On Edge and Line Linking with Connectionist Models

Jayanta Basak, Bhabatosh Chanda, *Member, IEEE*, and Dwijesh Dutta Majumder

***Abstract*— In this paper two connectionist models for mid-level vision problems, namely, edge and line linking, have been presented. The processing elements (PE) are arranged in the form of two-dimensional lattice in both the models. The models take the strengths and the corresponding directions of the fragmented edges (or lines) as the input. The state of each processing element is updated by the activations received from the neighboring processing elements. In one model, each neuron interacts with its eight neighbors, while in the other model, each neuron interacts over a larger neighborhood. After convergence, the output of the neurons represent the linked edge (or line) segments in the image. The first model directly produces the linked line segments, while the second model produces a diffused edge cover. The linked edge segments are found by finding out the spine of the diffused edge cover. The experimental results and the proof of convergence of the network models have also been provided.**

I. Introduction

EDGE AND/OR LINE DETECTION is an inevitable step in many vision-based tasks, and edges (or lines) should be detected as accurately as possible. An edge, in an intensity image, is a boundary between two adjacent extensive regions where graylevel changes abruptly. On the other hand, a line (straight or curved) is found in an intensity image if graylevel along a thin strip is approximately uniform and distinctly different from its neighboring regions.

In real images, it is very difficult to detect edge or line points due to the presence of noise in a scene. The presence of noise creates a randomness in intensity to a great extent. Moreover, the intensity in a natural image changes over a wide range of scale. Marr [1] has shown that there exists some inherent difficulty in detecting edge point location exactly. The inexactness in detecting the edge point location is governed by the uncertainty principle given as

$$\Delta x.\Delta\omega \geq \pi/4. \tag{1}$$

Here, $\Delta\omega$ is the variance in the edge filter's spectrum in the frequency domain, and Δx is the variance in edge location. In other words, if Δx is very small, i.e., edges are accurately located, then $\Delta\omega$ would be large and a number of redundant or noisy edges would be detected. On the other hand, in order to decrease noise, if $\Delta\omega$ is decreased, then there would be smoothing effect and the edges will be dislocated. The uncertainty principle leads to the idea of optimal filtering where an intensity image is first convolved with a Gaussian mask and then the edges are detected as the zero crossings of the Laplacian of the convolved image. The linearity of the Laplacian and the convolution operator give rise to the symmetric filter, namely, Laplacian of Gaussian (LoG). Canny [2] has shown that the detection of the zero crossings in the second directional derivative (instead of using symmetric Laplacian operator) gives better results.

The line-detection algorithms are also involved and there exist the same kind of problems as in the edge-detection techniques. The main idea in line-detection algorithms is to match suitable templates corresponding to the desired type of line segments. In literature [3], quite a few line-detection algorithms are available. In an approach of template matching, as described in [4], nine different 3×3 templates are proposed to measure the likeliness to linear features like edges and lines. Semilinear and nonlinear operators [5], [6] were designed incorporating various constraints. In these approaches, maximum of the responses due to these templates is considered as index of existence of the line points. The responses are later thresholded to separate out the true line points from the non-line ones.

In vision problems, continuous edges and lines are almost always desired. Some portions of the edge or line segments may be lost due to presence of noise, shadowing and different other reasons. Moreover, thresholding the output in line-detection algorithms results in discontinuities and thickening of the lines. Disconnected line or edge segments may be connected to produce continuous lines or edges. Hough transform may be used to link various line or edge segments, where the image space is transformed into parameter space [7]–[9]. However, use of Hough transform is limited by the fact that it does not yield any information about the extent of the line segment being considered. Secondly selection of threshold used in these techniques is also a nontrivial task and completely depends on the nature of the scene.

The problem of separating out the line or edge pixels from the noisy pixels may be solved on the basis of neighborhood information. It can possibly be assumed that a perfect line or edge would be continuous, whereas the noisy pixels do not follow any such continuity. Therefore, based on the information derived from the neighbors the possibility of a point being an edge or line point can be determined (in order to satisfy the continuity constraint). This gives rise to the popular idea of "edge following" or "contour tracking." Ballard and Brown [10] have developed a sequential algorithm for edge following. A variation of this concept has been used in developing the relaxation methods for line and/or edge linking [11], [12].

Manuscript received May 4, 1991; revised April 21, 1993. This work was partly supported by DoE/UNDP under Project No. IND/85/072.

The authors are with the Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta 700 035, India.

IEEE Log Number 9214597.

In a different approach [13], a line segment of predefined length is predicted to exist at some particular location depending on the response to the operator. The prediction is then verified against statistical tests performed on that line. Symbolic processing of closeness and orientation of small line segments are also used to detect long continuous line [14]. There is another appealing approach [15], [16], where a cover over the broken edge segments was formed and broken lines and edges are linked by the edge induction.

The problem of line or edge linking may be solved using *neural networks* or *connectionist architectures* [17], [18]–[21], which are often considered to be suitable for low-level information processing tasks. A number of applications of connectionist models in image processing and early visual tasks have been performed so far. In most of the applications, the problems have been mapped onto any one of the several standard network models like Hopfield model [22], [23], Kohonen model [24], or multilayered perceptron [25]. In many cases the problems have been posed as an optimization problem and solved by using the Hopfield model [26]. Chua and Yang [27] have proposed a cellular neural network model and solved several image processing tasks on it.

In the present investigation two network models are presented to solve the problem of line and edge linking. The first one is structurally similar to the cellular network models [27], although the processing elements, used here, are functionally more versatile than that used in the cellular network models. The paper is organized as follows.

In Section II, the concept and overall methodology for linking the line and/or edge segments is presented. The first network model for line linking is presented in Section III, which includes the line-detection algorithm, the network architecture, and mathematical treatment for the network convergence. Section IV presents the concept of edge diffusion, the architecture of the second network model, and the corresponding analytical treatment of convergence. Section V comes up with the experimental results of line and edge linking for a number of synthetic and real life images. Section VI discusses over the present investigation.

II. Overall Methodology

An edge, as described in Section I, has a step- or ramp-like intensity profile. This suggests the simplest and most popular edge-detection technique: detection of local maxima of the first spatial derivative (gradient) of the intensity values. Similarly, the description of a line indicates that it has a spike- or roof-like intensity profile, i.e., a step-like gradient profile with or without overshoot and undershoot. Consequently, the bright line can be detected as the local maxima of the negative of second directional derivative of the intensity values. In discrete domain, only sixteen different types of line segments (Fig. 1) can appear. For each kind of line segment a particular type of template is designed. When the input image is convolved with the template, the magnitude of the second directional derivative, i.e., the strength of the line, is produced, and the template type indicates the direction of the line. In linking the edge and line segments the information in the neighborhood of each pixel must be processed. The neighboring edge segments, which are aligned, i.e., whose edge directions are very close, should reinforce each other. If two nearby edge or line segments belong to the same continuous curve and there exists some gap (or discontinuity which may be created due to noise) between the segments then the reinforcement process would help in rebuilding the gap or the portion of the curve lost. In most of the classical algorithms, in general, a set of edge (or line) segments which are believed to belong to the same curve are linked to form the continuous line or edge. In these cases finding out the subset of edge (or line) segments which do belong to the same curve is the basic problem. On the other hand, if the process of reinforcement is considered, then edge (or line) segments may be iteratively linked with the neighborhood segments without any prior selection of appropriate subset. In the present approach of edge or line linking two assumptions are made, which are as follows:

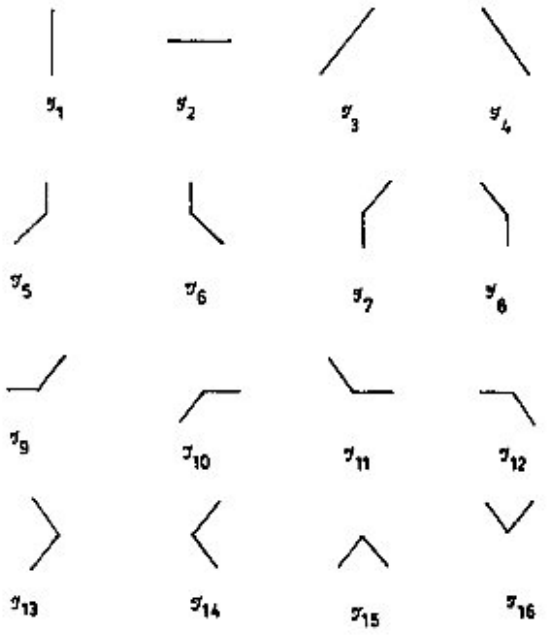

Fig. 1. Digital line segments used in line detection.

1) True edge and line points in a scene follow some continuity pattern, whereas the noisy pixels do not follow any such continuity.
2) The strengths of the true edge or line pixels are greater than those of the noisy pixels.

Although the strength of some noisy pixels may be more than that of true edge or line pixels, the above assumption holds in most of the cases with moderate signal-to-noise ratio. The "strength" of an edge or line will be defined in due course.

In the process of reinforcement, an edge (or line) segment should reinforce the neighborhood segments depending on its direction. Suppose, image contains an edge segment APB, where P is the center pixel (Fig. 5). In that case maximum reinforcement should occur along the extensions of PA and

PB. On the other hand, minimum (or zero) reinforcement should occur in the perpendicular direction of PA and PB. Secondly, the edge segment APB will be reinforced mostly by the segments close to the extensions of PA and PB having similar directions. Thirdly, if two neighboring edge segments reinforce each other, then the edge strengths of the true edge pixels (which are recursively reinforced) will increase, and noisy edge pixels will retain their strengths. In order to stabilize the strength of the true edge (or line) pixels, and to remove noisy edge (or line) pixels, each pixel should inhibit its own edge (or line) strength.

In the implementation of the connectionist models for edge (or line) linking, each model is considered to consist of $m \times n$ neurons (or processing element, or PE) if the image contains $m \times n$ pixels. The neurons are spatially arranged in a two-dimensional lattice in such a way that each neuron corresponds to a separate pixel. Each neuron is connected to surrounding neurons over a neighborhood, and each one has a negative self-feedback. The self-feedback helps in removing the noise in the scene during the process of iterative edge (or line) linking. The edge (or line) information in the image is extracted by standard edge (or line) detection operator. The edge strength and the information for edge (or line) direction at each pixel are fed to the corresponding neuron. This signifies that the input to the connectionist model consists of the edge (or line) information in the image before linking. The output of each neuron represents the edge (or line) strength at that particular location. In other words, the output of the connectionist model represents the edge (or line) map of the image after linking the segments.

