONN model, the aging of the links is done differently and the link adaptations are done more efficiently. In NGN and GNGN models, a link is constructed/destroyed in a sudden manner. A link is constructed between two processors as soon as an input arises for which these two processors are the closest and the second closest. Similarly, a link is destroyed as soon as its age exceeds the lifetime. This may cause sudden loss of previous learning. Note that if the lifetime is small then the previous learning may be lost soon when input keeps on coming from a new region [5]. If the lifetime is big then some undesirable links may survive which should not. So the choice of the lifetime which is done manually is crucial. In our model, the links are not constructed or destroyed suddenly. Instead, it assigns a strength variable to each edge and these strengths are adapted (lying between 0 and 1) gradually in the learning process. Thus it does not forget the previous learning in a sudden manner. Moreover, in Kohonen's SONN model and in NGN/GNGN model, a link between two processors either exists or does not exist (deterministic). But in TASONN model a link is allowed to have a kind of degree of its existence in the form of β and this can be of use in shape analysis (as explained in Fig. 1).

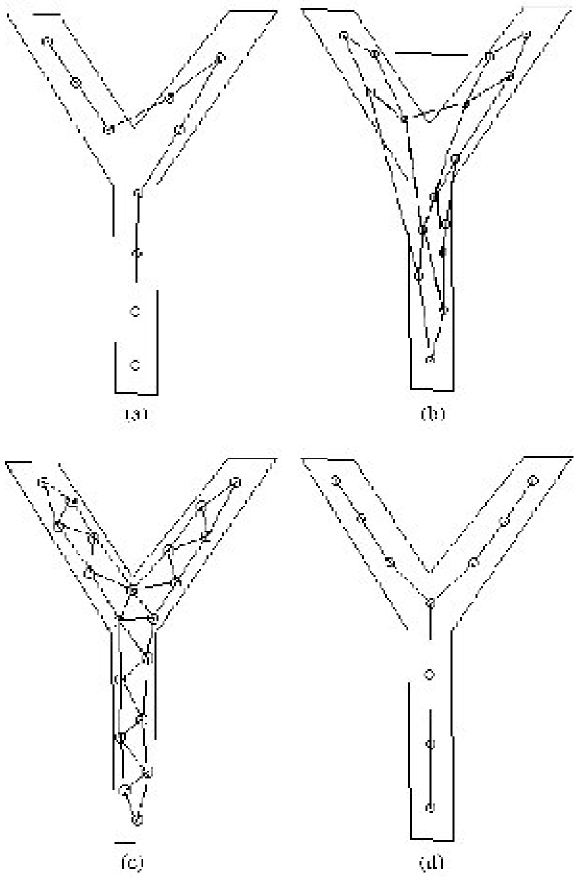

Fig. 6. Responses of different models for Y-shaped pattern. (a) SONN model using linear topology, (b) SONN model using rectangular topology, (c) NGN/GNGN models, (d) TASONN model (only the links with nonzero β are shown).

Finally, in NGN model, all the processors are created initially at rondom positions. This may lead to a number of processors being dead and finally coming to no use. The GNGN model overcomes this problem by taking only two processors initially and then by growing the size of the network. The present model starts with an empty set of processors and processors are created where the input demands them. Thus a new input region can be adapted easily and more efficiently. Moreover, in GNGN model, new processors are inserted at constant (decided manually) time intervals while in our model insertions are done when the current network stabilizes. After the insertion the learning again continues.

4. Performance evaluation of the proposed algorithm

Performance of the proposed neural algorithm is analysed in comparison with some existing conventional thinning algorithms. The comparisons are carried out with respect to the following desirable properties of a skeletonization algorithm: (a) robustness or noise immunity; (b) medial axis representation; (c) rotation invariance; (d) data reduction efficiency; (e) extendibility to other input types (e.g., *dot patterns* and *gray-level patterns*). These properties are discussed in different subsections below. The response of our algorithm with respect to these properties are also presented.

4.1. Robustness

Two types of noise namely, boundary noise and object noise, are considered here. An important aspect of our skeletonization algorithm is noise immunity which makes it qualitatively different from the conventional ones. If the original object contains noise, the skeleton should not deviate much from the skeleton in the ideal situation. A serious problem with many of the existing algorithms is that they sometimes produce noisy skeletons if the input patterns contain noisy boundary (see [9,10]). Moreover, these algorithms cannot handle object noise. On the contrary, our algorithm can take care of both types of noise.

