A Connectionist Model for Corner Detection in Binary and Gray Images

Jayanta Basak, *Senior Member, IEEE,* and Debashis Mahata

***Abstract*—A connectionist model along with its state dynamics is developed for detecting corner points in binary and gray images. For a given binary/gray image, each pixel in the image is assigned with some initial cornerity (our measurable quantity) which is a vector representing the direction and strength of the corner. These cornerities are then mapped onto a neural-network model which is essentially designed as a cooperative computational framework. The cornerity at each pixel is updated depending on the neighborhood information. After the network dynamics settles to stable state, the dominant points are obtained by finding out the local maxima in the cornerities. Theoretical investigations are made to ensure the stability and convergence of the network. It is found that the network is able to detect corner points even in the noisy images and for open object boundaries. The dynamics of the network is extended to accept the edge information from gray images also. The effectiveness of the model is experimentally demonstrated in synthetic and real-life binary and gray images.**

***Index Terms*—Corner detection, cornerity vector, gray-level corner detection, neural network.**

I. Introduction

CORNER is considered to be one of the important image features, and the detection of corner points plays a significant role in many vision problems including shape analysis, object recognition, scene analysis, motion detection, and stereo matching [1], [2]. In digital images, the points with high curvature values are considered as the corner points.

Early attempts to find the corners or high curvature points or the dominant points include the methods developed by Rosenfeld and Johnston [3], Rosenfeld and Weszka [4], Freeman and Davis [5] and many others including [6]. The basic approach in these attempts is to detect the dominant points directly through the measurement of angle at the prospective corner points, resulting in computationally expensive algorithms. Piecewise linear approximation to digital arcs has been used in [7], [8], and the points of intersections of the adjacent linear segments are detected as corner points. In a different approach by Wang *et al.* [9], a bending value for every point on a digital arc is computed. The bending value at a point, in turn, provides a measure of curvature and thereby the cornerness at the corresponding point. In this method, direct computation of angle between adjacent segments is replaced by the estimation of bending value which involves only addition and subtraction operations, leading to faster computation. Several algorithms have been developed based on the curvature estimates of the object boundaries. Mokhtarian and Soumela [10] presented a corner detection algorithm through curvature scale space where corners were first detected at higher scale (coarser level) and then tracked through multiple lower scales for better localization. A nice description of the state-of-the-art corner detection algorithms has also been presented in [10]. Sohn *et al.* [11] presented a mean field annealing-based boundary smoothing for estimation and subsequent robust corner detection technique. Arrebola *et al.* [12] used local histograms of contour chain code for computing curvature followed by the detection of dominant points. These algorithms generally accept closed object boundaries only. In [10], special linking is performed to obtain closed boundaries. Moreover, the algorithms are not directly extendible for accepting gray images.

Corner detection algorithms have also been developed for accepting gray level images directly. Zheng *et al.* [13] provided a good literature survey of the existing gray level corner detection algorithms. They proposed an improvement over the Plessy corner detector (see [13]) which provides a cornerness measure based on the intensity changes along horizontal and vertical directions. Stammberger *et al.* [14] provided a faster corner detection scheme by introducing a set of orthogonal second-order Gaussian derivative kernels. Trajkovioca [15] computed the intensity change in different directions and subsequently defined a corner response function. The graylevel corner detectors, in general, perform second-order differential geometric analysis of the image intensity surface and define certain cornerness measures. The performance of the algorithms is dependent on the cornerness measure itself.

Apart from the algorithms mentioned above, various other corner detectors have been developed including those based on morphological operators [16], fuzzy set theoretic tools [17] and neural networks [18], [19]. In [18], a multilayered perceptron (MLP) is trained for detecting the corners. The MLP-based approach [18] accepts closed object boundaries and applicable for binary images only. Dias *et al.* [19] also employed a multilayered neural architecture which learns (supervised mode) certain generic corner images of different angles. In a test image, the pixels are classified either as corner or noncorner points. Although for noiseless images the algorithm exhibits good performance, for noisy images the performance deteriorates rapidly.

Here we developed a method for detecting corner points in binary and gray images using neural network. In the network model, a pair of nodes corresponds to a pixel in the image. The output of the pair of nodes together represent the corner vector of the corresponding pixel. Each pair of neurons is laterally connected over a neighborhood, and the node activations are updated by the neighborhood information. Initial cornerity infor-

Manuscript received April 24, 1999; revised March 28, 2000.

The authors are with the Machine Intelligence Unit, Indian Statistical Institute, Calcutta, 700 035, India.

Publisher Item Identifier S 1045-9227(00)06026-4.

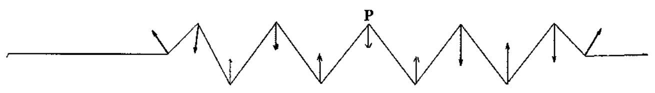


Fig. 1. Suppression of corners in a jagged edge.

mation of each pixel is assigned to the corresponding pair of nodes, and the final corner points are obtained after the network converges to a stable state. Here no explicit information about the entire object boundary is required, and only the local information can update the output of each node. As a result, the network is able to find out corner points from open object boundaries also. The network exhibit robust performance against the presence of noise. The dynamics of the network is then extended to accept the cornerity information from gray images by embedding the edge strength information. A graceful performance by the network is observed for the gray images.

II. Corner Detection in Binary Images

A. Overall Methodology

Here we assume that the given image consists of only the boundary information, the boundary not necessarily being a closed one. In the case of a gray level image (see Section IV), edge/line segments are considered. The measurable parameter, cornerity is a vector, the magnitude of which gives the strength of corner and the direction gives the direction of the corner at that point. The cornerity at every point on the boundary is initialized considering only its 3 $\times$ 3 neighborhood. Note that, in discrete domain, in a 3 $\times$ 3 neighborhood only 16 different types of patterns [20] are possible. The updating process is such that the cornerity vectors are enhanced at the true corner points and get suppressed at other points. The corner points are then identified by finding out the points of local maxima in the magnitude of cornerity vectors.