Initially, the states of the neurons are clamped by the input (the edge (or line) strengths and the directions) from the graylevel image. The state of the neurons are then updated with the neighborhood information. In the process of updation, the external input (i.e., the input from the image) does not affect the states of the processing elements any more. The edge strengths and the directions extracted from the graylevel image are used only to clamp the state of the neurons at the very initial stage. During updation, the nature of the output of a neuron will depend on the cumulative support received from the neighborhood. Therefore, how the neighborhood neurons are connected will play a key role in the linking process.

III. Eight-Neighborhood Connection

A. Line Detection Operator

As stated before, line strength at any pixel can be computed by matching the templates representing the negative of second directional derivative operator. The strength and corresponding type are considered only when some conditions are satisfied. For example, for the first line segment (τ_1) in Fig. 1, let

$$d_1 = f(x-1,y) + f(x,y) + f(x+1,y) - f(x-1,y-1) - f(x,y-1) - f(x+1,y-1)$$

and

$$d_2 = f(x-1,y) + f(x,y) + f(x+1,y) - f(x-1,y+1) - f(x,y+1) - f(x+1,y+1)$$

where $f(x,y)$ is the grayvalue of the pixel (x,y). In this case, $(d_1+d_2)/2$ is taken as the line strength corresponding to the template type τ_1, if

$$\frac{1}{2}(d_1+d_2) > \epsilon|d_1-d_2|$$

where ϵ is a constant. This condition avoids detecting any overshoot at edge of an extended region as line segment. Line strengths corresponding to all the sixteen templates are computed. Instead of considering only the maximum strength and the corresponding type (of line segment), strengths and types of n-best matches are considered. This is performed with the expectation that true line segment may be a member of the set of n-best matches, and that segment would get support from the neighborhood and would preserve continuity. In the second stage, where line segments are linked (instead of declaring line and non-line pixels in a single step), responses are updated (with the help of a connectionist model) depending on the support they receive from the neighborhood. In the present work, ϵ is chosen as unity.

B. Network Model

Each neuron in this model is connected to its eight neighbors as shown in Fig. 2. The structure has similarity to that of cellular model [27]. Each processing element in this connectionist model has some special functions apart from that in the conventional neural network models. In conventional neural network models [17], [18], [27], [22], [23], each neuron takes a number of inputs in the form of numerical values and produce a single output. In the present model, each neuron (Fig. 3) has a number of independent state elements which are updated by the signals from neighboring neurons depending on the type of the signals. A part of the output of each neuron represents the activation level and the other represents the type of the output. Similarly, the input to a neuron has two parts, one of them is the numerical value, representing the activation level, and the other represents the type of activation. Note that the type of activation (for input and output) actually depends on the template type matched at that location. Mathematically the output of a neuron can be written as

$$\boldsymbol{v}_i = [v_i, T_i^0]$$

where $\boldsymbol{v}_i$ is the output of processing element i, v_i is the neumerical part and T_i^0 is the type. The state vector of an element can be written as

$$\boldsymbol{u}_i = [(u_{i1}, T_{i1}), (u_{i2}, T_{i2}), \cdots, (u_{in}, T_{in})]$$

where $u_{i1}, u_{i2}, \cdots, u_{in}$ are the numerical values of states of processing element i and $T_{i1}, T_{i2}, \cdots, T_{in}$ are the corresponding types. The transfer function maps the state vector to the output. Mathematically,

$$v_i = g\left(\max_{1 \le k \le n}(u_{ik})\right)$$
$$T_i^0 = T_{il} \quad \text{when } u_{il} \ge u_{ik} \forall k, \quad 1 \le k \le n. \tag{2}$$

The transfer function $g(.)$ can be chosen as a ramp function or a sigmoid function.

TABLE I
INTERACTION TABLE (M)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1, 7				7	7	1	1								
2		3, 5							5	3	3	5				
3			2, 6		2		6		6	2			2	6	2	6
4				0, 8		0		8			8	0	8	0	0	8
5	1		6				1, 6	1	6					6		6
6	1			8			1	1, 8			8		8			8
7	7		2		2, 7	7				2			2		2	
8	7			0	7	0, 7						0		0	0	
9		3	2		2					2, 3	3		2		2	
10		5	6				6		5, 6					6		6
11		5		0		0		5			0,		5	0	0	
12		3		8				8			3, 8		8			8
13			6	0		0	6		6			0		0, 6	0	6
14			2	8	2			8		2	8		2, 8		2	8
15			6	8			6	8	6		8		8	6		6, 8
16			2	0	2	0				2		0	2	0	0, 2	

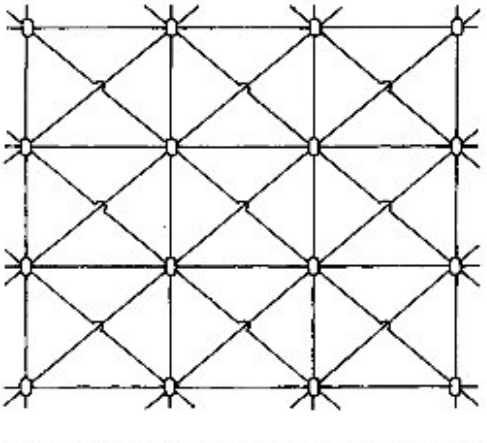

Fig. 2. Connections among the processing elements in a cellular connectionist model.

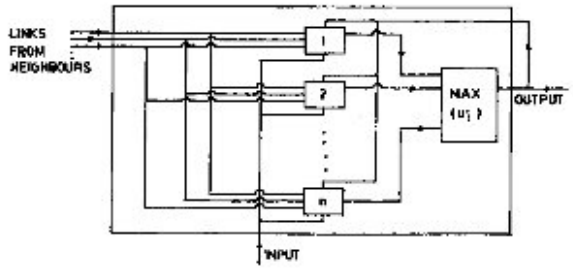

Fig. 3. Schematic diagram of a processing element (PE) for line linking.

The states of each processing element is updated as

$$\frac{du_{ik}}{dt} = \sum_{j \in N(i)} w_{ij} D_{ij}^{k} v_j \tag{3}$$

where w_{ij} is the weight of the link from neuron j to neuron i. It is assumed that all interconnections in the network are symmetric, i.e., $w_{ij} = w_{ji}$. The 8-neighborhood of the neuron i is represented as $N(i)$. D_{ij}^{k} is either 1 or 0 which determines whether or not the neuron i would receive activation from the neuron j or not. Mathematically,

$$D_{ij}^{k} = \mathcal{N}_{ji}(T_j^o, T_{ik}). \tag{4}$$

The neighborhood function ($\mathcal{N}$) is presented in Table I.

In Table I, let $M(m_1, m_2) = k$. If PE i and j are such that their positions are given as (x, y) and $(x + \lfloor k/3 \rfloor - 1, y + \lfloor k/3 \rfloor - 1)$ respectively, then the neighborhood function $\mathcal{N}_{ji}(\tau_{m1}, \tau_{m2}) = 1$ and zero otherwise. A blank entry in the table indicates $\mathcal{N}_{ji} = 0$ for all i, j.

The neighborhood function can also be described in the following way. There can be sixteen different types which can be presented as an ordered pair (η_1, η_2). Let the types at ith and jth neurons be defined as

$$T_j^o = (\eta_{j1}^o, \eta_{j2}^o)$$
$$T_{ik} = (\tau_{ik1}, \tau_{ik2})$$

The form is defined by considering each line segment in Fig. 1 as a collection of two segments joined at the center pixel.

The values in Table II shows the positions and values of η_1 and η_2 except the center point. For example, the template type τ_1 can be represented as $\tau_1 = (-3, 3)$ or $(3, -3)$. Similarly, τ_2 can be represented as $\tau_2 = (-1, 1)$ or $(1, -1)$. It is to be noted that any template (τ_{k1}, τ_{k2}) is same as (τ_{k2}, τ_{k1}). Let the positions of the processing elements i and j be (x_i, y_i) and (x_j, y_j) respectively in the lattice structure of the network. Let a variable η be defined as

$$\eta = (x_j - x_i) + 3(y_i - y_i).$$

In that case, if j is in the 8-neighborhood of i then η will take the same set of values as shown in Table II. The output type of PE j will affect the type of kth state element of PE i when either of the following conditions hold:

$$a)\ \eta = \eta_{ik1} \wedge (\eta_{j1}^o = -\eta_{ik1} \vee \eta_{j2}^o = -\eta_{ik1})$$
$$b)\ \eta = \eta_{ik2} \wedge (\eta_{j1}^o = -\eta_{ik2} \vee \eta_{j2}^o = -\eta_{ik2})$$

For example, let the kth state element of PE i be τ_5 or $(2, -3)$. In that case, according to condition a), PE j should be placed in such a way that $\eta = 2$; i.e., PE j should be placed at the

TABLE II
RELATIVE POSITIONS OF THE PROCESSING ELEMENTS IN THE NETWORK

	x_{j-1}	x_j	x_{j+1}
u_{i-1}	−1	−3	−2
y_i	−2	0	1
y_{i+1}	2	3	4

lower left corner of PE i, and T_j^o should be $(-2, *)$. Similarly, according to condition b), PE should be placed in such a way that $\eta = -3$; i.e., PE j should be just above PE i, and T_j^o should be $(-3, *)$. If either condition a) or condition b) is satisfied, then only PE j will affect the kth state element of PE i. Mathematically, it can be written as

$$\mathcal{N}_{ji}(T_j^o, T_{ik}) = \max\{1 - (|\eta - \eta_{ik1}| + |(\eta_{j1}^o + \eta_{ik1})(\eta_{j2}^o + \eta_{ik1})|) \cdot (|\eta - \eta_{ik2}| + |(\eta_{j1}^o + \eta_{ik2})(\eta_{j2}^o + \eta_{ik2})|), 0\}. \quad (5)$$

The updation rule described in (3) has been derived on the basis of continuity of the line (or edge) points. The image, after computing the response to line (or edge) detection operator at each point, is mapped onto the array of processors such that the output of each neuron represents the strength of the line (or edge) at that point. Ideally, after convergence of the network, the output of a neuron corresponding to a line (or edge) point should be unity and zero otherwise. On a continuous line there should be two line (or edge) points on the opposite sides of a point (except for the end points). Therefore, the neurons corresponding to a line (or edge) point will receive activations from two neighboring neurons. On the other hand, the noisy pixels in an image do not follow any continuous pattern. As a result, the neurons corresponding to the noisy pixels do not receive proper support from the neighbors, and consequently their activation levels remain unchanged. Therefore, the resultant effect is an enhancement of the line (or edge) pixels without any reduction of noise. To reduce the effect of noise, each neuron has a negative self-feedback. Usually, for a pixel on a continuous line, the activations received by the corresponding neuron from the neighborhood is greater than the negative feedback. The updation rule with negative feedback can be written as

$$\frac{du_{ik}}{dt} = \sum_{j \in N(i)} w_{ij} D_{ij}^k v_j \quad - w_s v_i \quad (6)$$

where w_s is the strength of the self-feedback which is same for all neurons in the network.