Justifications of the above claim are as follows. The resulting skeleton here is given by the weight vectors after convergence, and their links. Its final position is highly insensitive to noise pixels because of two factors. First, the weight vector converges to the centre of gravity of the respective Voronoi region (S_i) and this centre is not greatly affected by a few noise pixels. Second, the activation region of a processor decreases over time and as a result, the boundary noise pixels are kept outside it to a great extent. Thus, the noise insensitivity of the present algorithm is clear from its learning mechanism and convergence property. Most of the existing conventional algorithms use a rigid definition of connectedness of the object — which in effect causes noise sensitivity. The proposed neural approach relaxes the concept of connectivity and it is found that such a relaxation is useful for robustness even when SNR (signal-to-noise ratio) is close to 1.

4.1.1. Boundary noise

The proposed skeletonization algorithm is found to be highly robust to boundary noise. The boundary noise is distributed on the boundary of the object (white noise) and on its immediate neighbourhood in the background (black noise). Here we add black and white boundary noise pixels and study their effect on the skeleton. We have experimented on several example patterns with different SNR values where the SNR is defined as follows:

$$SNR_B = \frac{\text{Number of boundary black pixels}}{(B+W)},$$

where B is the number of black noise pixels and W the number of white noise pixels. The robustness of our algorithm to boundary noise is demonstrated in Fig. 7. A conventional iterative thinning algorithm (for example, the algorithm by Jang and Chin [11]) is found to be noise sensitive (Fig. 7(b)) while our proposed algorithm is insensitive to it (Fig. 7(c), (d)).

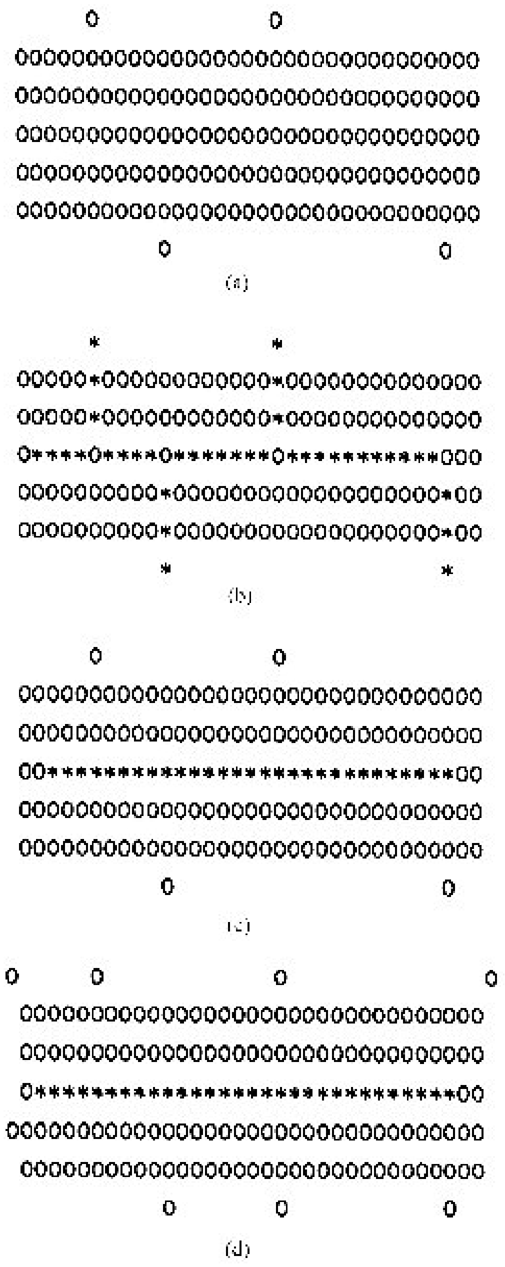

Fig. 7. Robustness to boundary noise, an illustration. '0' represents object and '*' represents skeletal pixel: (a) a line segment with four boundary noise pixels, (b) result of a conventional iterative thinning algorithm, (c)–(d) results of our neural algorithm with the same noise and higher noise.

