M.Tech. (Computer Science) Dissertation Report

Corner Detection in Binary and Gray Images using Neural Network

a dissertation submitted in partial fulfilment of the requirements for the M.Tech. (Computer Science) degree of the Indian Statistical Institute

By

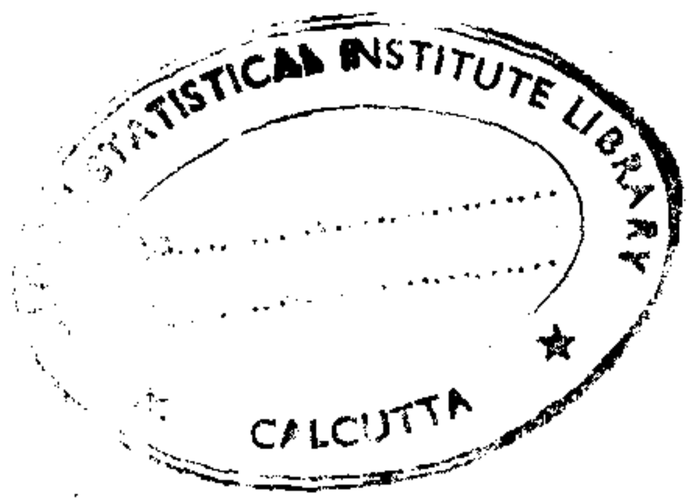
DEBASHIS MAHATA

Under the Supervision of

Dr. Jayanta Basak

Machine Intellegence Unit
Indian Statistical Institute
203, Barrackpore Trunk Road
Calcutta - 700035

JULY, 1999



Acknowledgements

At first, I would like to express my profound gratitute to my superviser Dr. Jayanta Basak who guided me throughout my dissertation and helped me a lot whenever approached for. I am greateful to Mr. Snigdhanshu Bhusan Chatterjee, who helped me to solve diferent mathematical problems. I would like to thank Mr. Arijit Bishnu for his kind co-operation. I would also like to thank all our co-hostellers and classmates specilly Debrup, Shubhodeep, Saurav, Anil, and Analavha to help me a lot.

Debashis Mahata

Contents

Abstract

In this dissertation, neural network based methodologies are developed for the detection of corner points in both 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 repensenting the direction and strength of the corner. These corneritis are then mapped onto a neural network model which is essentially designed as a cooperative computational framework. A pair of neurons in the network model corresponds to a pixel in the image. The cornerity at each pixel position (i.e., at each pair of neurons) is updated according to the corneritis at the surrounding locations (i.e., the neighborhood information). The actual corner points are obtained after the network dynamics settles to stable state. 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.

Chapter 1

Introduction

1.1 Introduction

Detection of suitable feature points in images is an important problems in object recognition, scene analysis, stereo matching, and many other image processing, and computer vision tasks [18], [16]. 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 and stereo matching. A corner is theoretically defined as the point of discontinuity in the curvature on an arc. In digital images, the points with high curvature values are considered to be 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 [9], Rosenfeld and Weszka [10], Freeman and Davis [5] and many others [11, 1].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 [8, 4], and the points of intersections of the adjacent linear segments are detected as corner points. In another approach by Wang et al. [13], 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 cornerity at the correspond-

ing point. In this method, direct computation of angle between adjacent segments is replaced by the estimations of bending value which involves only addition and subtraction operations, leading to faster computation. These algorithms have several disadvantages. First, they accept closed object boundaries only. However, it may not always be possible to find out the closed object boundaries of objects in a gray level image by edge/line detection algorithms. Secondly, the algorithms developed for binary images are not extendible to gray images. This means that the computational framework is not uniform for both binary and gray images.

Neural networks [19, 17], composed of simple processing elements (neurons/nodes) with local computation, provides an adaptive computational framework with fast and parallel computational ability, exhibiting graceful degradation performance under noisy environment. Neural networks has also used to detect the corner points [12], where a multilayered perceptron (MLP) network is trained for this purpose. The MLP based approach [12] also has the same disadvantages as the other algorithms mentioned before.

Here we developed a method for detecting the corner points in binary and gray images using neural network. In the network model, a pair of nodes corrosponds to a pixel in the image. The output of the pair of nodes represent the corner vector of the corrosponding pixel. Each pair of neurons is laterally connected over a neighborhood, and the node activations are updated by the neighborhood information. Initial conerity information of each pixel is assigned to each 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 bounderies also. The network exhibit robust performence against the presence of noise. The dynamics of the network is then extended to accept the ~~cornerity~~ edge information from gray images by embeding the edge strength information. A graceful performance by the network is observed for the gray images.