C. Convergence of the Network

To prove the convergence and stability of the network, it is sufficient to show that both the output type and the output value converges and remain stable. Firstly, let us prove the stability of the output types.

Proof: Suppose, for any two neighboring neurons i and j, if $T_i^o = T_{ik}$ and $T_j^o = T_{jl}$ then

$$D_{ij}^k = 1 \Leftrightarrow D_{ji}^l = 1.$$

This indicates that if the neuron i supports the output type of neuron j, then neuron j must support the output type of neuron i (this is also evident from the structure of the templates).

For sake of simplicity, let us consider $n = 2$. Therefore, for PE i, the state vector can be written as $[(u_{i1}, T_{i1}), (u_{i2}, T_{i2})]$. Let the output be $[v_i, T_i^o]$. Suppose initially the activation of the neuron is such that $T_i^o = T_{i1}$; i.e., $u_{i1} > u_{i2}$ and $v_i = g(u_{i1})$. In the process of updation, T_i^o will change to T_{i2} only if u_{i2} becomes greater than u_{i1}. This will happen only when the state element (u_{i2}, T_{i2}) gets more support than the state element (u_{i1}, T_{i1}) from the neighborhood processing elements. If $T_i^o = T_{i1}$, the processing elements supporting the state element (u_{i1}, T_{i1}), would receive support from the processing element i. On the other hand, the neurons that support the state element (u_{i2}, T_{i2}), would not receive support from the neuron i. In such a situation if T_i^o changes to T_{i2}, then processing elements of the second category would receive even more support, and therefore, the support to (u_{i2}, T_{i2}) would increase further. This ensures that u_{i2} increases recursively over u_{i1} and T_i^o remains the same as T_{i2}. Therefore there can be at most one interchange between the first and the second state elements. For $n > 2$, the same condition holds true and there can be at most $n - 1$ changes in the output types. Finally, output types of all processing elements will stabilize depending on the neighborhood information. □

It can be shown that under the updation rule given by (6), the output value of the network stabilizes.

Proof: Let us consider the energy function of the network to be

$$E(t) = -\frac{1}{2} \sum_i \sum_{j \in N(i)} w_{ij} D_{ij} v_i v_j + \frac{1}{2} \sum_i w_s v_i^2 \quad (7)$$

where the term D_{ij} is given as

$$D_{ij} = \begin{cases} 1 & \text{if } D_{ij}^k = 1 \text{ and } T_i^o = T_{ik} \\ 0 & \text{otherwise.} \end{cases}$$

From (6) it is clear that the updation of the output activation can be written as

$$\frac{du_i}{dt} = \sum_{j \in N(i)} w_{ij} D_{ij} v_j \quad - w_s v_i \quad (8)$$

where u_i represents the state value of the mostly activated state element of processing element i; i.e., $u_i = g^{-1}(v_i)$. Equation (8) is true because if some neuron j does not update the mostly activated state element of neuron i, then the change in output of neuron i due to j will be zero.

From (7), it is clear that the energy function is bounded for some finite network structure. Moreover between two neighboring neurons,

$$D_{ij}^k = 1 \wedge T_i^o = T_{ik} \Leftrightarrow D_{ji}^l = 1, \text{where } l \text{ is such that } T_{jl} = T_j^o.$$

In other words,

$$D_{ij} = D_{ji}. \quad (9)$$

Moreover, in the design of the network, w_{ij} was selected to be symmetric, i.e., $w_{ij} = w_{ji}$. Therefore, from (7),

$$\frac{dE}{dt} = -\sum_i \left(\sum_{j \in N(i)} w_{ij} D_{ij} v_j - w_s v_i \right) \left(\frac{dv_i}{dt} \right). \quad (10)$$

Substituting the value dv_i/dt from (8), (10) becomes

$$\frac{dE}{dt} = -\sum_i g^{-1'}(v_i)\left(\frac{dv_i}{dt}\right)^2 \tag{11}$$

where $g^{-1'}$ is the first derivative of the inverse of $g(.)$, which is an increasing function. Therefore $g^{-1'}$ is positive, and evidently, $(dv_i/dt)^2$ is always nonnegative. Therefore,

$$\frac{dE}{dt} \le 0, \quad \forall t > 0.$$

Since $E(t)$ is bounded, $dE/dt \to 0$ as $t \to 0$ and therefore $dv_i/dt \to 0$ as $t \to 0$ for all i. As a result, the output values of all processing elements converge. □

D. Modified Updation

In the network, the output of a state element of a neuron changes if it gets support from even one of its neighboring PEs according to (6). In a continuous line, a line point (except the end points) would get support from two of its neighbors, depending on the direction of the line at that point. Again, for noisy pixels the chance of getting support from both the neighbors simultaneously is much less than that from one neighbor. Therefore, to reduce the noise level in the output, the state vector of each processing element is updated if it gets support from both the neighbors simultaneously. On the other hand, this process degrades the continuity of the line to some extent.

The modified rule of updation of the state vector of each PE can be written as

$$\frac{du_{ip}}{dt} = w\sum_j \sum_{k\in N(i)} D^p_{ijk}\sqrt{v_j v_k} - w_s v_i \tag{12}$$

where the weights of interconnections in the network are considered to be uniform and equal to w. $D^p_{ijk} = D^p_{ij} \wedge D^p_{ik}$; i.e., the state element (u_{ip}, T_{ip}) is updated only if it gets support from both the PEs j and k in its neighborhood. In the process of updation, the output type is the type of the mostly activated state element. It can be shown as before (Section III-B) that the output type of any processing element in the network always converges; i.e., type of each PE can change at most $n-1$ times, where n is the number of state elements. The output value of each PE in the network can also be shown to converge as follows.

Let there exists some energy function $E(t)$ such that

$$\frac{dE}{dt} = -\sum_i \left(\frac{dv_i}{dt}\right)^2 g^{-1'}(v_i) \tag{13}$$

i.e.,

$$E(t) = -\int_0^t \frac{dE}{d\tau}d\tau \tag{14}$$

where τ is a dummy variable. If it can be shown that $E(t)$ is bounded for any value of t then the network can be claimed to converge. This is true because (13) makes it clear that

$$\frac{dE}{dt} \le 0, \quad \forall t.$$

Therefore, $dE/dt \to 0$ as $t \to \infty$ if and only if $E(t)$ is bounded.

Proof: It is evident that

$$\sqrt{v_j v_k} \le \frac{1}{2}(v_j + v_k).$$

Again,

$$D^p_{ijk} = D^p_{ij} \wedge D^p_{ik}.$$

Therefore,

$$D^p_{ijk}\sqrt{v_j v_k} \le \frac{1}{2}(D^p_{ij}v_j + D^p_{ik}v_k). \tag{15}$$

The updation rule (12) can be written as

$$\frac{du_{ip}}{dt} \le w\sum_{j\in N(i)} D^p_{ij}v_j - w_s v_i. \tag{16}$$

Let us assume

$$\left(\frac{du_{ip}}{dt}\right)_1 = w\sum_{j\in N(i)} D^p_{ij}v_j - w_s v_i. \tag{17}$$

In other words,

$$\frac{du_{ip}}{dt} \le \left(\frac{du_{ip}}{dt}\right)_1.$$

As a result, if (u_{ip}, T_{ip}) is the state element such that $v_i = g(u_{ip})$ then it can be said that

$$\frac{dv_i}{dt} \le \left(\frac{dv_i}{dt}\right)_1, \quad \forall i. \tag{18}$$

Again, for all $i, 0 \le v_i \le 1$. Therefore we have

$$v_j v_k \le \sqrt{v_j v_k}.$$

Let

$$\left(\frac{du_{ip}}{dt}\right)_2 = w\sum_j \sum_{k\in N(i)} D^p_{ijk}v_j v_k - w_s v_i. \tag{19}$$

In that case,

$$\left(\frac{du_{ip}}{dt}\right)_2 \le \left(\frac{du_{ip}}{dt}\right).$$

As a consequence we have

$$\left(\frac{dv_i}{dt}\right)_2 \le \left(\frac{dv_i}{dt}\right), \quad \forall t. \tag{20}$$

Therefore, from (18) and (20),

$$\left(\frac{dv_i}{dt}\right)_2 \le \left(\frac{dv_i}{dt}\right) \le \left(\frac{dv_i}{dt}\right)_1. \tag{21}$$

Let us consider two energy functions, E_1 and E_2 given as

$$E_1 = -\frac{1}{2}w\sum_i \sum_{j\in N(i)} D_{ij}v_i v_j + \frac{1}{2}\sum_i w_s v_i^2 \tag{22}$$

and

$$E_2 = -\frac{1}{2}w\sum_i \sum_j \sum_{k\in N(i)} D_{ijk}v_i v_j v_k + \frac{1}{2}\sum_i w_s v_i^2. \tag{23}$$

In these equations D_{ij} has the same meaning as before. $D_{ijk} = 1$, if $D^p_{ijk} = 1, D^q_{jik} = 1$ and $D^r_{kij} = 1$ where $v_i = g(u_{ip}), v_j = g(u_{jq})$ and $v_k = g(u_{kr})$. From (22) and (23), it is clear that

$$\frac{dE_1}{dt} = -\sum_i g^{-1'}(v_i)\left(\frac{dv_i}{dt}\right)^2_1 \tag{24}$$

and

$$\frac{dE_2}{dt} = -\sum_i g^{-1'}(v_i)\left(\frac{dv_i}{dt}\right)^2_2. \tag{25}$$

Therefore, from (18), (20), (24), and (25) it is evident that at any state of the network,

$$\frac{dE_1}{dt} \le \frac{dE}{dt} \le \frac{dE_2}{dt}. \tag{26}$$

Again, the energy functions E_1 and E_2 are bounded (from (22) and (23)). Therefore, the energy function $E(t)$ is also bounded [28]. □

IV. Larger Neighborhood

In the present section, neighborhood size is considered to be larger than the 8-neighborhood considered so far. If a larger neighborhood is used, it is difficult to determine which neighboring pixels should reinforce each other depending only on the template types. Therefore, instead of using the template types, the edge (or line) directions are stored here explicitly. The input to the network consists of two parts, in this case also, to represent both the edge strength and the direction.