An error measure, as given below, has been suggested by Jang and Chin [10]:

$$m_e = \min\left\{1, \frac{Area\,[S_K - S'_K] + Area\,[S'_K - S_K]}{Area\,[S_K]}\right\},$$

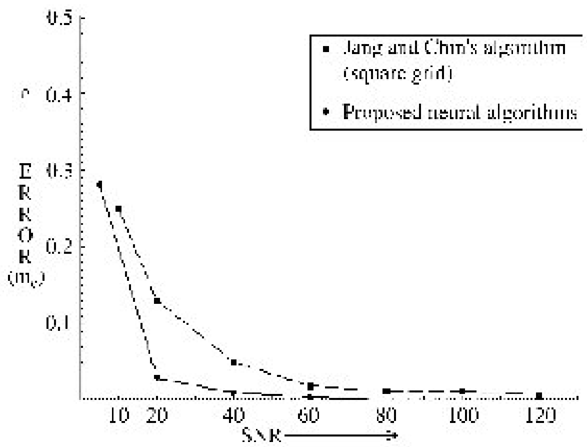

Fig. 8. Boundary noise immunity.

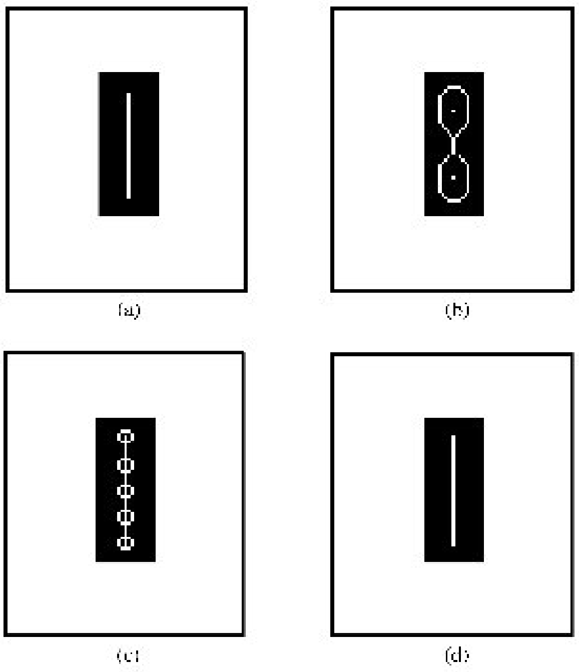

Fig. 9. Output skeleton generates big holes in the presence of single-noise pixels. Output of conventional thinning algorithms (a) without any noise; (b) with two single-noise pixels; (c) vector skeleton by the proposed algorithm with the same noise as in (b); (d) raster skeleton by the proposed algorithm with the same noise.

where S_K and S'_K are the skeletons obtained from S and its noisy version, respectively. $Area[.]$ is an operator that counts the number of pixels.

The average values of m_e against different SNR values are computed for comparing the algorithms. Using the above measure, Jang and Chin's algorithm [10] has been found to be superior to several other conventional algorithms [12–16] in terms of boundary noise immunity. The average m_e values, in our proposed algorithm, are found to be more encouraging (see Fig. 8) and reflect higher robustness to boundary noise.

4.1.2. *Object noise*

Many conventional algorithms are not able to handle noise which is interior to the object. By object noise we mean the white noise distributed over the entire object including its boundary. Such noise may occur in practice due to several reasons. The conventional algorithms (particularly, the iterative ones) use the property of local connectivity within a small window (3×3 or 5×5) and try to preserve such local connectivities throughout. They treat a single white noise pixel as a hole consisting of a single pixel. As a result, in the output skeleton it produces a big hole (Fig. 9(b)). Fig. 9 shows how only two noise pixels misclassify a '1'-like pattern as an '8'-like pattern. On the contrary, the proposed algorithm uses the connectivity concept in a more general sense (note that two processors are joined by a link if the two respective regions are adjacent). The algorithm treats small holes as white noise at the cost of a possibility of missing a true small hole. Thus, very small holes have hardly any effect on the resulting skeleton (Figs. 9(c) and (d)). But if the hole is large enough and is a part of the pattern (for example, consider an 'A'-shaped or 'R'-shaped pattern), it is output as a hole in the resulting skeleton. We have experimented and found that the algorithm is robust and performs satisfactorily with moderately low SNR (moderate amount of noise) where the SNR is defined by

$$SNR = \frac{\text{Number of object pixels}}{\text{Number of noise pixels on the object}}.$$

For a very low SNR, that is, for a very high amount of noise, the binary object becomes merely a set of scattered pixels or a dot pattern. The proposed algorithm can still produce the global skeletal shape of the pattern assuming the noise to be uniformly distributed.