1.2 Organization of the Report

The rest of the dissertation is organized as follows. Chapter 2 elaborates the proposed method for detecting the corner points in binary images. The basic ideas behind the algorithm is given in section 2.1. The network model is described in section 2.3. The initialization procedure for the network is clarified with example an in section 2.1. The convergence of the network is given in section 2.4. The convergence of the network requires certain restrictions on the choice of the network parameters and thay are analytically derived in section 2.5.

In chapter 3 we mainly concentrat on the extension of the network used for the binary images in order to accept gray level images. For gray level images the network accepts the edge/line information. The edge detection scheme is described in section 3.1. The initialization of the network for gray images uses the output of the edge detection algorithm. The initialization procedure is described in section 3.2. The network model is then described in section 3.3 along with the convergence of the network in section 3.4.

Chapter 4 demonstrats the effectiveness of the neural network model for both binary and gray images. The experimental simulation of the network is described in section 4.1. The experimental results obtained for two tone images and gray images are given in section 4.2 and 4.3 respectively.

Finally chapter 5 provides the overall conclussions and the scope of future work.

Chapter 2

Corner Detection in Binary Images

2.1 Overall Methodology

Here we assumed that the given image consists of only the boundary information, the boundary not necessarily being a closed one. The required input image may be obtained after the boundary detection of a segmented image, or even after edge/line detection directly from the gray image [3]. However, presently we do not consider the edge/line strength and the edge/line direction associated with each pixel. The input image (e.g. Figure 4.1) consists of black and white pixels, black pixels representing edge/line boundary points. The cornererity (which is a vector representing the direction and strength of each corner) at every point on the digital arc is initialized considering only a small neighborhood (here in this case it is a 3×3 neighborhood). The way we initialized the corner vectors is described in section 2.2. Note that, in discrete domain, in a 3×3 neighborhood only sixteen different types of patterns [2] are possible.

Here 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 updating process is such that the cornerity vectors are enhanced at the true corner points and get suppressed at the other points. The corner points can then be identified by finding the local maxima in the magnitude of cornerity vector in a local neighborhood. The enhancement or suppression of the cornerity of a point during the updating process, depends on the cornerity direction at the point of interest and that of the neighboring points. The way of suppression or enhancement of the of the cornerity vectors is performed, is exemplified in Figure 2.1 and Figure 2.2 respectively.

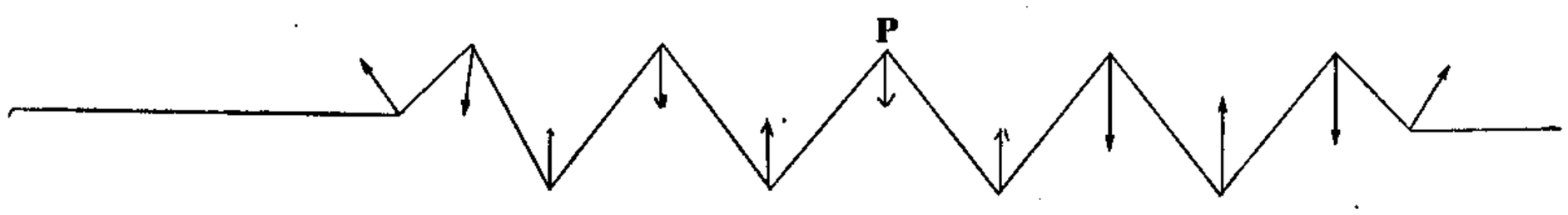


Figure 2.1: Suppression of corners in a juggled edge.

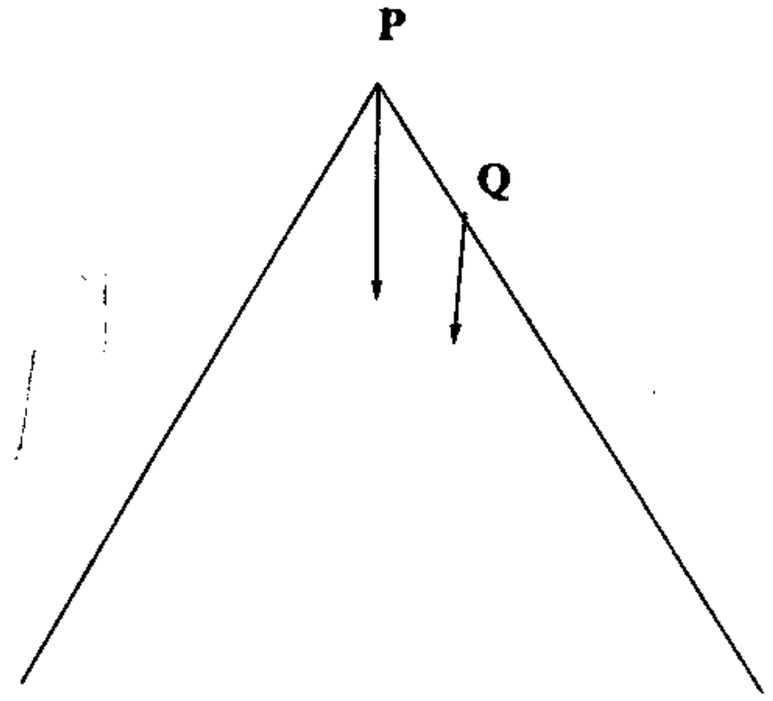


Figure 2.2: Finer detail of the juggled edge.