A. Strategy for Edge Detection

In the present approach, every point has an edge vector, whose direction is the direction of the intensity change (from lower to higher intensity), and the magnitude is the absolute value of the first derivative. Note that the direction is normal to the edge (or line). The concept is that each edge vector induces edge points over a neighborhood. In the present model the neighborhood is selected as circular. The induction of edge points is maximum along the edge (or line), i.e., normal to the edge direction. The process of induction by an edge point is discussed below.

Consider any edge point P (Fig. 4) with the edge vector $\mathbf{e} = (e, \alpha)$, where e is the magnitude and α is the direction of the vector. N_P is the circular neighborhood of P (of radius R_P) over which P induces edge points. In a rectangular coordinate system, the edge vector can be written as (e_x, e_y), where e_x and e_y are the components along x- and y-axes respectively; i.e., $e_x = e\cos\alpha$ and $e_y = e\sin\alpha$. From the continuity principle (assumption i in Section II), edge points are expected to appear along a line AB which is perpendicular to the edge direction. Therefore, P should try to induce edge vectors at every point on AB. If it is assumed that in a real scene the edges are separated by an amount greater than R_P then there should be no edge point along the direction of edge vector. In other words, P should not affect any point on CD. If the effect of P is considered to be smoothly distributed over its neighborhood then it can be stated that the effect of P on

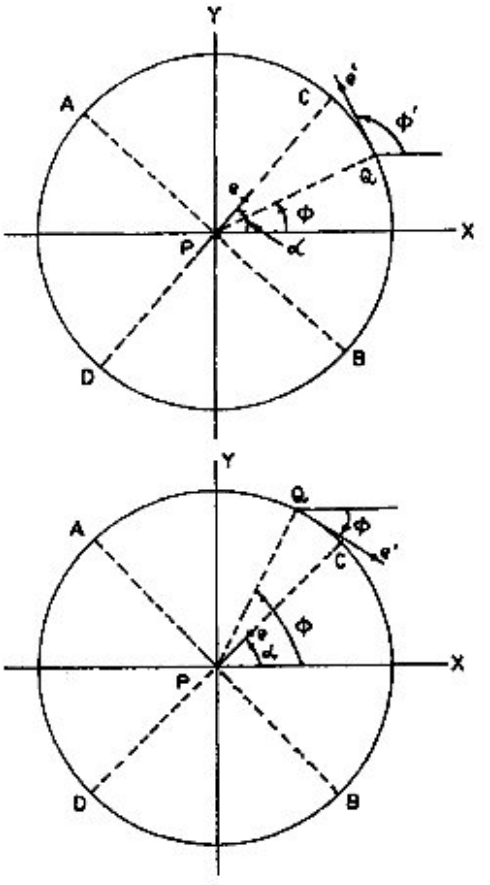

Fig. 4. Edge induction on a circular neighborhood; AB is perpendicular to CD, e is the actual edge strength, e' is the induced edge vector.

the points on PQ increases as Q approaches B and decreases as it approaches C. If P induces an edge at any point on PQ, then it is reasonable to assume that the direction of the induced edge vector is perpendicular to PQ. Let the effect of P on any point P' in N_P be $e' = (e', \phi')$, where e' and ϕ' are the magnitude and direction of the induced vector. Then

$$\phi' = \begin{cases} \pi/2 = \phi & \text{if } \alpha > \phi \\ \phi - \pi/2 & \text{otherwise} \end{cases} \tag{27}$$

where ϕ is the angle subtended by the line PP' with x-axis. The induced edge strength is given as

$$e' = e\cos\theta, \text{ where } \theta = |\alpha - \phi'|.$$

The induced edge vector $\mathbf{e}'$ can also be written as (e'_x, e'_y) in 2-D rectangular coordinate system. It can be proved (Appendix I) that

$$\begin{aligned} e'_x &= e_x\sin^2\phi - e_y\sin\phi\cos\phi \\ e'_y &= e_y\cos^2\phi - e_x\sin\phi\cos\phi. \end{aligned} \tag{28}$$

In the process of edge-point induction, spreading of edges will occur due to presence of any edge point. If there exists some broken edge (or line) segments, and the length of the

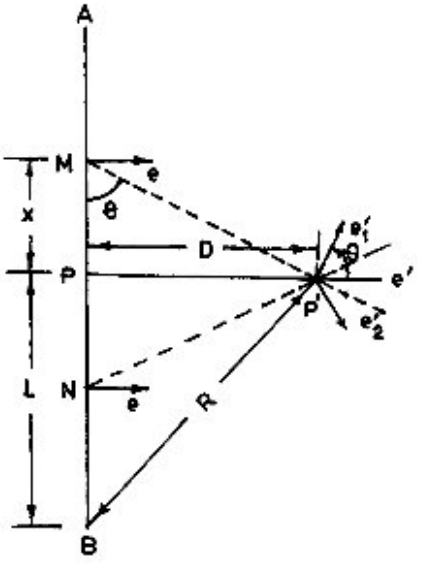

Fig. 5. Edge induction at a point P' due to edge segment AB. PP' is normal to AB. The edge strength at every point on AB is considered to be equal to e.

missing segment is less than or equal to the diameter of the region of induction, then the broken segments are linked by induced edges. The noisy edges also spread their activity and create regions of diffused edges. In order to decrease the effect of noise, each edge point should inhibit its own strength, just like the feedback in the previous algorithm (Section III). In the process of reinforcement of edge strengths by edge-point induction, a ridge of edge responses forms along a true edge (or line), where maximum response occurs at the true edge points.

Since each edge point on an edge segment spreads its activity over a neighborhood, the normal distance of the induced edge point is less than or equal to the radius of the neighborhood of activity. Let P be a point near an edge segment AB, such that $PA = PB = L$, where L is the maximum distance from which a point on AB can induce P (Fig. 5). It can be shown (Appendix II) that the induced edge strength e' at point P is

$$e' = 2eL\left[1 - \frac{\tan^{-1}(L/D)}{(L/D)}\right]. \tag{29}$$

In the above expression it is assumed that all points on AB have their edge vectors in the same direction and have same magnitudes equal to e. Since $R^2 = L^2 + D^2$, the expression for e' can be written as

$$e' = 2e\sqrt{R^2 - D^2}\left[1 - \frac{\tan^{-1}\left(\sqrt{(R/D)^2 - 1}\right)}{\sqrt{(R/D)^2 - 1}}\right]. \tag{30}$$

It is clear from the above expression that $e' = 0$ when $D = R$, and is maximum ($e' = 2eR$) when $D = 0$. The value of e' at a point decreases as the point moves farther from the true edge segment. Therefore, the maximum values occur at the true edge points with a gradual decrease in edge strengths on either sides of the edge. If lateral nonmaximum suppression is performed, then the actual edge points can be found out. The algorithm is implementable on a network model which is discussed below.

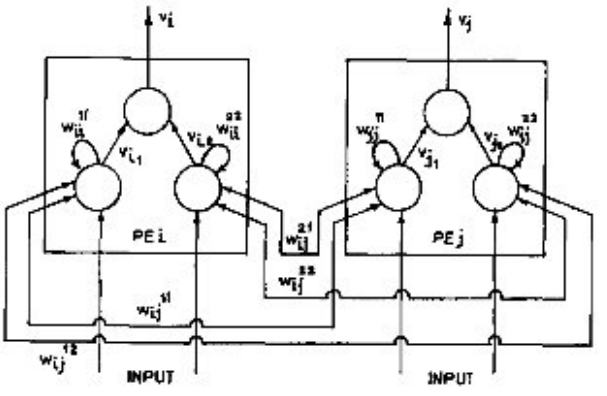

Fig. 6. Interconnections between two processing elements i and j in the network. Each processing element has three active elements.

B. Network Model

The network model is composed of processing elements arranged in a two-dimensional lattice similar to the model used in the previous method (Section III). Here each PE is connected over a large neighborhood unlike the eight-neighborhood in the previous model. The model is designed to have a massively parallel implementation of the algorithm described before. The interconnection between any two processing elements is shown in Fig. 6. Each neuron has two different intermediate outputs. Outputs v_{i1} and v_{i2} represent the edge responses along the x and y directions (i.e., e_x and e_y) respectively at the location corresponding to processing element i. The weights of the interconnections depend on the spatial distribution of the processing elements which is governed by (31). For any two PEs i and j, if $i \in N_j$, then the weights of the links from j to i can be mathematically written as

$$\begin{aligned} w_{ij}^{11} &= w \sin^2 \phi_{ij} \\ w_{ij}^{22} &= w \cos^2 \phi_{ij} \\ w_{ij}^{12} &= -w \sin \phi_{ij} \cos \phi_{ij} \\ w_{ij}^{21} &= -w \sin \phi_{ij} \cos \phi_{ij}. \end{aligned} \tag{31}$$

Here, ϕ_{ij} is the angle subtended with the x-axis by the imaginary line joining PE j to PE i (in the lattice structure). w_{ij} represents the weight of the link from PE j to PE i. The distribution of weights is shown in Fig. 7. Each processing element i has negative self-feedback w_{ii}^{11} and w_{ij}^{22} as shown in Fig. 6.