An illustration is given in Fig. 10 for the pattern 'A' with different SNR values. We have taken a very high amount of object noise and tested the algorithm for several character patterns. It has been found that even in the presence of very high noise ($SNR = 2.0$, 1.5 and 1.1), the proposed algorithm is able to extract the skeletal shape of the original object as can be seen in the example figure (Fig. 10). The existing conventional algorithms fail in such situations.

4.2. *Medial-axis representation*

The primary objective of skeletonization is to approximate the medial axis of the object pattern. So it is

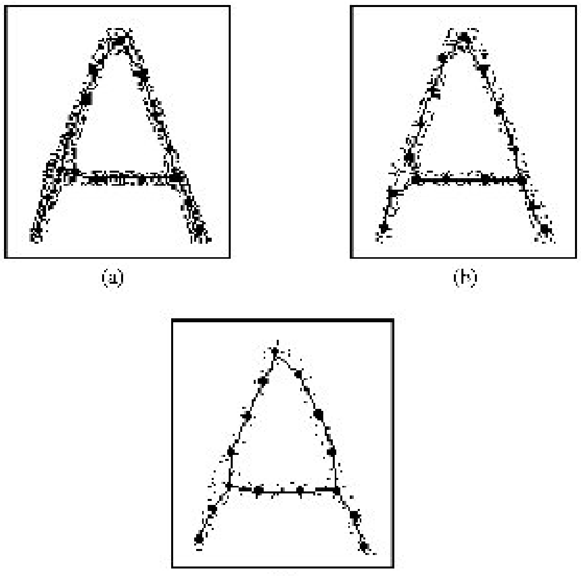

Fig. 10. The final skeletons obtained for the pattern 'A' with object noise. (a) $SNR = 2$, (b) $SNR = 1.5$, (c) $SNR = 1.1$.

important that the output skeleton should approximate the medial axis as closely as possible. After getting the raster skeleton as mentioned earlier, the following measure [11] is computed for comparison of the goodness of fit of medial axis representation with some other thinning algorithms:

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

where S is the set of all object pixels in the input pattern, S'' is the union of the maximal digital disks (included in S) centred at all the skeletal pixels.

Clearly, η (lying between 0 and 1) measures the closeness of the output skeleton to the ideal medial axis. The derived skeleton is identical to the ideal medial axis if η is 1.

For the proposed algorithm, the average η value is found to be 0.890 (with $\delta = 3$). Average values of η are computed for a number of conventional algorithms for the same set of test patterns. The results are summarized in Table 1. It is found that for the proposed algorithm, the medial axis representation index is as satisfactory as in the conventional algorithms.

4.3. Rotation invariance

It is easy to see that, in the proposed algorithm, the output skeleton is invariant under rotation of the input pattern by arbitrary angles. This is due to the facts that the proposed method does not assume any underlying grid and that the measure of closeness is done in terms of Euclidean distance. The iterative methods for skeletonization that use square grid are usually invariant under rotation by multiples of 90° only. The work due to Jang and Chin [10], based on derived grid, reports invariance under 45° rotation also. As an illustrative example, we have rotated the pattern 'X' by different angles (45°, 22° and 11°) and shown the output skeletons in Fig. 11. It shows hardly any effect of rotation of the input pattern on the output skeleton.