Let us consider an example of an edge segment (line boundary) as shown in Fig. 1. The initial corner vectors at each point are shown in the Fig. 1. If we consider a larger scale space, i.e., a coarse description of the edge/boundary segment then it appears to be a smooth one. On the other hand, a finer description of the same gives rise to jagged nature of the segment. If we provide a coarse description then P should not be treated as a corner point. This can be obtained by adding the cornerity vectors of the neighbors with that of P resulting in a zero (or very close to zero) cornerity vector at P. The coarse or fine description can be related to the size of the concerned neighborhood. As in Fig. 1, if we select a larger neighborhood around P then the vector sum of the cornerities will be closed to zero, i.e., the segment appears to be a smooth one. On the other hand, for a small neighborhood size (Fig. 2) no oppositely directed corners in the neighborhood affect each other. In Fig. 2, although Q does not have any initial cornerity, it will get some induced cornerity value from P. The induced vector has the same direction as the corner vector at P. The induced cornerity value at Q, in turn, gives some induction to P, resulting in an enhancement of the cornerity at P. Thus in the updating process all points will have nonzero cornerity vectors with the point P having a local maxima.

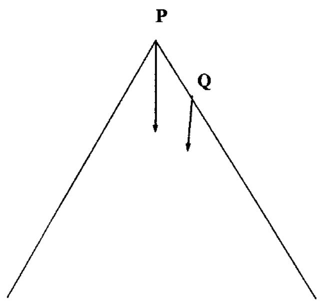


Fig. 2. Finer detail of the jagged edge.

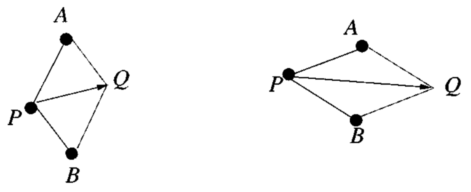


Fig. 3. Two typical case of initializing the cornerity vector.

The neural network model consists of $2m \times 2n$ neurons for an $m \times n$ image. A pair of neurons (nodes) corresponds to a single pixel. The activation of the pair of neurons together represent the cornerity vector at the corresponding pixel. Each node is connected to its surrounding neurons over a neighborhood. The connections are only between neurons of same type. Each node has a negative self-feedback which is essential for eliminating noise. Initially, the state of the neurons are clamped by the initialized cornerity vectors in the input image. The state of the neurons are then updated according to the neighboring information. After initialization, the external input is no more required.

B. Initialization of Cornerity Vectors

Initialization of the states of neurons in the network is very important in the sense that the neurons in the network update their states starting from the initial cornerity vectors only. For example, a point lying on a straight line segment should have zero initial vector, and a boundary point at a very high curvature region should get high initial cornerity. The initial cornerity vector at a pixel is determined over its 3 $\times$ 3 neighborhood. The resultant vector of the relative position vectors of the neighboring points with respect to the center point, gives the direction of the cornerity at the center point. The magnitude of this resultant vector is a measure of the strength of the cornerity. This is illustrated in Fig. 3 which illustrates the two typical situations. In Fig. 3 black blobs represent the boundary pixel positions. The center pixel is denoted by P, and the neighboring pixels are denoted by A and B respectively. The angle between two arms at P is small in the case shown in the right side of Fig. 3 than that in the case shown in the left side of Fig. 3. This implies that the initial cornerity vector is larger in the case shown in the right side of Fig. 3 than the other. In Fig. 3, $\overrightarrow{PQ}$ represents the initialized vector which is the resultant of $\overrightarrow{PA}$ and $\overrightarrow{PB}$. The resultant cornerity vector $\overrightarrow{PQ}$ can then be represented by two components, horizontal and vertical, respectively. Mathematically, let (i, j) be a boundary pixel and $(i + l_1, j + k_1)$ and $(i + l_2, j + k_2)$ be

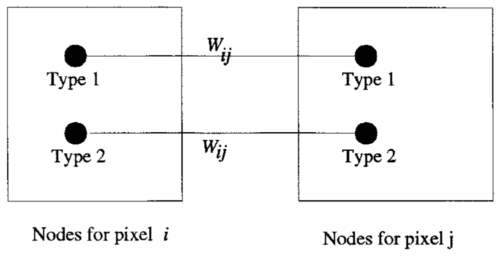


Fig. 4. Interaction between the nodes of two different pixels.

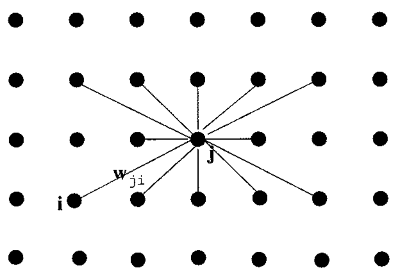


Fig. 5. Lattice structure of a given type of nodes.

the two neighboring boundary pixel positions in a 3 × 3 neighborhood where $\{l_1, l_2, k_1, k_2\} \in \{-1, 0, +1\}$. Then the horizontal and vertical components of the initial cornerity vector at the pixel position (i, j) is given by

$$c_x = \frac{l_1}{\sqrt{l_1^2 + k_1^2}} + \frac{l_2}{\sqrt{l_2^2 + k_2^2}}$$

and

$$c_y = \frac{k_1}{\sqrt{l_1^2 + k_1^2}} + \frac{k_2}{\sqrt{l_2^2 + k_2^2}}$$

respectively. A point is assigned a nonzero initial cornerity only if it is a boundary pixel and it has exactly two neighboring boundary pixels. Thus the terminal points on an open boundary or the nonboundary points are not assigned any cornerity (i.e., zero cornerity).