At any instant of time t, say, let $R(t)$ be the radius of the circular neighborhood $N_j(t)$. It is evident that if i is in the neighborhood of j (i.e., the distance between i and j is less than the radius of neighborhood of activity) then j is in the neighborhood of i (since the neighborhood is circular). In other words, the weights connecting the processing elements

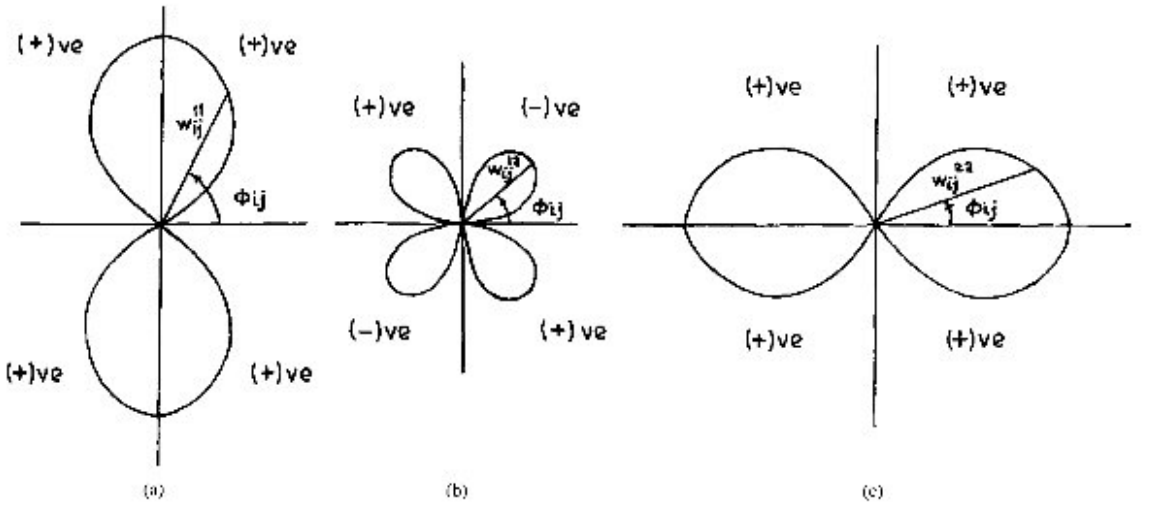

Fig. 7. Distribution of weights of the interconnection links in polar coordinates. The radius vector denotes the absolute value of the weight. (a) Distribution of w_{ij}^{11}. (b) Distribution of w_{ij}^{12}. (c) Distribution of w_{ij}^{22}.

are symmetric. For all processing elements the amount of self-feedback are same. In the present architecture, the amount of self-feedback is proportional to the radius of the neighborhood of activity, i.e.,

$$w_{ii}^{11} = -w_s R$$
$$w_{ii}^{22} = -w_s R \tag{32}$$

where w_s is a constant and R is the radius of the neighborhood at any instant of time.

The output of any processing element i is given as

$$v_i = \sqrt{v_{i1}^2 + v_{i2}^2} \tag{33}$$

where v_{i1} and v_{i2} are intermediate outputs of the processing element i which are given as

$$v_{i1} = g(u_{i1})$$

and

$$v_{i2} = g(u_{i2}).$$

u_{i1} and u_{i2} are the instantaneous inputs to the processing element i as shown in Fig. 6. The transfer function $g(.)$ is a semilinear nondecreasing function. It can be chosen as a sigmoid function. In the present case the functional form of $g(.)$ is

$$g(x) = \begin{cases} 0 & \text{if } x < -1 \\ x & \text{if } -1 \le x < 1 \\ 1 & \text{otherwise.} \end{cases} \tag{34}$$

The dynamics of the network can be written as

$$\frac{du_{i1}}{dt} = \sum_{j,j\in N_i} w_{ij}^{11} v_{j1} + \sum_{j,j\in N_i} w_{ij}^{12} v_{j2} - u_{i1} + w_{ii}^{11} v_{i1} \tag{35}$$

and

$$\frac{du_{i2}}{dt} = \sum_{j,j\in N_i} w_{ij}^{21} v_{j1} + \sum_{j,j\in N_i} w_{ij}^{22} v_{j2} - u_{i2} + w_{ii}^{22} v_{i2}. \tag{36}$$

The size of the neighborhood of each processing element decreases with time. As the radius of neighborhood reduces to zero, the process of updation of the states of processing elements stops. Mathematically, for any time instances t_1, t_2, t_3, such that $t_1 < t_2 < t_3$, the radii of neighborhoods follow

$$R(t_1) > R(t_2) > R(t_3)$$

and

$$\lim_{t\to\infty} R(t) = 0.$$

Since the self-feedback is proportional to the radius of the neighborhood, the self-feedback also reduces to zero as the radius decreases to zero, i.e., the neighborhood region shrinks to a point. When the radius becomes zero, there exists no activation from the neighborhood, and the self-feedback also reduces to zero. If the network converges for a neighborhood of fixed radius, then evidently the network must converge for shrinking neighborhood. The proof of convegence for fixed radius of neighborhood is as follows.

Proof: Let E_1 and E_2 be two energy functions given as

$$E_1 = -\frac{1}{2}\sum_i\sum_j w_{ij}^{11} v_{i1} v_{j1} - \frac{1}{2}\sum_i\sum_j w_{ij}^{12} v_{i1} v_{j2} + \int_0^{v_{i1}} g^{-1}(\xi)d\xi - \frac{1}{2}\sum_i w_{ii}^{11} v_{i1}^2 \tag{37}$$

$$E_2 = -\frac{1}{2}\sum_i\sum_j w_{ij}^{21} v_{i2} v_{j1} - \frac{1}{2}\sum_i\sum_j w_{ij}^{22} v_{i2} v_{j2} + \int_0^{v_{i2}} g^{-1}(\xi)d\xi - \frac{1}{2}\sum_i w_{ii}^{11} v_{i1}^2. \tag{38}$$

The energy function of the system is given as

$$E(t) = E_1(t) + E_2(t). \tag{39}$$

Therefore,

$$\frac{dE}{dt} = \frac{dE_1}{dt} + \frac{dE_2}{dt}. \tag{40}$$

Using the first derivatives of E_1 and E_2 (in (37) and (38)), the expression for dE/dt can be written as

$$\begin{aligned}\frac{dE}{dt} = &-\sum_i\sum_j w_{ij}^{11}v_{j1}\left(\frac{dv_{i1}}{dt}\right) - \sum_i\sum_j w_{ij}^{22}v_{j2}\left(\frac{dv_{i2}}{dt}\right)\\ &+\sum_i u_{i1}\left(\frac{dv_{i1}}{dt}\right)\\ &-\sum_i w_{ii}^{11}v_{i1}\left(\frac{dv_{i1}}{dt}\right) + \sum_i u_{i2}\left(\frac{dv_{i2}}{dt}\right)\\ &-\sum_i w_{ii}^{22}v_{i2}\left(\frac{dv_{i2}}{dt}\right)\\ &-\frac{1}{2}\sum_i\sum_j\left[w_{ij}^{12}v_{j2}\left(\frac{dv_{i1}}{dt}\right) + w_{ij}^{21}v_{i2}\left(\frac{dv_{j1}}{dt}\right)\right.\\ &\left.-\frac{1}{2}\sum_i\sum_j\left[w_{ij}^{21}v_{j2}\left(\frac{dv_{i2}}{dt}\right) + w_{ij}^{12}v_{i1}\left(\frac{dv_{j2}}{dt}\right)\right]. \quad (41)\end{aligned}$$

Since the weights are symmetric, (31) can be written as

$$\begin{aligned}\frac{dE}{dt} = &-\sum_i\left[\sum_j w_{ij}^{11}v_{j1} + \sum_j w_{ij}^{12}v_{j2} - u_{i1} + w_{ii}^{11}v_{i1}\right]\left(\frac{dv_{i1}}{dt}\right)\\ &-\sum_i\left[\sum_j w_{ij}^{21}v_{j1} + \sum_j w_{ij}^{22}v_{j2} - u_{i2} + w_{ii}^{22}v_{i2}\right]\left(\frac{dv_{i2}}{dt}\right). \quad (42)\end{aligned}$$

From (35) and (36),

$$\frac{dE}{dt} = -\sum_i g^{-1'}(v_{i1})\left(\frac{dv_{i1}}{dt}\right)^2 - \sum_i g^{-1'}(v_{i2})\left(\frac{dv_{i2}}{dt}\right)^2. \quad (43)$$

Since $g(.)$ is a semilinear increasing function, $g^{-1'}(.)$ is positive. Therefore, it is evident that $dE/dt \leq 0$ for all $t > 0$. Since $E(t)$ is bounded (from eq. (39))

$$\frac{dE}{dt} \to 0 \qquad \text{as } t \to \infty.$$

Consequently, dv_{i1}/dt and dv_{i2}/dt approaches zero as t goes to infinity, and the system converges. □

As the radius of the neighborhood of activity of each processing element decreases, the energy of the system changes. At any point of time the change of energy is finite, and as a result, the rate of energy change reduces to zero as the neighborhood shrinks to a point. The output of the network depends on the rate at which the radius of the neighborhood decreases. If the radius decreases too fast, then the true edge points may not get sufficient time to be linked properly. On the other hand, if radius is decreased too slowly, then flat bars of edge responses along the true edge points is formed. Let the radius of neighborhood of any processing element is decreased linearly at a rate γ; i.e., $dR/dt = -\gamma$. In that case, to prevent the saturation of the edge strength near a true edge point, it is required that (from Appendix III)

$$\gamma > cwR^2(0).$$

In this expression, c is the edge strength at any point on a true edge, and $R(0)$ is the initial radius of the neighborhood. For any image, γ is considered as $\lambda wR^2(0)$, where λ is a constant. Considering the inhibitory effect of self-feedback and excitatory effect from the induced edge points, the lower and upper bounds for γ can be fixed. In real images considering average edge responses, the value of λ is selected as 0.6.

V. Simulation and Results

The lines and edges have been detected from graylevel images, and consequently linked by the proposed network models. In the first set of experiments, the results are obtained with 8-neighborhood using the model of the first kind. In another set of experiments, edge induction method is applied and the results are obtained with the network of the second kind. In both cases the simulation consists of two parts. In the first stage, the line or edge points' strengths and directions are computed at every pixel of the graylevel images. This is performed either by matching sixteen templates as mentioned before or using some suitable edge operator. In the second stage of experiment, the line or edge information are fed to the network models, and the states of the neurons are updated to get the linked line or edge segments at the output.

A. Results from Model 1

The possible line pixels are found by matching all the templates corresponding to sixteen different line segments described in Section III-A. At each pixel only the maximum and the second maximum responses are considered for subsequent processing. These two response values and the corresponding types (of the templates) for each pixel are fed to the corresponding neuron. Since only two response values are fed, it is assumed that the state vector of each processing element consists of two state elements, i.e., $n = 2$. It is observed that the third maximum responses for most of the pixels in an image are much less than the first and second ones. Therefore, if the number of state elements are considered to be three or more, then there would be negligible change in the output of the network.

The network was simulated for different interconnection weights and different amounts of self-feedback. It is observed that the time of convergence of the network would increase if the weights are chosen to be very small. On the other hand, there would be unwanted oscillation in the output if the weights are too large. The weights are fixed experimentally. The amount of self-feedback is selected to be less than the strength of the interconnections in order to ensure that the output of the neurons corresponding to genuine line (or edge) pixels do not reduce to zero. In the present investigation, the weights of interconnection are selected to be 100 and the amount of self-feedback is chosen as 50. The ratio of self-feedback and the interconnection weight may be typically chosen as 0.5, although it depends on the particular image.