Let us have an edge segment (line boundary) as shown in Figure 2.1 and the corner is to be detected at P. The initial corner vectors at each point are shown in the Figure 2.1. 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 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 cornerity vector at each point is updated by the neighborhood cornerity information, i.e., the vector sum of the cornerities of the neighbouring points. The size of the neighborhood has an important role in determining whether the description of the edge segment is a coarse one or a finer one. As in Figure 2.1, 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 neighbourhood size no oppositely directed corners in the neighborhood affect each other. If we consider a small neighborhood, the segment around P can be represented as shown in Figure 2.2. In Figure 2.2 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 the true corners points (i.e., a corner point without any neighboring oppositely directed corners) are enhanced.

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 pair of neurons corresponding to a pixel are of different type. The activation of the pair of neurons together, represinting the cornerity vector at the corresponding pixel. Each neuron, in the pair, individually represints a component of the cornerity vector (Note that, here a cornerity vector is represented by two components, horizental and vertical respectively.). 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. The negative self-feedback helps to eleminate noise. The initial cornerities are assigned considering the boundary points in a 3×3 neighborhood. The input to the model is, therefore the initial corner information of very small digital arcs (over only 3×3 neighborhood). The output of each node is the modified cor-

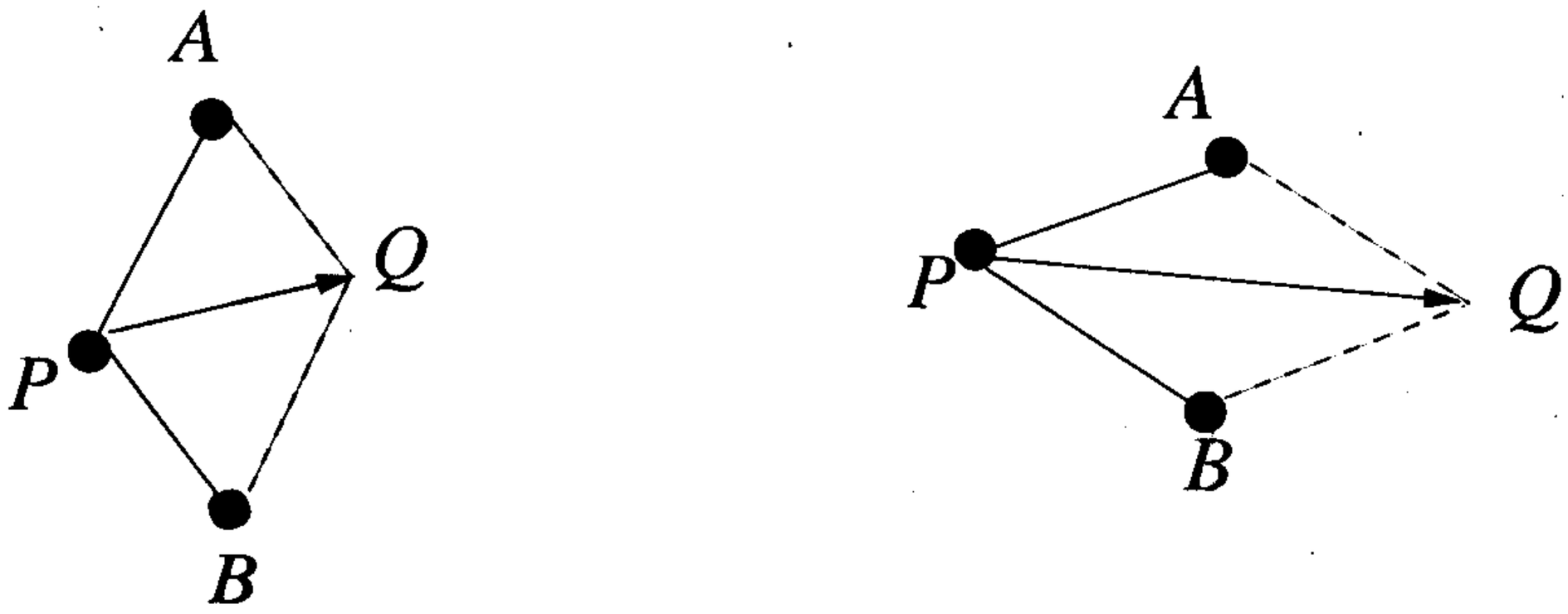


Figure 2.3: Two typical cases of initializing the cornerity vector

ner vector at the corresponding point for a certain detail of description. Initially, the state of the neurons are clamped by the input (corner vectors) from 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.