C. Network Model

Corresponding to each pixel, there are two different types of neurons. Each neuron is connected with the other neurons of the same type over a neighborhood (Figs. 4 and 5). The output of two types of nodes are denoted by c_x and c_y and their internal states are denoted by u and v, respectively. c_{x_j} and c_{y_j}, represent the horizontal and vertical components, respectively, of the corner vector at the jth pixel. Node j is connected to its neighboring nodes (denoted by index i) of same type with weight w_{ji} (for both types) if node i is within a given neighborhood $N(j)$ of j. The connection weights are symmetric in nature, i.e., $w_{ji} = w_{ij}$. The state dynamics of the processing element j is given by

$$\frac{du_j}{dt} = \sum_{i \in N(j)} w_{ji} c_{x_i} - w_s c_{x_j} \tag{1}$$

and

$$\frac{dv_j}{dt} = \sum_{i \in N(j)} w_{ji} c_{y_i} - w_s c_{y_j}. \tag{2}$$

The output of the node j is given by

$$c_{x_j} = g(u_j) \text{ and } c_{y_j} = g(v_j) \tag{3}$$

respectively. $g(.)$ is a ramp function given by

$$g(x) = \begin{cases} m & \text{if } x > m \\ -m & \text{if } x < -m \\ x & \text{otherwise} \end{cases} \tag{4}$$

where m is the saturation level of the ramp function. The second term in (1) and (2) are negative feedback terms used to eliminate the noise points. w_s is the weight of the negative feedback. The neighborhood $N(j)$ chosen here is a circular one. The radius r of the neighborhood is decreased with time resulting in higher interaction between the nearby nodes as compared to that between the distant ones. If r is decreased very fast then the nodes will not interact properly and the desired smoothing of the boundary segment may not be obtained. On the other hand, a very slow decrease in r may smooth out the true dominant points on a boundary. Here r is decreased such that [21]

$$\lim_{t \to \infty} r(t) = 0 \quad \text{and} \quad \sum_t r(t) \to \infty.$$

In this model the radius r follows a schedule, given by

$$r = \frac{k}{1 + bt}. \tag{5}$$

The parameters k and b are positive constants determining the initial radius of the neighborhood and the rate of decrement of the radius. The weights w_{ij} and w_s are also proportional to the radius of the neighborhood, i.e.,

$$w_{ij} = w_{ji} = w_1 r \quad \text{and} \quad w_s = w_2 r \tag{6}$$

where w_1 and w_2 are the constants of proportionality.

D. Convergence of the Network

Evidently, if the network converges for a neighborhood of fixed radius then it must converge for shrinking neighborhood also. Consider a Lyapunov (or the energy function) of the network for a given r

$$E = -\frac{1}{2} \sum_i \sum_{j \in N(i)} w_{ij} c_{x_i} c_{x_j} + \frac{1}{2} w_s \sum_i c_{x_i}^2 - \frac{1}{2} \sum_i \sum_{j \in N(i)} w_{ij} c_{y_i} c_{y_j} + \frac{1}{2} w_s \sum_i c_{y_i}^2. \tag{7}$$

Therefore, from (1) to (3), dE/dt is given by

$$\frac{dE}{dt} = -\sum_i g^{-1'}(u_{x_i}) \left(\frac{dc_{x_i}}{dt} \right)^2 - \sum_i g^{-1'}(u_{y_i}) \left(\frac{dc_{y_i}}{dt} \right)^2 \tag{8}$$

where $g^{-1'}$ is the first derivative of the of the inverse of $g^{-1}(.)$, which exists and is an increasing function, provided $-m < u < m$ and $-m < v < m$. The parameters are selected in such a way that this condition is satisfied. The way of selecting the parameters is described in the next section (i.e., in Section III). Since, in the given range g^{-1} is an increasing function, i.e., $g^{-1'}$ is positive, and $(dc_{x_i}/dt)^2$ and $(dc_{y_i}/dt)^2$ are always nonnegative, $dE/dt \leq 0, \forall t \geq 0$. Since $E(t)$ is bounded, $dE/dt \to 0$ as $t \to \infty$ and therefore $dc_{x_i}/dt \to 0$ and $dc_{y_i}/dt \to 0$ as $t \to \infty$ for all i. As a result, the output values of all processing elements converge.

III. Selection of Network Parameters

The analysis in this section, is done considering c_x (i.e., the horizontal part) components only. The same arguments are valid for c_y component also. Let us use the notation c_j instead of c_{x_j} in the sequel. Let u_0 be the maximum initialized value of c_j, θ be the threshold of the resultant cornerity value such that if the magnitude of cornerity vector is less than θ for some node j after the convergence of the network then the corner at j is eliminated, and m is the saturation level of the ramp function.

Theoretically, a noise point is one which initially have some cornerity but does not get support from the neighborhood. Therefore, for a noise point [from (1)]

$$\frac{du_j}{dt} = -w_s c_j. \tag{9}$$

Assuming that the value of u_j is within the saturation limit of the ramp function, we replace c_j in (9) by u_j, i.e., [from (6)]

$$\frac{du_j}{dt} = -w_2 r u_j. \tag{10}$$

It is required that after convergence of the network, the cornerity at the noise points should less than θ so that they can be eliminated. Let us consider a noise point with maximum initialization u_0. From (5) and (10)

$$u_j(t) = \frac{u_0}{(1+bt)^B} \tag{11}$$

where B is a constant given by

$$B = (w_2 k)/b. \tag{12}$$

Since the network dynamics is stopped when $r < 1$ (let at time instant $t = t_m$), the value of $u_j(t)$ in (11) should satisfy

$$u_j(t_m) \leq \theta. \tag{13}$$

From (5) $r = 1$ implies that

$$1 + bt_m = k. \tag{14}$$

From (11), (13), and (14)

$$\frac{u_0}{k^B} \leq \theta. \tag{15}$$

Therefore

$$B \geq \log_k \frac{u_0}{\theta}. \tag{16}$$

This equation gives a lower bound on B, i.e., weight of the self-feedback [from (12)].