In the first type of updation (6) it is found that the effect of noise is greater than that in the second type of updation (12). On the other hand, in second type of updation, discontinuities at some points are not resolved. Therefore, to have both the effects of linking and noise reduction, two methods are interleaved. It is to be noted that different ratios of intermixing

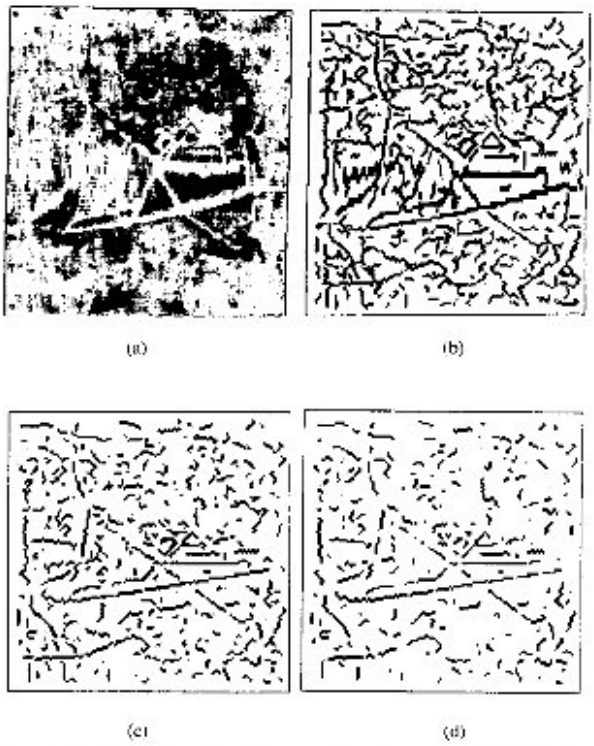

Fig. 8. (a) Satellite image showing a part of an airport (128 × 128). (b) Lines detected using the interleaved ratio $s : p = 1 : 5$. (c) Lines detected using the interleaved ratio $s : p = 1 : 20$. Here, s is the number of iterations using the updation rule in (6) and p is the number of iterations using the updation rule in (12). Parameters are selected as $w = 100$, $w_e = 50$, and time step = 0.005.

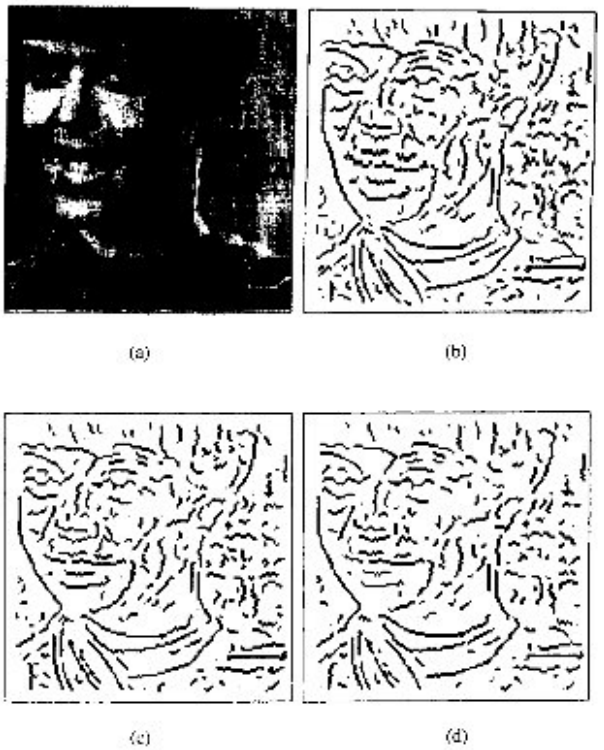

Fig. 9. (a) An image of a girl (125 × 125). Lines are detected from the gradient image using interleaved ratio (b) $s : p = 1 : 10$, (c) $s : p = 1 : 15$, (d) $s : p = 1 : 20$. Here, s is the number of iterations using the updation rule in (6), and p is the number of iterations using the updation rule in (12). Parameters are selected as $w = 100$, $w_e = 50$, and time step = 0.005.

the updation rules results in outputs of different quality. The results have been presented in Fig. 8 and Fig. 9.

In Fig. 8(a), a portion of an airport in a satellite image is presented. The bright lines are detected in the image using the line detecting templates as mentioned before. The lines have been linked with different interleaved ratio, and the results are presented in Fig. 8(b) (d). The runway in satellite image has been detected.

Image of a girl is presented in Fig. 9(a). The gradient response of the image was found using Sobel operator. The edges are treated as lines in the gradient image and are detected using the line detecting templates. The lines were then linked and the result is presented in Fig. 9(b) (d).

It is clear from the results that for a ratio of 1:5 (Fig. 8(b)) of the updation rules (6) and (12) the linking of the line segments is better, but at the same time quite a few redundant line segments appear. For a ratio of 1:20 (Fig. 8(d)), the noise decreases to a great extent while the linking is not as good as that in Fig. 8(b). The results for a ratio of 1:10 (Fig. 8(c)) presents an output of intermediate quality where a tradeoff between the quality of linking and the noise reduction is achieved. Similar effects of intermixing are observed with the lines detected from the gradient image of the girl.

B. Results from Model 2

In the second set of experiments, the edge responses have been found by the following method. The edge responses r_x and r_y at every pixel are found by using the first partial derivatives along the x- and y-axes. The resultant response r and the edge direction α at a pixel are found by using the relation

$$r = \frac{r_x^2 + r_y^2}{|r_x| + |r_y|}$$

and

$$\alpha = \tan^{-1}\left(\frac{r_y}{r_x}\right).$$

The resultant response at every point is divided by $|r_x|+|r_y|$ to make it insensitive to the edge orientation. The edge responses at all pixels are then normalized to (0, 1) by maximum value of r. After normalization edge vectors are denoted by $\mathbf{e}$. The components of the normalized edge vector along x- and y-axes (e_x and e_y respectively) are fed to the network as input. As described before, the states of the processing elements are initially clamped to normalized edge components.

In the network each neuron has a circular neighborhood of activity. For any point i, a set of points S_i is taken as the circular neighborhood of i in discrete domain, equivalent to the analog neighborhood, such that for any $j \in S_i$,

$$d(i, j) \leq R + 0.5$$

where $d(i,j)$ is the Euclidian distance between i and j, and R is the radius of the equivalent circular neighborhood in analog domain.

The weights of the interconnection between the processing elements are set according to (31). The value of w determines the speed of convergence of the network. A low value of w would cause a slow convergence, whereas a very high value of w would result in rather unwanted oscillations. The value of w was selected experimentally and is chosen as 100. In fact, the value of w depends, to some extent, on the nature of the image also. The self-feedback, as discussed before, is taken to be proportional to the radius of the neighborhood. To prevent the noise to be detected as edge pixels, the self-feedback should be large enough so that the strength of the redundant edge pixels can reduce to zero with the emergence of the states of the processing elements. On the other hand, the amount of self-feedback should be small enough to ensure that the response of the true edge pixels are not reduced causing discontinuity. In ideal case, any edge point on a true edge will get support from $2R$ edge points (this is true when the edge is aligned with x- or y-axes). The edge point receives slightly less support, if the edge is not aligned with either x- or y-axis. The self-feedback can be chosen slightly less than $2wR$. In the present system, the value of self-feedback was selected to be $180/t$.

The network was simulated to find out the edges of two different images as shown in Fig. 10 and Fig. 11. The results show the edges linked for different radii of neighborhood (radius 2 and 3). It is evident from the results that the diffusion of edges is more with higher radius, and consequently more thick edges appear (Fig. 11(b) and 11(d)). This causes comparatively more smoothing with radius 3 and also better linking which also supports the theory of edge induction as posed here.

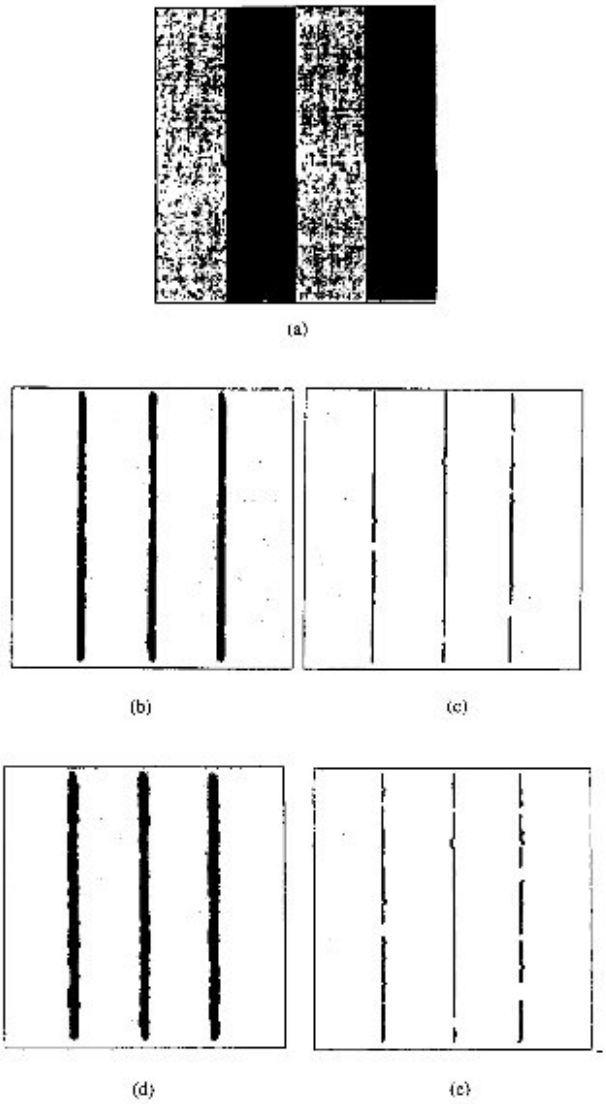

Fig. 10. (a) Image (125×125) of noisy vertical bars. (b) Diffused edge cover (initial neighborhood, $R(0) = 2$). (c) Edges detected by lateral nonmaximum suppression from (b). (d) Diffused edge cover (initial neighborhood radius, $R(0) = 3$). (e) Edges detected by lateral nonmaximum suppression of (d). [$w = 100$, $w_s = 180$, $\lambda = 0.6$].

VI. Discussion and Conclusions

In the present paper, two different connectionist models have been presented. The models have been designed for finding out the continuous line and edge segments in a graylevel image. The state updation rules and the corresponding convergence proof of the network have also been presented. Using these models the disadvantage of seed point selection in sequential line or edge linking is avoided.