2.2 Initialization of Corner Vectors

Initialization of the neurons in the networks is very important in the sense that the neurons in the network updates there states starting from there initial cornerity vectors only. The initial corner vectors represent the cornerity corresponding to a edge/line segment in a 3×3 neighborhood. These initialized vectors should reflect the appropriate direction and the magnitude of the corner vectors for small edge/line segments. For example, the point which lie on a straight line segment should have zero initial vector. On the other hand a boundary point with its two arms separated by a small angle should get high initial cornerity. Considering a 3×3 neighborhood the center pixel can have at most two neighboring points on a

single pixel thick boundary. 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 Figure 2.3 which illustrates the two typical situations. In Figure 2.3 black dlobs 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 Figure 2.3 than that in the case shown in the left side of Figure 2.3. This implies that the initial cornerity vector is larger in the case shown in the right side of Figure 2.3, than the other. In 2.3 $\overrightarrow{PQ}$ 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, horizental and vertical respectively. Mathematically let (i, j) be a boundary pixel in an object and $(i + l_1, j + k_1)$ and $(i + l_2, j + k_2)$ be 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 horizental 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. The cornerity vectors of non boundary points are assigned to zero.

2.3 Network Model

As mentioned in section 2.1, there are two neurons of different type for each pixel. Each neuron is connected with the other neurons of the same type over a neighboorhood. The lattice structure of the nodes of a given type is shown in Figure 2.5. The other type of nodes also have the similar structure. Since the interaction is restricted only between the neurons of the same type, we will use only pixel index