Table 1
Index of medial-axis representation

Algorithms	η
Holt et al. [12]	0.891
Hall [13]	0.902
Chin et al. [14]	0.819
Zhang and Suen [15]	0.886
Lu and Wang [16]	0.898
Arcelli and Sanniti di Baja [17]	0.867
Jang and Chin [11]	0.881
Fan et al. [18]	0.811
Proposed neural algorithm	0.890

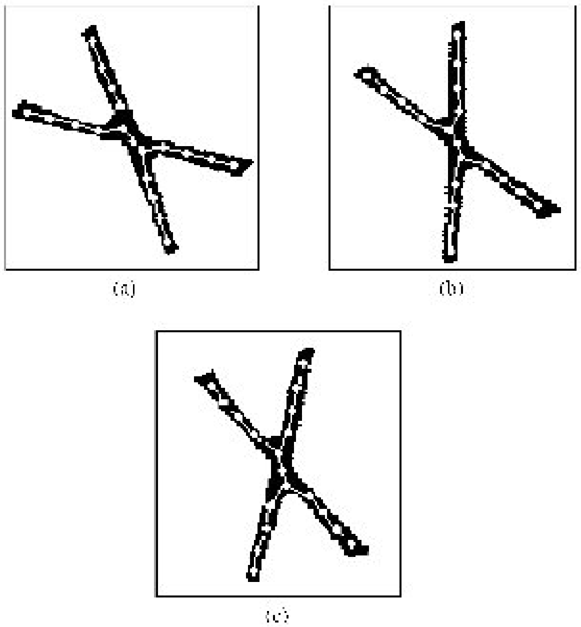

Fig. 11. Effect of rotation by angles: (a) 45°, (b) 22°, and (c) 11°.

4.4. Data reduction efficiency

After convergence, the proposed neural network model creates an adaptive vector quantization [4] of the input set. Each weight vector tends to the centroid of the respective Voronoi regions. Within each region all pixels have the same weight vector as their nearest one. In general, the output weight vectors give the prototypes or examplar vectors from the corresponding regions and these can be an encoded version of the input in less storage space. In the present algorithm, the set of weight vectors along with their interconnections, or the graph (planar straight line graph) with the weight vectors as its nodes and the interconnections as its edges, represents a vector skeleton of the input pattern. This skeleton requires much less space than the original input set and hence a considerable data reduction is achieved.

One of the basic purposes of skeletonization is to reduce the storage space required to store the image data without losing the essential structural information. The proposed method can achieve more data reduction compared to most of the existing skeletonization algorithms. It can be seen that the less the number of processors in the network the more is the data reduction. By choosing larger value of δ, we can make higher data reduction. But this might worsen the accuracy of medial axis representation (see Fig. 3). A proper choice of δ as mentioned earlier can balance this trade-off.

4.5. *Extendibility to dot patterns and gray-level images*

Unlike binary images, a skeleton cannot be properly defined for a dot pattern. But human visual system can still perceive its skeletal shape and extract the *perceptual skeleton* from a dot pattern. For example, a dot pattern having a definite shape can be recognised by the human brain almost as easily as a binary image having the same shape. The conventional thinning algorithms that extract skeletons from binary images do not work for dot patterns. On the contrary, as already seen, the proposed neural algorithm can be used to extract the perceptual skeleton of a dot pattern (see Fig. 10).

Another advantage of the proposed algorithm is that it can take care of gray-level patterns also. The conventional binary image thinning algorithms work only on binary images. They do not work on gray-level images. On the other hand, the neural technique discussed here works on gray-level images also. In case of gray-level images the area of interest (i.e., the object portion) can be interpreted as a multi-valued foreground emerging from a single-valued background (see [19]). Suppose for a gray-level pattern, g_{ab} is the gray value of the pixel at the ath row and bth column. If the weight update rules (4) and (5), are rewritten as

$$W_u(t+1) = W_u(t) + \alpha_1(t)[X(t) - W_u(t)]g_{ab},$$
$$W_v(t+1) = W_v(t) + \alpha_2(t)[X(t) - W_v(t)]g_{ab},$$

then the TASONN model takes care of gray-level patterns as well. It is to be noted that if $g_{ab} = 0$ or 1 (for all