Both the network models, proposed here, work on the basis of neighborhood information. The first model relies on the information from 8-neighborhood. Two different updation rules have been provided for the first model. Due to the presence of noise, two consecutive line or edge pixels on a segment may be lost. The first updation rule (6) is able to restore such kind of defects in a better way compared to the second updation rule (12). On the other hand, the noise-reduction capability of the second updation rule is much better than that of the first updation rule. Therefore, both the updation rules have been interleaved in linking the line segments. In the second model, larger neighborhood has been considered, and as a result, larger line or edge segments may be restored.

In the second model, the effect of noise is less severe than that in the first one. This is due to the fact that the strengths of noisy pixels are compensated by the pixels only in an 8-neighborhood in the first model. On the other hand, in the second model, the effect of noise is compensated by the pixels over a larger neighborhood. Although the noise reduction is better in the second model, the effect of smoothening of the edges or lines is more in this case.

In the first model, the open-ended edge or line segments do not suffer lengthening. The output of a processing element, in this case, is updated only if it receives support from two neighbors (using second updation rule (12)). As a result, the states of the processing elements corresponding to the end points of the lines and edges are not updated (since they

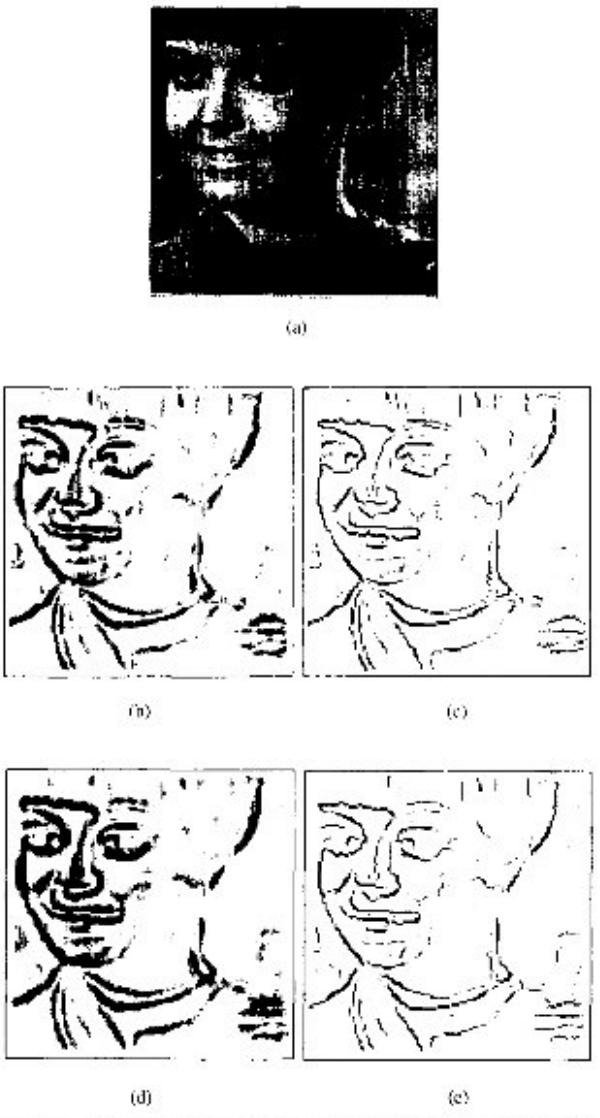

Fig. 11. (a) Image (128×128) of a noisy girl. (b) Diffused edge cover (initial neighborhood. $R(0) = 2$). (c) Edges detected by lateral nonmaximum suppression from (b). (d) Diffused edge cover (initial neighborhood radius. $R(0) = 3$). (e) Edges detected by lateral nonmaximum suppression of (d). $[\alpha = 100, w_0 = 180, \lambda = 0.6]$.

receive input only from single neighbor). Due to the presence of self-feedback, the output of PEs corresponding to end pixels decrease to some extent. If the first updation rule (6) is used then the line segments can increase in length at their ends to an extent. Since the first and second updation rules are interleaved, they compensate each other, and as a result, the open ends of the line segments may be shortened slightly.

In the second model, the edge or line segments are lengthened with reduced strength at their open ends. This is due to the fact that the edge points at the open ends recursively induce edge vectors over a large neighborhood. The amount of lengthening depends on the rate at which the radius of neighborhood is decreased. In real life images, open edge segments are less frequent than the closed loops of edges.

In the second connectionist model, the selection of initial radius of the neighborhood is more or less dependent on the average separation of the edge segments in an image. Moreover, at the junction of different edges, a gap may be created due to cross induction (i.e., simultaneous induction in different directions due to different edges). This phenomenon does not happen in the first model, because in this case information about more than one type of line segment is present in each processing element. In the second model, the final result is obtained by selecting the points corresponding to the spine of the induced edge cover. There are many different techniques, available in literature [3] to perform this selection. Here, non-maximum suppression technique is used for sake of simplicity.

In the second connectionist model, the neighborhood of activity is considered to be circular for each processing element. Therefore, the extent to which an edge can induce edge points is same along any direction. The better results may be obtained if elliptical neighborhood is used instead of circular one. The major axis of the ellipse should be aligned along the edge. The selection of the ratio of the major and minor axes may be a plausible problem.

Although the works presented here are conceptually analogous to the relaxation labeling algorithms [11], [12], [15], a new direction of implementing the cooperative processes (required for the linking) onto neural network models is the main contribution. Secondly, the present investigation comes up with a couple of new connectionist models which may be applied to some other problems apart from line and edge linking. Moreover, the models may provide a better understanding of the human psychovisual phenomenon. Although the characteristics of preattentive vision were extensively studied by Grossberg and Mingolla [29], the realization of the neural modeling was lacking there. The current investigation may find its use as a step toward establishing the bridge between the classical image processing techniques and cognitive psychology.

The linking of the broken line (or edge) segments would be better if there is feedback from the higher-level entities. Grossberg and Mingolla [29], [30] have described the region filling-in and the perceptual grouping phenomena in this context. The connectionist realization of these processes should require both feed-forward and feed-back paths between lower- and higher-level entities. The present work is a part of the investigation for building up connectionist models for visual pattern recognition using feed-forward and feed-back connections [31]–[33].

Appendix I

Let us consider $0 \leq \alpha \leq \pi/2$, and $0 < \phi < \pi/2$, where ϕ is the angle subtended by PQ with x-axis. P, is the point whose effect on Q (due to edge induction) is currently being considered (Fig. 4). The edge vector induced at Q is considered to be perpendicular to the direction of PQ (this is governed by the nature of the proposed edge point induction). Therefore,

the induced edge e' due to P will subtend an angle $\pi/2+\phi$ with the x-axis. The induced edge strength is given as

$$\begin{aligned} e' &= e\cos\left(\frac{\pi}{2}+\phi-\alpha\right) \\ &= e\sin(\alpha-\phi) \\ &= e\sin\alpha\cos\phi - e\cos\alpha\sin\phi. \end{aligned}$$

The components of e along the horizontal and vertical directions are $e_x = e\cos\alpha$ and $e_y = e\sin\alpha$, respectively. Therefore,

$$e' = e_y\cos\phi - e_x\sin\phi. \tag{44}$$

The horizontal and vertical components of e' are

$$\begin{aligned} e'_x &= e'\cos\left(\frac{\pi}{2}+\phi\right) \\ &= -e'\sin\phi \end{aligned}$$

and

$$\begin{aligned} e'_y &= e'\sin\left(\frac{\pi}{2}+\phi\right) \\ &= e'\cos\phi \end{aligned}$$

respectively. Substituting the value of e' (from (44)) we get

$$\begin{aligned} e'_x &= e_x\sin^2\phi - e_y\cos\phi\sin\phi \\ e'_y &= e_y\cos^2\phi - e_x\cos\phi\sin\phi. \end{aligned} \tag{45}$$

Similarly, for $\phi > \alpha, e'$ will be perpendicular to PQ subtending an angle $\phi - \pi/2$ with the positive direction of x-axis. The induced edge strength is

$$\begin{aligned} e' &= e\cos(\phi-\pi/2-\alpha) \\ &= -e\sin(\alpha-\phi) \end{aligned}$$

i.e.,

$$e' = e_x\sin\phi - e_y\cos\phi. \tag{46}$$

Therefore,

$$\begin{aligned} e'_x &= e'\cos(\phi-\pi/2) \\ &= e'\sin\phi \end{aligned}$$

and

$$\begin{aligned} e'_y &= e'\sin(\phi-\pi/2) \\ &= -e'\cos\phi \end{aligned}$$

Substituting the value of e' (46), the above expressions become

$$\begin{aligned} e'_x &= e_x\sin^2\phi - e_y\sin\phi\cos\phi \\ e'_y &= e_y\cos^2\phi - e_x\sin\phi\cos\phi. \end{aligned} \tag{47}$$

The equations for e'_x and e'_y can be proved to be true in all quadrants for all possible values of α and ϕ.

Appendix II

Consider any two edge points M and N on the edge AB, such that $MP = PN = x$. PP' is perpendicular to AB (Fig. 5). In this case, the edges at P' induced by edge points M and N subtend equal and opposite angles with PP', and as a result, an edge vector along PP' is induced (the edge strengths at M and N are considered to be equal). The strength of the edge induced at P due to edge point M is $e'_1 = e\cos\theta$ where θ is the angle of the induced edge with PP'. The edge induced by the edge point N has the same strength and subtends an angle $-\theta$ with PP'. Therefore, the resultant induced edge strength (due to M and N) is $e'_{12} = 2e\cos^2\theta$.