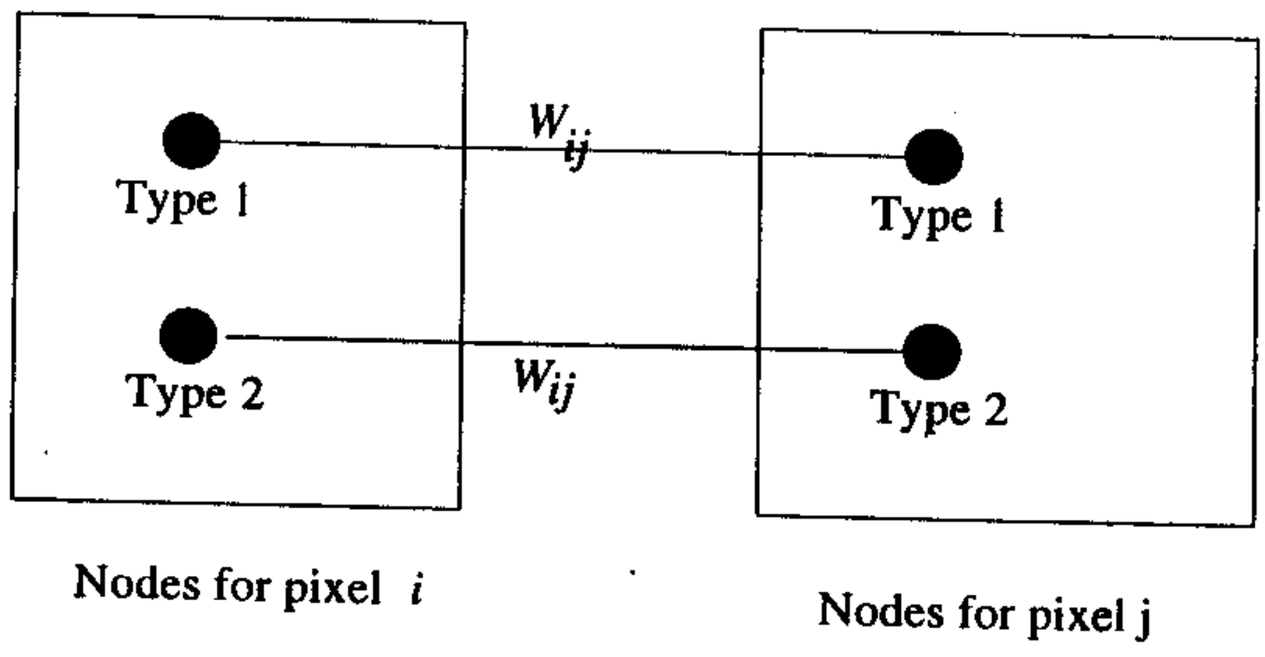


Figure 2.4: Interaction between the nodes of two different pixels

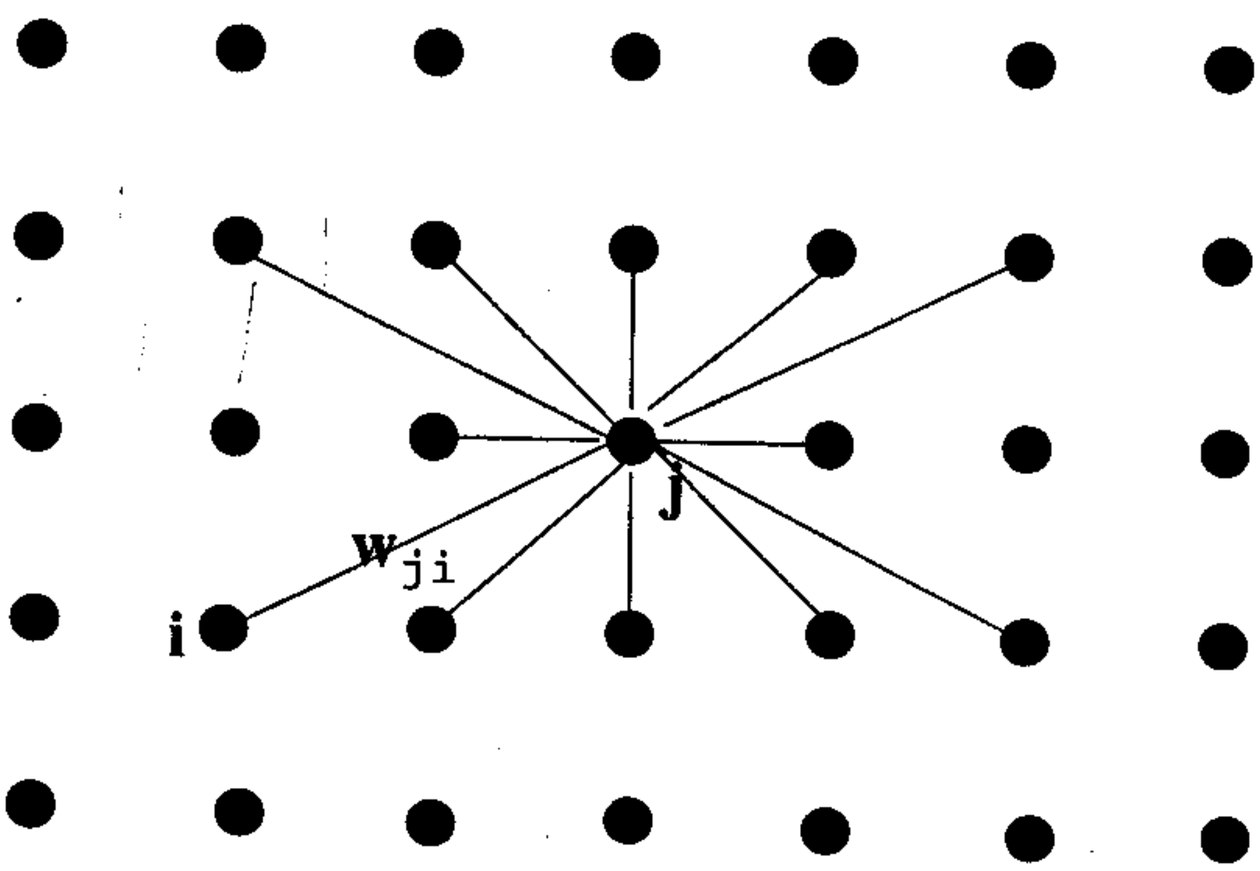


Figure 2.5: Lattice structure of a given type of nodes.

to represent a neuron, the appropriate type being implicit. The connection between two pairs of neurons corresponding to two different pixels is shown in Figure 2.4. 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 horizental and vertical components respectively of the corner vector at the j^{th} pixel. Node j is connected to its neighborhood nodes (denoted by index i) of same type with weight w_{ji} if node i is within a given neighborhod $N(j)$ of j. The interconnections weights are symmetric in nature, i.e., $w_{ji} = w_{ij}$. The state dynamics of the processing element j is given by

$$\frac{du_j}{dt} = \sum_{i \in N(j)} \left(w_{ji} c_{x_i} \right) - w_s c_{x_j} \tag{2.1}$$

or

$$\frac{dv_j}{dt} = \sum_{i \in N(j)} w_{ji} c_{y_i} - w_s c_{y_j} \tag{2.2}$$

The output of the node j is given by

$$c_{x_j} = g(u_j) \tag{2.3}$$

or

$$c_{y_j} = g(v_j) \tag{2.4}$$

respectively. $g(.)$ is a ramp function (Figure 2.6) given by

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

where m is the saturation level of the ramp function. The second term in (2.1) and (2.2) are negative feedback terms used to eliminate the noise points. w_s is the weight of the negative feedback. $N(j)$ is a neighborhood of j^{th} neuron in which the outputs of the neurons affect each other.