Again, from (1) and (6)

$$\frac{du_j}{dt} = \sum_{i \in N(j)} w_1 r u_i - w_2 r u_j. \tag{17}$$

In the updating process, a maximum level of activation can be attained by an individual neuron if each neuron in the network has got a maximum initial activation u_0. Again, according to (17), the network behavior is isotropic, i.e., the activation of all nodes grow symmetrically. This means that each node gets the same contribution from its neighborhood. Thus the internal state u_j is independent of index j, i.e., $u_j = u_i$. Therefore, the dynamics of node j can be written as

$$\frac{du_j}{dt} = G_j(t) \tag{18}$$

where

$$G_j(t) = \sum_{i \in N(j)} w_1 r u_i - w_2 r u_j. \tag{19}$$

From (18)

$$\int_0^t \frac{du_j}{u_j} = \int_0^t \frac{G_j}{u_j}\, dt$$

i.e.,

$$\log u_j - \log u_0 = \int_0^t \frac{G_j}{u_j}\, dt. \tag{20}$$

Using the expression of G_j from (19)

$$\frac{G_j}{u_j} = \frac{1}{u_j} \sum_{i \in N(j)} (w_1 r u_i) - w_2 r.$$

Since we assumed that all the nodes in the network grow in the same way, and their initial activations are the same (i.e., u_0), we can infer that at any time instant t

$$\frac{G_j}{u_j} = \sum_{i \in N(j)} (w_1 r) - w_2 r. \tag{21}$$

For a boundary point the number of points in the neighborhood is proportional to its radius r and say the constant of proportionality is p. Then

$$\frac{G_j}{u_j} = p w_1 r^2 - w_2 r. \tag{22}$$

On an average, we can assume that p is equal to two. This is due to the fact that the number of boundary points in a circular neighborhood (radius r) of a particular boundary point is equal to $2r$ if both the arms are straight line segments. If the arms are curved segments then the concerned point may not be a sharp corner.

From (5), (20), and (22), we get

$$\log u_j(t) = \int_0^t \left(\frac{2w_1 k^2}{(1+bt)^2} - \frac{w_2 k}{1+bt} \right) dt + \log u_0 \tag{23}$$

i.e.,

$$\log u_j(t) = A\left(1 - \frac{1}{1+bt}\right) - B \log(1+bt) + \log u_0 \quad (24)$$

where

$$A = (2w_1 k^2)/b. \quad (25)$$

From (24), it can be shown that u_j increases for small values of t, reaching maximum at some point of time (say $t = t_{\max}$) and the decreases to zero. The value of $t_{\max}$ can be found out by setting $(du_j/dt) = 0$

$$1 + bt_{\max} = \frac{A}{B}. \quad (26)$$

The maximum value of u_j can be found out from (24) and (26), which is

$$(u_j)_{1+bt=(A/B)} = \exp\left(A\left(1 - \frac{B}{A}\right) - B \log \frac{B}{A} + \log u_0\right). \quad (27)$$

If m is the saturation level of the ramp function then it is required that $(u_j)_{1+bt=(A/B)} \leq m$, i.e.,

$$A\left(1 - \frac{B}{A}\right) - B \log(A/B) + \log u_0 \leq \log m$$

or

$$A - B \log A \leq B - B \log B + \log\left(\frac{m}{u_0}\right). \quad (28)$$

Again we need that network dynamics should stop before the time when u_j reaches its maximum. Since our algorithm stops when $1 + bt = k$

$$1 + bt_{\max} = \frac{A}{B} \geq k \quad (29)$$

Equation (29) provides the lower limit of A.

We should ensure that the for a given set of parameters (k, m, u_0, θ) there always exists at least one value of A such that (28) and (29) satisfied. In the inequality in (28), the left-hand side (LHS) increases with A for a given value of B. Therefore, from (29), the minimum value of the LHS of (28) can be obtained by considering $A = kB$, i.e., $Bk - B \log(Bk) \leq B - B \log(B) + \log(m/u_0)$ i.e.,

$$B \leq \frac{\log(m/u_0)}{k - 1 - \log k}. \quad (30)$$

From (16) and (30) we get

$$\frac{\log(u_0/\theta)}{\log k} \leq \frac{\log(m/u_0)}{k - 1 - \log k}. \quad (31)$$

As mentioned before, the parameter θ is a threshold such that if the node activation decreases below θ then the corner at the corresponding pixel is eliminated. θ can be chosen to be small fraction of the initial activation u_0. Let

$$\theta = \alpha u_0 \quad (32)$$

where $\alpha < 1$. Then from (31), we can write

$$u_0 \leq m\alpha^{(k-1-\log k/\log k)}. \quad (33)$$

In other words, for a given m and $\alpha(<1)$, if the initial radius of neighborhood, k, increases then u_0 needs to be decreased. Again from (16)

$$B \geq \frac{\log \frac{u_0}{\theta}}{\log k}$$

i.e.,

$$B \geq \frac{\log(1/\alpha)}{\log k}. \quad (34)$$

Since $B = w_2 k/b$ (12), we can write

$$b \geq \frac{w_2}{\log(1/\alpha)} k \log k. \quad (35)$$

Since α is a constant (specified by the user) and w_2 can be taken as a constant, we can write $b \propto k \log k$.