On the other hand $\tan\theta = D/x$. Therefore the induced edge strength can be given as

$$\begin{aligned} e'_{12} &= \frac{2e}{1+\left(\frac{D}{x}\right)^2} \\ &= \frac{2ex^2}{D^2+x^2}. \end{aligned}$$

Since all such pair of points, like M and N, lying on AB and equidistant from P, induce an edge along PP', the resultant response can be given as

$$\begin{aligned} e' &= \int_0^L \frac{2ex^2}{D^2+x^2}dx \\ &= 2e(x - D\tan^{-1}(L/D)) \end{aligned}$$

i.e.,

$$e' = 2eL\left(1 - \frac{\tan^{-1}(L/D)}{(L/d)}\right). \tag{48}$$

Appendix III

From Appendix II it can be said that the increase in edge response at any point P' (Fig. 5) in time dt is

$$de' = 2ew(\sqrt{R^2-D^2} - D\tan^{-1}(\sqrt{(R/D)^2-1}))dt. \tag{49}$$

Let the radius of the neighborhood decreases linearly at a rate γ; i.e., $dR/dt = -\gamma$, or $R = R_0 - \gamma t$, where R_0 is the initial radius at $t = 0$. It is clear that if $R < D$, then $de' = 0$. Therefore, the resultant induced edge strength at a point P' due to presence of an edge AB is

$$e' = \frac{2ew}{\gamma}\int_D^{R_0}(\sqrt{R^2-D^2} - D\tan^{-1} + (\sqrt{(R/D)^2-1}))dR. \tag{50}$$

After simplification, the expression reduces to

$$\begin{aligned} e' = \frac{2ew}{\gamma}\Bigg[&\frac{1}{2}R_0\sqrt{R_0^2-D^2} + \frac{D^2}{2}\log\left(\frac{R_0}{D}\right) + \sqrt{(R_0/D)^2-1}\Bigg] \\ &- R_0D\tan^{-1}(\sqrt{(R_0/D)^2-1}). \end{aligned} \tag{51}$$

The expression has been derived without considering the induction from other edges in the neighborhood and the self-inhibition. From the expression (51) of induced edge strength it is clear that the induced edge strength will be more if P' is nearer to AB and will be less otherwise. To avoid saturation of induced edge strength at any point in the neighborhood of an

edge segment like AB, we must have ϵ' to be less than unity for any position of P'. Since ϵ' increases as D decreases, we must consider $R_0 \gg D$. Then we have

$$\epsilon' = \frac{2\epsilon w}{\gamma}\left[\frac{1}{2}R_0^2 + \frac{1}{2}D^2\log(2R_0/D) \quad R_0 D\tan^{-1}(R_0/D)\right]. \tag{52}$$

as D becomes very small with respect to R_0, i.e., $D/R_0 \to 0$, the expression reduces to

$$\epsilon' = \frac{\epsilon w R_0^2}{\gamma}. \tag{53}$$

Since we require ϵ' to be less than unity, we must have

$$\gamma > \epsilon w R_0^2. \tag{54}$$

Acknowledgment

The authors greatly acknowledge the reviewers of the article for their helpful suggestions.

References

[1] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Royal Society London B*, vol. 207, pp. 187–287, 1980.
[2] J. F. Canny, "Finding edges and lines," Master's thesis, Massachusetts Institute of Technology, 1983.
[3] A. Rosenfeld, "Digital straight line segments," *IEEE Trans. Computer*, vol. 23, pp. 1264–1269, 1974.
[4] W. Frei and C. C. Chen, "Fast boundary detection: A generalization and a new algorithm," *IEEE Trans. Computer*, vol. 26, pp. 988–998, 1977.
[5] G. J. Vanderbrug, "Semilinear line detectors," *Computer Vision, Graphics and Image Process.*, vol. 4, pp. 287–293, 1975.
[6] G. J. Vanderbrug and A. Rosenfeld, "Linear feature mapping," *Ibid: Trans. Syst. Man, Cybern.*, vol. 8, pp. 768–774, 1978.
[7] R. O. Duda and P. E. Hart, "Use of Hough transform to detect lines and curves in pictures," *Communications ACM*, vol. 15, pp. 11–15, 1972.
[8] J. Sklansky, "On the Hough transform techniques for curve detection," *IEEE Trans. Computer*, vol. 27, pp. 923–926, 1978.
[9] S. D. Shapiro, "Feature space transforms for curve detection," *Pattern Recog.*, vol. 10, pp. 129–143, 1978.
[10] D. Ballard and C. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice Hall, 1982.
[11] S. W. Zucker, R. A. Hummel, and A. Rosenfeld, "An application of relaxation labelling to line and curve enhancement," *IEEE Trans. Computer*, vol. 26, pp. 394–403, 1977.
[12] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labelling by relaxation operations," *IEEE Trans. Syst., Man, Cybern.*, vol. 6, pp. 420–433, 1976.
[13] A. Mansouri, A. S. Malowany, and M. D. Levine, "Line detection in digital pictures: A hypothesis prediction/verification paradigm," *Computer Vision, Graphics and Image Process.*, vol. 40, pp. 95–114, 1987.
[14] S. Vasudevan, R. L. Cannon, J. C. Bezdek, and W. L. Cameron, "Heuristics for intermediate level road finding algorithm," *Computer Vision, Graphics and Image Process.*, vol. 44, pp. 175–190, 1988.
[15] C. David and S. W. Zucker, "Potentials, valleys, and dynamic global coverings," Tech. Rep. TR-CIM-89-1, Computer Vision and Robotics Laboratory, McGill University, Canada, 1989.
[16] S. Zucker, A. Dobbins, and L. Iverson, "Two stages of curve detection suggest two styles of visual computation," *Neural Computation*, vol. 1, pp. 68–81, 1989.
[17] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. 4, pp. 4–22, 1987.
[18] B. M. Forrest, D. Roweth, N. Stroud, D. Wallace, and G. Wilson, "Neural network models," *Parallel Computing*, vol. 8, pp. 71–83, 1988.
[19] S. E. Fahlmann and G. E. Hinton, "Connectionist architecture for artificial intelligence," *IEEE Computer*, vol. 20, pp. 100–109, 1987.
[20] J. A. Feldman, "Dynamic connections in neural networks," *Biological Cybern.*, vol. 46, pp. 27–39, 1982.
[21] J. A. Fahlmann and D. H. Ballard, "Connectionist models and their properties," *Cognitive Sci.*, vol. 6, pp. 205–254, 1982.
[22] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. National Acad. Sci., USA*, pp. 2554–2558, 1982.
[23] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two state neurons," in *Proc. National Acad. Sci., USA*, pp. 3088–3092, 1984.
[24] T. Kohonen, *Self-Organization and Associative Memory*. Berlin: Springer-Verlag, 1988.
[25] D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing: Explorations in Microstructures of Cognition, vol. I*. Cambridge, MA: Bradford Books/MIT Press, 1986.
[26] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimizing problems," *Biological Cybern.*, vol. 52, pp. 141–152, 1985.
[27] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1273–1290, 1988.
[28] R. B. Ash, *Real Analysis and Probability*. London: Academic Press, 1972.
[29] S. Grossberg and E. Mingolla, "Neural dynamics of form perception: Boundary completion, illusory figures, and neon color spreading," *Psych. Rev.*, vol. 92, pp. 173–211, 1985.
[30] S. Grossberg and E. Mingolla, "Neural dynamics of perceptual grouping: Textures, boundaries, and emergent segmentations," *Perception and Psychophysics*, vol. 38, pp. 141–171, 1985.
[31] J. Basak, B. Chanda, and D. Dutta Majumder, "An approach to line linking based on neural network model," in *Proc. Neuro-Nimes*, pp. 491–506, 1991.
[32] J. Basak, C. A. Murthy, S. Chaudhury, and D. Dutta Majumder, "A connectionist model for category perception: Theory and implementation," *IEEE Trans. Neural Networks*, vol. 4, pp. 257–269, 1993.
[33] J. Basak, C. A. Murthy, S. Chaudhury, and D. Dutta Majumder, "A connectionist network for simultaneous perception of multiple categories," in *Proc. Eleventh IAPR Int. Conf. on Pattern Recognition*, 1992.

Jayanta Basak completed his undergraduate studies in electronics and telecommunications engineering from Jadavpur University, Calcutta, India, in 1987. He received the M.E. degree in computer science and engineering from the Indian Institute of Science, Bangalore, in 1989.

He served as a computer engineer in National Centre for Knowledge Based Computing, Calcutta, from 1989 to 1992. In April 1992 he joined as a programmer in ECSU, ISI. Presently, he is working in the Machine Intelligence Unit of the same institute. His current research interests include neural networks and computer vision.

Bhabatosh Chanda (S'82–M'85) received the B.E. degree in electronics and telecommunication engineering and the Ph.D. degree in electrical engineering from the University of Calcutta in 1979 and 1988 respectively.

In 1979 he joined the Electronics and Communication Sciences Unit, Indian Statistical Institute, as a Research Fellow where he is presently working as an Associate Professor. He has been actively engaged in research and development in the field of image processing, pattern recognition, and computer vision for more than a decade.

Dr. Chanda also worked at Intelligent System Lab at the University of Washington as a UNDP Fellow. He is a recipient of Indian National Science Academy medal for young scientist and also is a recipient of Diamond Jubilee Fellowship Award from the National Academy of Science, India.

Dwijesh K. Dutta Majumder obtained the M.Sc.(Tech.) degree in radio physics and electronics and the Ph.D. degree in digital computers' memory technology in 1955 and 1962 respectively from Calcutta University.

Since 1955 he has been working in ISI in different capacities and was Professor and Head of the Electronics and Communication Science Unit for twenty years. In 1992 Professor Dutta Majumder was made Professor Emeritus of ISI. Emeritus Scientist of CSIR, and Chairman of the National Centre for Knowledge Based Computing at ISI. He is the author of more than 325 research papers in international journals and conference proceedings and is the author/editor of six books all published by John Wiley and Sons in the areas of memory technology, pattern recognition, man–machine communication, image processing, computer vision, artificial intelligence, expert systems, cybernetics systems theory, data communication, neural modeling, and fuzzy mathematics. He visited and delivered series of lectures on his work in some of the above-mentioned subjects in large number of universities and industrial Labs in the USA, U.K., Japan, and Europe as a Guest Professor several times. He is a member of the editorial boards of *Pattern Recognition Letters* (PRL), *Indian Journal of Pure and Applied Mathematics* (IJPAM), the *Journal of Fuzzy Mathematics*, and is the consulting editor of the *Journal of Computer Division* of the Institution of Engineers (India).

Dr. Dutta Majumder is a Fellow of the Indian National Science Academy, Indian National Academy of Engineering, the National Academy of Sciences, Computer Society of India and Institution of Electronics and Telecommunication Engineers, and is a member of the governing Board of IAPR, Director of the World Organization of Cybernetics and Systems, President of Indian Society of Fuzzy Mathematics and Information Processing (ISFUMIP) and Indian Association for Pattern Recognition and Artificial Intelligence (IAPRAI). He is a recipient of the Sir J. C. Bose Award of IETE, Sir C. V. Raman Award of ASI, Norbert Wiener Award of WOSC, R. L. Wadhwa Gold Medal of IETE, P. C. Mahalanobis Gold Medal of INSA, and several other honors and awards.