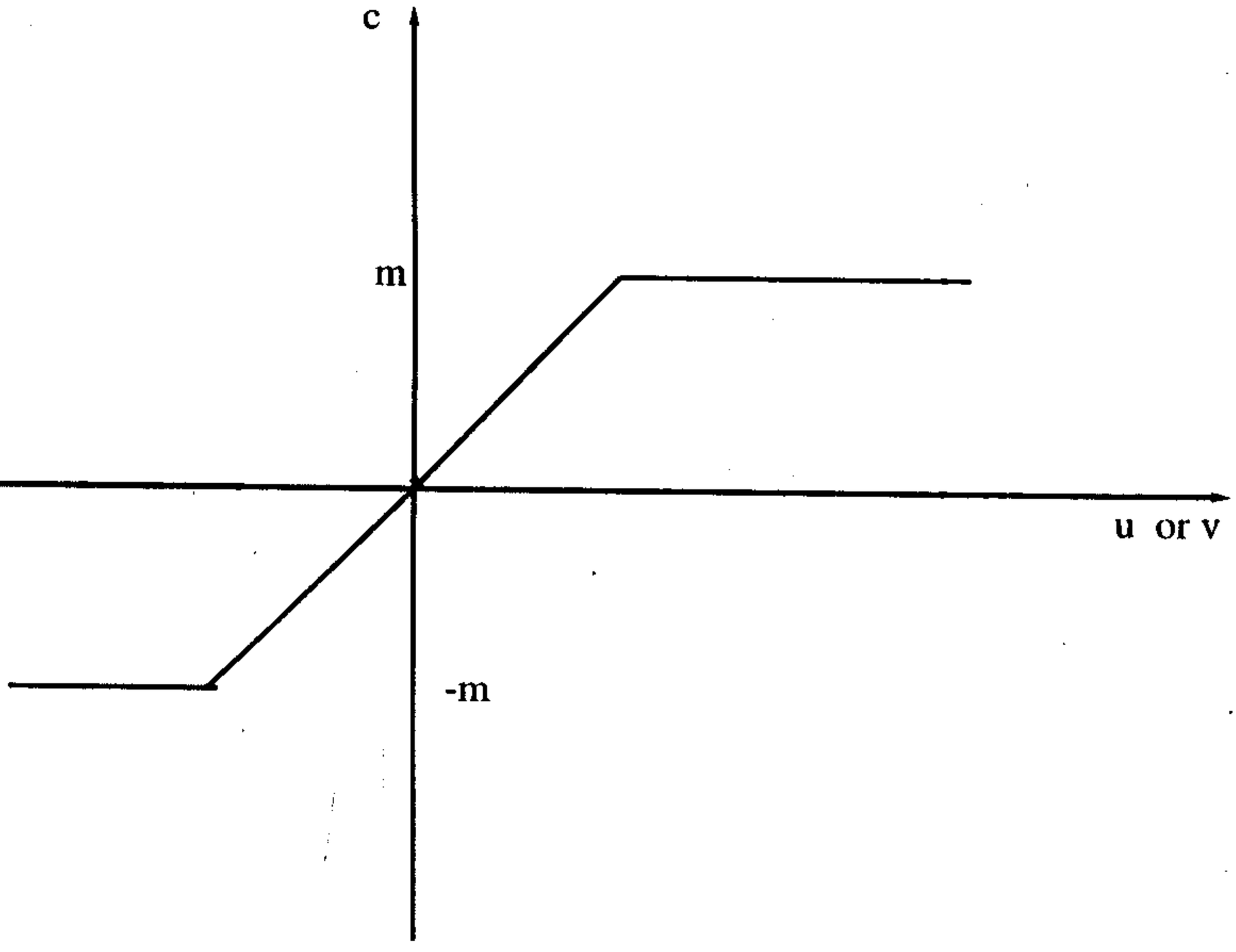


Figure 2.6: The ramp function at the output of each node

The neighborhood chosen here is a circular one. The radius r of the neighboorhood decreased with time, i.e.,

$$\text{for} \quad t_1 > t_2 > t_3 ... > t_n$$

$$r(t_1) < r(t_2) < r(t_3) ... r(t_n).$$

The shrinking of the neighboorhood size, in effect, results 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 [15]

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

$$\sum_{t} r(t) \to \infty.$$

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

$$r = \frac{k}{1 + bt} \tag{2.5}$$

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.,

$$w_{ji} \propto r \tag{2.6}$$

$$w_{ji} \propto r \tag{2.7}$$

2.4 Convergence of the Network

The size of the neighborhood of every processing element decreases with time. As the radius of the neighborhood reduces to zero, the process of updating the states

of the processing elements stops. 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. If the network converges for a neighborhood of fixed radius, then evidently the network must converges for shrinking neighborhood also. The proof of convergences for a fixed radius of neighborhood is as follows.