For a given image first we select the initial radius of neighborhood k which is essentially driven by the requirements of the higher level recognition system. The saturation level m is fixed and it can be considered as the characteristics of the individual nodes. The parameter α is specified by the user (normally it is <1). Then first u_0 is selected from (33). After selecting u_0, B is chosen in the range given by (30) and (34). Then A is selected such that (28) and (29) are satisfied. Then from (12) and (25) we select w_2 and w_1. For example, let $m = 6.0$ and $k = 5.0$. Let the parameter α be 0.3. Then from (33) we can select $u_0 = 0.25$. From (30) and (34) we can choose $B = 0.708$. If we choose A as 5.0 then the conditions given in (28) and (29) are satisfied. Therefore from (25) and (12), $w_1 = 0.05$ and $w_2 = 0.071$ for $b = 0.5$.

IV. Corner Detection in Gray Images

In a gray image, object boundary may not be well defined as in the case of binary images, where it has been assumed that the boundary information (not necessarily a closed one) is given. In other words, in the case of a gray image, at every pixel an edge/line strength needs to be considered, which in the case of binary images is either zero or one. Also the information regarding the edge/line direction at every point is essential in finding out the initial cornerity vector, since we can no more consider at most 16 possible configurations over a 3×3 window as in the case of a binary image. The initialization of the cornerity vector at a point is performed considering the edge strengths and edge directions at the points within a neighborhood of the point concerned.

A. Edge Detection

For detecting edge points, we used the Sobel operator [22] which uses a mask as shown below

z_1	z_2	z_3
z_4		z_5
z_6	z_7	z_8

The horizontal and vertical components of the edge strength is given as

$$\begin{aligned} x_{\text{strength}} &= (z_1 + 2 * z_4 + z_6) - (z_3 + 2 * z_5 + z_8) \\ y_{\text{strength}} &= (z_1 + 2 * z_2 + z_3) - (z_6 + 2 * z_7 + z_8). \end{aligned} \quad (36)$$

The edge vector $\vec{e}$ at a point is defined as $(e_x,\, e_y)$ where e_x and e_y are the normalized x_{strength} and y_{strength}. The edge strength is given as $e = \sqrt{e_x^2 + e_y^2}$. Note that, normalization by the maximum absolute value may suppress the weaker edges. In such cases, normalization over a local window or a logarithmic scaling function [22] may be adopted.

B. Initialization of the Cornerity Vectors

The cornerity vectors are initialized as

$$\vec{C_i} = \sum_{j \in N(i)} |\vec{\boldsymbol{e}}_{\boldsymbol{j}}|\, |\sin \theta_{ij}| \hat{\boldsymbol{p}}_{\boldsymbol{ij}} \quad (37)$$

where

$\hat{\boldsymbol{p}}_{\boldsymbol{ij}}$	is the unit vector in the direction from pixel i to pixel j;
$\vec{\boldsymbol{e}}_{\boldsymbol{j}}$	is the edge strength vector;
θ_{ij}	is the angle between the edge vector $\vec{\boldsymbol{e}}_{\boldsymbol{j}}$ and $\hat{\boldsymbol{p}}_{\boldsymbol{ij}}$.

Equation (37) is based on the fact that an edge gives maximum induction along the direction perpendicular to its edge strength vector direction [20]. Unlike the case of binary images, in gray images, the neighborhood for initialization is not 3 × 3 neighborhood, instead here we used a circular neighborhood of size k. The parameter k has the same meaning as in the case of binary images.

C. Network Model

The corner detection algorithm for gray images is the same as that of the binary images, the only difference is that the contribution from the node i to the node j is multiplied by a factor $e_i e_j$, where e_i and e_j are the edge strengths at the respective pixel locations. This is because of the fact that unlike binary images, in a gray image we may not get well-defined boundaries and there will be many pixels with low edge strength (noise points) along with the actual boundary(high edge strength) points. Thus the dynamics of a node j is given by the equations

$$\frac{du_j}{dt} = \sum_{i \in N(j)} w_{ji} c_{x_i} e_i e_j - w_s c_{x_j} \quad (38)$$

$$\frac{dv_j}{dt} = \sum_{i \in N(j)} w_{ji} c_{y_i} e_i e_j - w_s c_{y_j} \quad (39)$$

and the output of the node j is $c_{x_j} = g(u_j)$, $c_{y_j} = g(v_j)$ where $g(.)$ has the same meaning as in the binary case [see (3)].

D. Convergence of the Network

The Lyapunov of the network is again same as that of the network for the binary images except for the factors e_i and e_j. The Lyapunov is chosen as

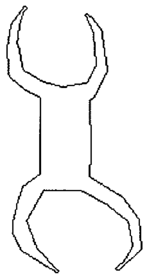
$$\begin{aligned} E = &-\frac{1}{2} \sum_i \sum_{j \in N(i)} w_{ij} e_i e_j c_{x_i} c_{x_j} + \frac{1}{2} w_s \sum_i c_{x_i}^2 \\ &-\frac{1}{2} \sum_i \sum_{j \in N(i)} w_{ij} e_i e_j c_{y_i} c_{y_j} + \frac{1}{2} w_s \sum_i c_{y_i}^2. \end{aligned} \quad (40)$$

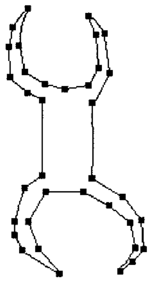
Since both e_i and e_j are nonnegative and have constant values, they can be treated as constant multiplication factor of w_{ij}. The rest of the proof is the same as that in Section II-D.

V. Experimental Results

To simulate the methods described in Sections II–IV, we use linear approximation of (1), (2), (38), and (39) with Δt as the increment in time. The parameter Δt has to be small enough in order to get good approximation of the original differential equations. To perform the experiments we used the value of Δt as 0.06 for the binary images and 0.05 for the gray images. The results for two-tone images are shown in Figs. 6–9 and those for graytone images in Figs. 10–12. The detected corner points are marked by black dots. The corners are detected for different values of the initial radius and the way of transition from coarse to finer resolution is illustrated in Figs. 6–9 and Figs. 10–12. The parameter m is a property of the neurons and it is selected as six (note that, the performance of the network is practically independent on the selection of m so long as the other parameters are changed proportionately). The parameter α is a constant specified before initializing the network. The parameter indicates the percentage decrease of the strength of a corner from its initial value that can be allowed before eliminating it. From the given values of k, m, and α the values of b, u_0, θ, w_1, and w_2 are selected (as described in the Section III) and these parameters are shown in the respective figures.