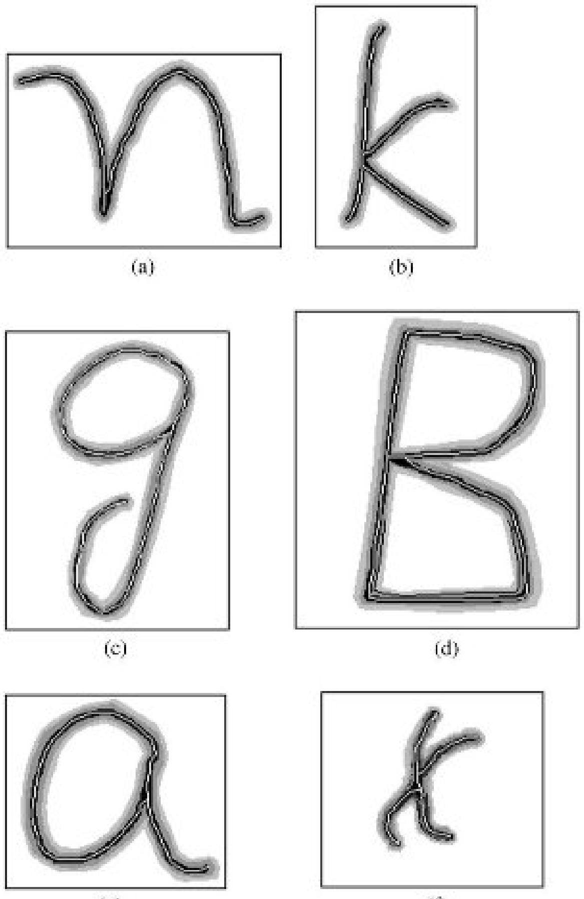

Fig. 12. Output raster skeletons of gray-level patterns.

a, b) then the above update rules are equivalent to rules (4) and (5). The algorithm has been tested on several gray-level images. Fig. 12 shows the results of such examples. For all the test patterns in this paper, the initial value of α is taken as 0.01 and the initial weight vectors are chosen at random. The values of α are changed over time as $\alpha_1(s) = 0.01/(1 + s/50)$ and $\alpha_2(s) = 0.01/(1 + s/20)$, where s is the sweep number.

4.6. *Computational complexity*

In general, finding skeletons is a computationally expensive task. That is why parallel algorithms have been encouraged in this field. The proposed neural network-based method is parallel in nature (in general, neurocomputing has been considered to be a new form of parallel computing). In the proposed skeletonization algorithm, the input vectors are presented to the network sequentially. All the processors present in the network compute the distances from their respective weight vectors in parallel in a constant time and the winner is selected by a 'maxnet' [20]-like network. The winner and the second winner processors update their weights simultaneously. Thus for a single input, the time taken by the above process is not dependent on the network size. This process is repeated for all the input vectors. Hence one complete pass of the input, that is, one sweep requires $O(N)$ time where N is the total number of object pixels. It can be seen that the total number of sweeps does not depend on the input size N.

It should be mentioned here that in many existing parallel thinning algorithms multiple processors are used where all the processors work in parallel. But in these algorithms, all the input values are fed together in an iteration and all the processors (or a large subset of them) compute their output in parallel which would be the input in the next iteration. The total number of passes that are required by this class of algorithms is $O(width_{max})$ where $width_{max}$ is the maximum width of the input pattern. These algorithms use *cellular networks* where each pixel in the image is assigned a processor. Thus the number of processors required is $O(N_I)$ where N_I is the total number of pixels in the whole (including the background) image. In our algorithm, the size of the network is $O(N_S)$ where N_S is the number of skeletal pixels. In most of the applications, $N_I \gg N_S$.

5. Discussion and conclusion

A topology-adaptive self-organizing neural network model is proposed which is applicable in skeletal shape extraction of an object. In Kohonen's SONN model, the network topology is initially set and it is maintained throughout. In the SONN model, the network is self-organizing in that it tends to approximate the input pattern space in an orderly fashion without any supervision. In the proposed model the same is achieved and, in addition, the net topology is adapted automatically unlike in the SONN model. The topology adaptivity is a major issue of the proposed model which makes it applicable to skeletonization tasks. Due to a fixed network topology set initially, Kohonen's model cannot always properly represent the shape of a pattern while the proposed model can do it more accurately. The proposed model evolves the topology of the network and it is self-organizing as well. Moreover, the model is especially tuned to generate a vector skeleton (from which a raster skeleton can also be obtained) of the underlying pattern.