Consider a Lyapunov (or the energy function) of the network as

$$E = -\frac{1}{2}\sum_i \sum_{j \in N(i)} w_{ij} c_{x_i} c_{x_j} + \frac{1}{2} w_s \sum_i c_{x_i}^2 - \frac{1}{2}\sum_i \sum_{j \in N(i)} w_{ij} c_{y_i} c_{y_j} + \frac{1}{2} w_s \sum_i c_{y_i}^2. \quad (2.8)$$

Therefore $\frac{dE}{dt}$ is given by

$$\frac{dE}{dt} = -\sum_i \left(\sum_{j \in N(i)} w_{ij} c_{x_j} - w_s c_{x_i} \right) \frac{dc_{x_i}}{dt} - \sum_i \left(\sum_{j \in N(i)} w_{ij} c_{y_j} - w_s c_{y_i} \right) \frac{dc_{y_i}}{dt}. \quad (2.9)$$

From 2.1 and 2.2 we get

$$\frac{dE}{dt} = -\sum_i \frac{du_i}{dt}\frac{dc_{x_i}}{dt} - \sum_i \frac{dv_i}{dt}\frac{dc_{y_i}}{dt} \quad (2.10)$$

From 2.3 and 2.4,

$$\frac{dE}{dt} = -\sum_i g^{-1'}(u_{x_i}) \left(\frac{dc_{x_i}}{dt}\right)^2 - \sum_i g^{-1'}(u_{y_i}) \left(\frac{dc_{y_i}}{dt}\right)^2 \quad (2.11)$$

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 2.4). Since, in the given range g^{-1} is an increasing function, i.e., $g^{-1'}$ is positive, and $(\frac{dc_{x_i}}{dt})^2$ and $(\frac{dc_{y_i}}{dt})^2$ are always nonnegative,

$$\frac{dE}{dt} \leq 0, \quad \forall t \geq 0. \quad (2.12)$$

$E(t)$ is bounded, $\frac{dE}{dt} \to 0$ as $t \to \infty$ and therefore $\frac{dc_{xi}}{dt} \to 0$ and $\frac{dc_{yi}}{dt} \to 0$ as $t \to \infty$ for all i. As a result, the output values of all processing elements converge.

2.5 Selection of Network Parameters

To find the bounds value of the link weights and the parameter values we consider the state dynamics of a node as given by (2.1) and (2.2). The analysis in this section, is done considering only c_x 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 is 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 eleminated, and m is the saturation level of the ramp function.

Theoretically, a noise point is one which initially have some cornerity but dose not get support from the neighborhood. Then the differential equation satisfied by a noise point (from (?? is

$$\frac{du_j}{dt} = -w_s c_j \tag{2.13}$$

Assuming that the value of u_j is within the saturation limit of the ramp function (Figure 2.6), we replace c_j in (2.13) by u_j.Therefore from (2.7) and (2.13),we get

$$\frac{du_j}{dt} = -w_2 r u_j \tag{2.14}$$

It is required that after convergence of the network,the cornerity at the noise points should less than θ so that thay can be eleminated. Let us consider a noise point with maximum initialization u_0. Now from (2.5) and (2.14),

$$u_j(t) = \frac{u_0}{(1+bt)^B} \tag{2.15}$$

where B is a constant given by

$$B = (w_2 k)/b. \tag{2.16}$$

Since we stops when $r < 1$ (let at time instant $t = t_m$), the value of $u_j(t)$ in (2.15) should satisfy the inequality

$$u_j(t_m) \leq \theta. \tag{2.17}$$

Again, from (2.5) $r = 1$ implies that

$$1 + bt_m = k. \tag{2.18}$$

From (2.15), (2.17), and (2.18),

$$\frac{u_0}{k^B} \leq \theta \tag{2.19}$$

Therefore,

$$B \geq \log_k \frac{u_0}{\theta} \tag{2.20}$$

This equation gives a lower bound on B, i.e., weight of the self-feedback (from (2.16).