The results in Figs. 6–9 and Figs. 10–12 indicate that the performance of the network is dependent on the initial size of the neighborhood. For a small initial neighborhood, the interaction between the distant points is less, and it effectively leads to a finer description of the object boundary. On the other hand, for a large neighborhood size, the distant corner points interact with each other leading to a coarse description. This provides a flexibility to the network model in generating the descriptions of the object boundaries (not necessarily a closed one) in different levels of resolution depending on the initial selection of neighborhood size. The level of resolution is, in turn, driven by the requirement of the higher order visual information processing tasks. This effect is analogous to finding out description at different scale space. In a noisy environment, if the noisy patterns do not follow any structural form then the network behaves in a robust manner. However, if the noisy patterns themselves form structurally viable dominant points then those points are also detected as corner points. This is due to the fact that the network


(a)


(b)

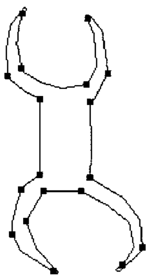
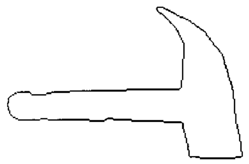
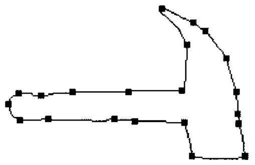

(c)

Fig. 6. (a) Original binary image. (b) Detected corner points ($k = 5, b = 1$, $u_0 = 0.25, \theta = 0.08, w_1 = 0.100, w_2 = 0.141$). (c) Detected corner points ($k = 10, b = 1, u_0 = 0.1, \theta = 0.035, w_1 = 0.025, w_2 = 0.045$).


(a)


(b)

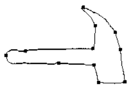
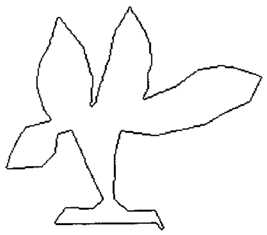
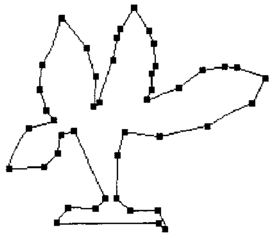

(c)

Fig. 7. (a) Original binary image. (b) Detected corner points ($k = 8, b = 1$, $u_0 = 0.1, \theta = 0.05, w_1 = 0.039, w_2 = 0.042$). (c) Detected corner points ($k = 12, b = 1, u_0 = 0.09, \theta = 0.05, iw_1 = 0.017, w_2 = 0.020$).


(a)


(b)

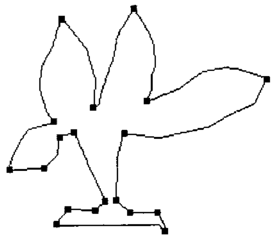
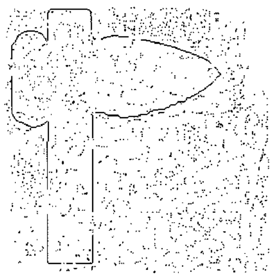
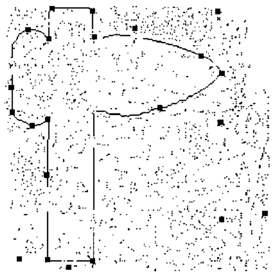

(c)

Fig. 8. (a) Original binary image. (b) Detected corner points ($k = 3, b = 0.5$, $u_0 = 0.25, \theta = 0.08, w_1 = 0.139, w_2 = 0.173$). (c) Detected corner points ($k = 5, b = 1, u_0 = 0.2, \theta = 0.05, w_1 = 0.088, w_2 = 0.172$).


(a)


(b)

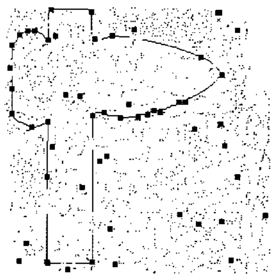

(c)

Fig. 9. (a) Original binary image. (b) Detected corner points ($k = 10, b = 0.2, u_0 = 0.1, \theta = 0.05, w_1 = 0.020, w_2 = 0.030$). (c) Detected corner points ($k = 12, b = 1, u_0 = 0.006, \theta = 0.003, w_1 = 0.012, w_2 = 0.023$).

analyzes only the local information and no explicit boundary information is available to the network. The use of only local information provides the network a capability of detecting corner points even from open or fragments of boundary segments. With

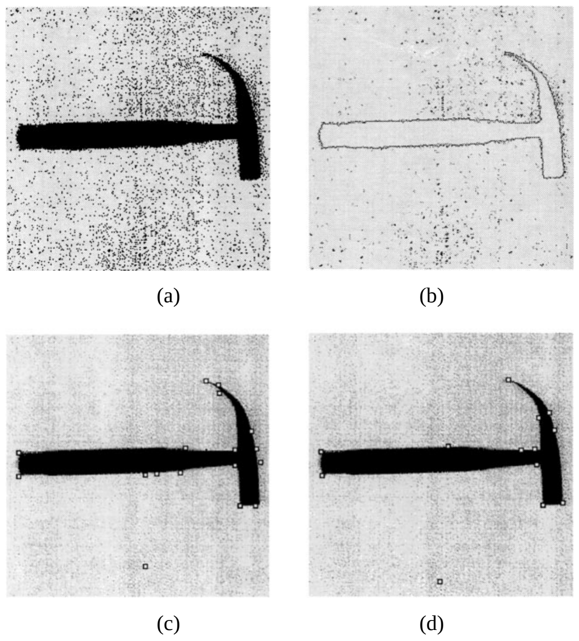


Fig. 10. (a) Original gray level image. (b) Edge Image. (c) Detected corner points ($k = 5, b = 0.2, u_0 = 0.5, \theta = 0.25, w_1 = 0.015, w_2 = 0.034$). (d) Detected corner points ($k = 9, b = 1, u_0 = 0.24, \theta = 0.1, iw_1 = 0.022, w_2 = 0.088$).