As a skeletonization algorithm, the present model possesses quite a few advantages over the existing conventional thinning algorithms. The most important one is the high robustness of the present algorithm with respect to boundary and interior noise. Next, it is rotation invariant to arbitrary angles while most of the conventional algorithms are not. The medial-axis-representation efficiency is found to be satisfactory. The algorithm is also

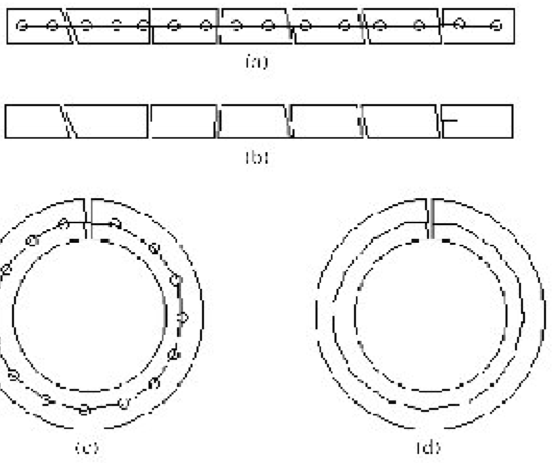

Fig. 13. (a), (c) The vector skeleton misses very small gaps. (b), (d) However, the raster skeleton is alright.

capable of higher data reduction. Finally, the proposed algorithm can be seen as a unified approach to skeletonization because it is applicable to all the three types of input patterns, namely, binary images, dot patterns and gray-level images.

Since the proposed technique is very noise insensitive, it may treat very small holes or gaps as noise. Thus the proposed model cannot strictly guarantee homotopy property of the object (see Fig. 9). Consider a broken line pattern. By our algorithm, the vector skeleton may miss the narrow gaps and recognize it as a single object (Fig. 13(a)). Similar situation will occur in a ring-like pattern with a narrow gap (Fig. 13(c)). This, however, does not cause any serious problem since the raster skeleton will be broken as found in Fig. 13(b) and (d).

Lastly, an important advantage of the TASONN model is fault tolerance. It is easy to see that damage of a few nodes or links can be automatically repaired in the TASONN model while it is not so in the SONN model. During learning, if any node or link is damaged, the input vectors in the surroundings will generate new nodes and new links. The link strengths (β values) are adapted from the subsequent input presentations.

References

[1] H. Blum, A transformation for extracting new descriptions of shape, IEEE Proceedings of the Symposium on Models for the Speech and Vision Form, Boston 1964, pp. 362–380.

[2] R.W. Smith, Computer processing of line images: a survey, Pattern Recognition 20 (1987) 7–15.

[3] L. Lam, S.W. Lee, C.Y. Suen, Thinning methodologies — a comprehensive survey, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1992) 869–885.

[4] T. Kohonen, Self-Organization and Associative Memory, Springer, Berlin, 1989.
[5] D. Choi, S. Park, Self-creating and organizing neural networks, IEEE Neural Networks 5 (1994) 561–575.
[6] J.A. Kangas, T. Kohonen, J. Laaksonen, Variants of self-organizing maps, IEEE Trans. Neural Networks 1 (1990) 93–99.
[7] T.M. Martinetz, K.J. Schulten, Topology representing networks, Neural Networks 7 (1994) 507–522.
[8] B. Fritzke, A growing neural gas network learns topologies, in: G. Tesauro, D.S. Towetzky, T.K. Leen, (Eds.), Advances in Neural Information Processing Systems, Vol. 7, MIT Press, Cambridge MA, 1995, pp. 625–632.
[9] A. Datta, S.K. Parui, A robust parallel thinning algorithm for binary images, Pattern Recognition 27 (1994) 1181–1192.
[10] B.K. Jang, R.T. Chin, One-pass parallel thinning: analysis, properties and quantitative evaluation, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1992) 1129–1140.
[11] B.K. Jang, R.T. Chin, Analysis of thinning algorithms using mathematical morphology, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1990) 541–551.
[12] C.M. Holt, A. Stewart, M. Clint, R.H. Perrott, An improved parallel thinning algorithm, Commun. ACM 30 (1987) 156–160.
[13] R.W. Hall, Fast parallel thinning algorithms: parallel speed and connectivity preservation, Commun. ACM 32 (1989) 124–131.
[14] R.T. Chin, H.-K. Wan, D.L. Stover, R.D. Iverson, A one-pass thinning algorithm and its parallel implementation, Comput. Vision Graphics Image Process. 40 (1987) 30–40.
[15] T.Y. Zhang, C.Y. Suen, A fast parallel algorithm for thinning digital patterns, Commun. ACM 27 (1984) 236–239.
[16] H.E. Lu, P.S. Wang, A comment on a fast parallel algorithm for thinning digital patterns, Commun. ACM 29 (1986) 239–242.
[17] C. Arcelli, G. Sanniti di Baja, A width-independent fast thinning algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 7 (1985) 463–474.
[18] K.C. Fan, D.F. Chen, M.G. Wen, Skeletonization of binary images with nonuniform width via block decomposition and contour vector matching, Pattern Recognition 31 (1998) 823–838.
[19] C. Arcelli, G. Ramella, Finding grey-skeletons by iterated pixel removal, Image Vision Comput. 13 (1995) 159–167.
[20] K. Mehrotra, C.K. Mohan, S. Ranka, Elements of Artificial Neural Networks, MIT Press, Cambridge, MA, 1997.

About the Author—AMITAVA DATTA received his Master's degree in Statistics from the Indian Statistical Institute in 1977 and did a post-graduate course in Computer Science from the same institute in 1978. After working for a few years in reputed computer industries he joined the Indian Statistical Institute as a System Analyst in 1988. Since then he has been working in image processing and pattern recognition. From 1991 to 1992 he visited GSF, Munich as a Guest Scientist and worked on query-based decision support system. His current interests are in neural net based image processing and computer vision.

About the Author—S.K. PARUI received his Master's degree in Statistics and his Ph.D. degree from the Indian Statistical Institute in 1975 and 1986, respectively. From 1985 to 1987 he held a post-doc position in Leicester Polytechnic, England and during 1988–89 he was a visiting scientist in GSF, Munich. He has been with the Indian Statistical Institute from 1987 and is now an Associate Professor. His current research interests include shape analysis, remote sensing, statistical pattern recognition and neural networks.

About the Author—B.B. CHAUDHURI received his B.Sc. (Hons), B.Tech. and M.Tech. degrees from Calcutta University, India in 1969, 1972 and 1974, respectively, and Ph.D. degree from the Indian Institute of Technology, Kanpur in 1980. He joined the Indian Statistical Institute in 1978 where he served as the Project Co-ordinator and Head of the National Nodal Center for Knowledge Based Computing. Currently, Professor Chaudhuri is the head of the Computer Vision and Pattern Recognition Unit of the institute. His research interests include Pattern Recognition, Image Processing, Computer Vision, Natural Language Processing and Digital Document Processing including OCR. He has published 150 research papers in reputed international journals and has authored two books entitled Two Tone Image Processing and Recognition (Wiley Eastern, 1993) and Object Oriented Programming: Fundamentals and Applications (Prentice-Hall, 1998). He was awarded the Sir J.C. Bose Memorial Award for best engineering science oriented paper in 1986, M.N. Saha Memorial Award (twice) for best application-oriented papers in 1989 and 1991, the Homi Bhabha Fellowship award in 1992 for OCR of Indian Languages and computer communication for the blind, Dr. Vikram Sarabhai Research Award in 1995 for his outstanding achievements in the fields of Electronics, Informatics and Telematics and C. Achuta Menon Prize in 1996 for computer-based Indian language processing. He worked as a Leverhulme visiting fellow at Queen's University, UK in 1981–82, a visiting scientist at GSF, Munich and guest faculty at the Technical University of Hannover during 1986–88. He is a Senior member of IEEE, member secretary (Indian Section) of International Academy of Sciences, Fellow of International Association for Pattern Recognition, Fellow of the National Academy of Sciences (India), and Fellow of the Indian National Academy of Engineering. He is serving as associate editor of the journals Pattern Recognition, Pattern Recognition Letters, and Vivek. He also acted as guest editor of a special issue of the Journal of the IETE on Fuzzy Systems.