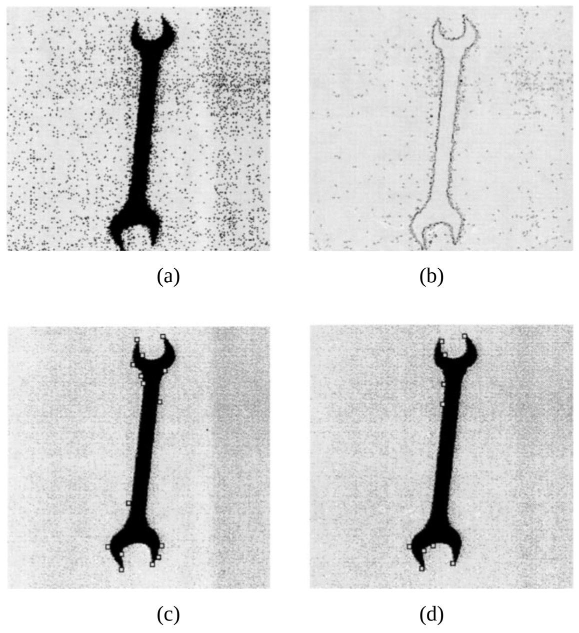


Fig. 11. (a) Original gray level image. (b) Edge image. (c) Detected corner points ($k = 5, b = 0.2, u_0 = 0.5, \theta = 0.25, w_1 = 0.015, w_2 = 0.034$). (d) Detected corner points ($k = 9, b = 1, u_0 = 0.1, \theta = 0.04, w_1 = 0.021, w_2 = 0.093$).

the extended dynamics, the network is able to detect corner points from gray images using only the local edge information.

VI. Conclusions and Scope of Future Work

A corner detection algorithm in a connectionist framework based on local cooperative computation has been developed. The convergence of the network has been proved with some restrictions on the link weights, maximum initialization value for

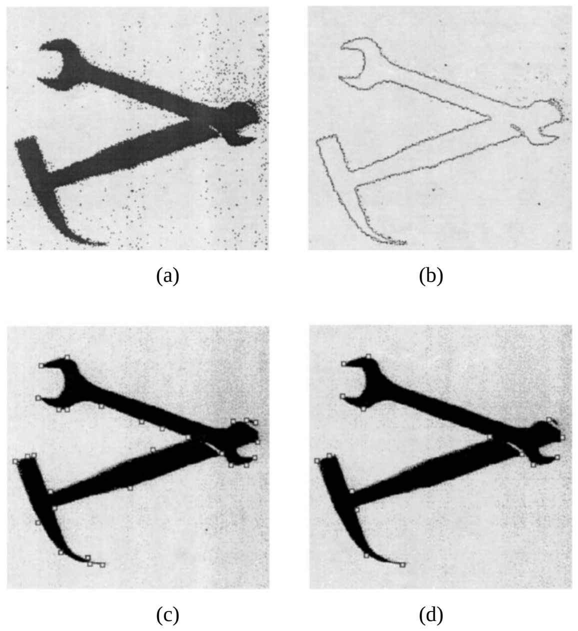


Fig. 12. (a) Original gray level image. (b) Edge image. (c) Detected corner points ($k = 5, b = 1, u_0 = 0.5, \theta = 0.25, w_1 = 0.077, w_2 = 0.034$). (d) Detected corner points ($k = 9, b = 1, u_0 = 0.24, \theta = 0.1, w_1 = 0.022, w_2 = 0.088$).

the neurons, noise threshold, and initial size of the neighborhood. These restrictions are helpful for a suitable design of the network. The performance of the proposed algorithm is dependent on the selection of initial radius of neighborhood which corresponds to the desired level of description (coarse or fine scale). The network provides a flexibility to obtain different resolution depending on the subsequent higher level visual task to be performed. It may be mentioned here that several algorithms including [10] take into account of the curvature scale space directly. Here an implicit correspondence between the curvature scale and the neighborhood size is achieved. Several graylevel corner detection algorithms are developed (see [13]) based on certain second-order differential geometric operations and certain cornerness measures are defined for mapping the image intensity surface to corner features. The proposed algorithm, on the other hand, provides a direct generalization to graylevel images from two-tone images. In graylevel images, the effective bell-shaped kernels (e.g., Gaussian kernel) for smoothing is achieved by the effect of shrinking neighborhood.

In the multilayered neural-network-based algorithm [19], only certain generic corner types are learned. In the presence of noise, the performance of this algorithm [19] deteriorates quickly. The present algorithm, on the other hand, performs gracefully in a noisy environment and in the presence of fragmented edge segments. However, unlike the existing neural-network-based algorithms [19], [18], the present network does not employ any learning procedure, rather the corner points are enhanced through cooperative computation of the neurons. In gray images, the performance of the algorithm depends on the edge detection scheme. The performance can further be enhanced with more sophisticated edge detection techniques [22]. Also the algorithm is not suitable for junction detection where more than one edge/line segment meet (e.g., in a chess board).

In this network, the effect of edge strength vectors on the cornerity vectors is taken into account, however, the effect of cornerity vectors on the edge strength vectors is not considered. The effect of corner points on the edge points and the reverse can be coupled together to obtain a better edge detection scheme along with the detection of corner points. The capability of the network can further be improved by incorporating certain learning mechanism along with the cooperative computational ability for the detection of certain different types of junctions in gray images.

Acknowledgment

The authors are thankful to the anonymous reviewers for extending helpful comments and useful pointers to other references.

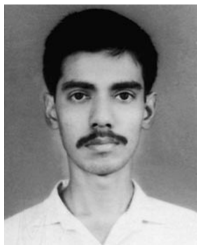
References

[1] D. H. Ballard and C. M. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[2] J. Basak and S. K. Pal, "PsyCOP: A psychologically motivated connectionist system for object perception," *IEEE Trans. Neural Networks*, vol. 6, pp. 1337–1354, 1995.
[3] A. Rosenfeld and E. Johnston, "Angle detection on digital curves," *IEEE Trans. Comput.*, vol. C-22, pp. 875–878, 1973.
[4] A. Rosenfeld and J. S. Weszka, "An improved method of angle detection on digital curves," *IEEE Trans. Comput.*, vol. C-24, pp. 940–941, 1975.
[5] H. Freeman and L. S. Davis, "A corner finding algorithm for chain coded curves," *IEEE Trans. Comput.*, vol. C-26, pp. 297–303, 1977.
[6] P. V. Sankar and C. V. Sharma, "A parallel procedure for the detection of dominant points on a digital curve," *Comput. Graphics Image Processing*, vol. 7, pp. 403–412, 1978.
[7] T. Pavlidis, "Algorithms for shape analysis and waveforms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 301–312, 1980.
[8] J. G. Dunham, "Optimum uniform piecewise linear approximation of planar curves," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 67–75, 1986.
[9] J. Wang, X. Wu, X. Huang, and X. Wang, "Corner detection using bending value," *Pattern Recognition Lett.*, vol. 16, pp. 575–583, 1995.
[10] F. Mokhtarian and R. Suomela, "Robust image corner detection through curvature scale space," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, pp. 1376–1381, 1998.
[11] K. Sohn, J. H. Kim, and W. E. Alexander, "A mean field annealing approach to robust corner detection," *IEEE Trans. Syst., Man, Cybern. B*, vol. 28, pp. 82–90, 1998.
[12] F. Arrebola, A. Bandera, P. Camacho, and F. Sandoval, "Corner detection by local histograms of contour chain code," *Electron. Lett.*, vol. 33, pp. 1769–1771, 1997.
[13] Z. Zheng, H. Wang, and E. K. Teoh, "Analysis of gray level corner detection," *Pattern Recognition Lett.*, vol. 20, pp. 149–162, 1999.
[14] T. Stammberger, M. Michaelis, M. Reiser, and K. Englmeier, "A hierarchical filter scheme for efficient corner detection," *Pattern Recognition Lett.*, vol. 19, pp. 687–700, 1998.
[15] M. Trajkovica, "Fast corner detection," *Image Vis. Comput.*, vol. 16, pp. 75–87, 1998.
[16] R. Laganiere, "A morphological operator for corner detection," *Pattern Recognition*, vol. 31, pp. 1643–1652, 1998.
[17] K. Lee and Z. Bien, "Grey-level corner detector using fuzzy logic," *Pattern Recognition Lett.*, vol. 17, pp. 939–950, 1996.
[18] D. M. Tsai, "Boundary based corner detection using neural networks," *Pattern Recognition*, vol. 30, pp. 85–97, 1997.
[19] P. G. T. Dias, A. A. Kassim, and V. Srinivasan, "A neural-network-based corner detection method," in *IEEE Int. Conf. Neural Networks*, 1995, pp. 2116–2120.
[20] J. Basak, B. Chanda, and D. D. Majumder, "On edge and line linking with connectionist model," *IEEE Trans. Syst. Man. Cybern.*, vol. 24, pp. 413–428, 1994.
[21] T. Kohonen, *Self-Organization and Associative Memory*. Berlin, Germany: Springer-Verlag, 1988.
[22] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Reading, MA: Addison-Wesley, 1993.

Jayanta Basak (M'95–SM'99) received the Bachelor's degree in electronics and telecommunication engineering from Jadavpur University, Calcutta, India, in 1987, the Master's degree in computer science and engineering from the Indian Institute of Science (IISc), Bangalore, in 1989, and the Ph.D. degree from the Indian Statistical Institute (ISI), Calcutta, in 1995.

He served as a Computer Engineer in the Knowledge —Based Computer Systems Project of ISI from 1989 to 1992. In 1992, he joined the Electronics and Communication Sciences Unit of ISI. Since 1996, he is has been an Associate Professor in the Machine Intelligence Unit of ISI. He was a Researcher in the RIKEN Brain Science Institute, Saitama, Japan from 1997 to 1998, and a Visiting Scientist in the Robotics Institute of Carnegie Mellon University, Pittsburgh, PA, from 1991 to 1992 under a UNDP fellowship. His research interests include neural networks, pattern recognition, image analysis, and fuzzy sets.

He is recipient of the gold medal from Jadavpur University in 1987, the Junior Scientist Award in computer science from Indian Science Congress Association in 1994, the Young Scientist Award in engineering sciences from Indian National Science Academy (INSA) in 1996, and the Young Investigator Award from the International Neural Network Society (INNS) in 2000.



Debashis Mahata was born in West-Bengal, India, on December 31, 1973. He received the B.Sc. degree in physics from Burdwan Raj College, Burdwan, India, in 1994 and the M.Sc. in physics with electronics specialization from Burdwan University, India, in 1996. He received the M.Tech degree in computer science from the Indian Statistical Institute, Calcutta, in 1999.

He is currently a Senior Software Engineer with Wipro Technologies, India. His interests include the application of neural networks in real life, Unix internals, theoretical physics, and quantum mechanics.