In order to get a bound in the value of the weight of the links between the nodes let us consider the updating rule (2.1) once again which is given as

$$\frac{du_j}{dt} = \sum_{i \in N(j)} w_{ji} u_i - w_s u_j$$

Replacing w_{ji} by $w_1 r$ where w_1 is a constant (2.7), and u_j by c_j assuming u_j does not exceed the limit of the ramp function, we get

$$\frac{du_j}{dt} = \sum_{i \in N(j)} w_1 r u_i - w_2 r u_j \tag{2.21}$$

In the updating process, in order to find out the maximum level of activation that can be attained by an individual neuron, let us assume that each neuron has got a maximum initial activation u_0. Again, according to (2.21), the network behavior is isotropic, i.e., the activation of all nodes grow symmetrically. This means that

the 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 the node j can be written as

$$\frac{du_j}{dt} = G_j(t) \tag{2.22}$$

where

$$G_j(t) = \sum_{i \in N(j)} w_1 r u_i - w_2 r u_j. \tag{2.23}$$

From (2.22) we get

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

or,

$$\log u_j - \log u_0 = \int_0^t \frac{G_j}{u_j} dt. \tag{2.24}$$

Now using the expression of G_j from (2.23),

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

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

$$\frac{G_j}{u_j} = \sum_{i \in N(j)} (w_1 r) - w_2 r. \tag{2.25}$$

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,

$$\frac{G_j}{u_j} = p w_1 r^2 - w_2 r. \tag{2.26}$$

On an avagage, we can assume that p is equal to 2.

From (2.24), (2.5), and (2.26), we get,

$$\log u_j(t) = \int_0^t \left(\frac{2 w_1 k^2}{(1+bt)^2} - \frac{w_2 k}{1+bt} \right) dt + log^{u_0}. \tag{2.27}$$

i.e.,

$$\log u_j(t) = A \left(1 - \frac{1}{1+bt}\right) - B \log(1+bt) + \log u_0 \tag{2.28}$$

where,

$$A = (2w_1k^2)/b. \tag{2.29}$$

From equation (2.28) 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 $(du_j/dt) = 0$,

$$1 + bt_{max} = \frac{A}{B} \tag{2.30}$$

The maximum value of u_j can be found out from (2.28) and (2.30), which is

$$(u_j)_{1+bt=(A/B)} = \exp\left(A\left(1 - \frac{B}{A}\right) - B\log\frac{B}{A} + \log u_0\right) \tag{2.31}$$

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,

$$A - B\log A \leq B - B\log B + log(\frac{m}{u_0}) \tag{2.32}$$

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

$$1 + bt_{max} = \frac{A}{B} \geq k \tag{2.33}$$

The (2.33) gives 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 (2.32) and (2.33) satisfied. In the inequality in (2.32), the left hand side increases with A for a given value of B. Therefore, from (2.33), the minimum value of the left hand side of (2.32) can be obtained by considering $A = kB$, i.e.,

$$Bk - Blog(Bk) \leq B - Blog(B) + log(m/u_0)$$

i.e.,

$$B \leq \frac{log(m/u_0)}{k - 1 - logk}. \tag{2.34}$$

From (2.20) and (2.34) we get,

$$\frac{\log(u_0/\theta)}{\log k} \leq \frac{log(m/u_0)}{k - 1 - \log k} \tag{2.35}$$

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 eleminatted. θ can be choosen to be small fraction of the initial activation u_0. Let

$$\theta = \alpha u_0 \tag{2.36}$$

where $\alpha < 1$. Then from (2.35), we can write

$$u_0 \leq m\alpha^{\left(\frac{k-1-\log k}{\log k}\right)}. \tag{2.37}$$

In other words, for a given m and α(¡ 1), if the initial radius of neighboorhood, k, increases then u_0 needs to be decreased.

Again from (2.20),

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

Since $B = w_2 k/b$ (Equation (2.16)), α is a constant and if we consider w_2 to be a constant then

$$b \propto k \log k. \tag{2.38}$$

The conditions for selecting the parameters are given below. For a given 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 (2.37). After selecting u_0, B is chosen in the range

$$\frac{\log(u_0/\theta)}{\log k} \leq B \leq \frac{\log(m/u_0)}{k - 1 - \log k} \tag{2.39}$$

Then A is selected such that

$$\frac{A}{B} \geq k \tag{2.40}$$

and

$$A - BlogA \leq B - BlogB + log(\frac{m}{u_0}) \tag{2.41}$$

satisfed.

From (2.16) and (2.29) we select w_2 and w_1. The way of selecting, w_1 and w_2, and u_0 is examplified below. Example: Let $m = 6.0$ and $k = 5.0$. Let the parameter α be 0.3. Then from (2.37) we can select $u_o = 0.25$. Therefore from (2.39) we can choose $B = 0.708$. If we choose A as 5.0 then the conditions given in (2.39), (2.40), and (2.41) are satisfied. Therefore from (2.29) ancd (2.16), $w_1 = 0.050$ and $w_2 = 0.071$ for $b = 0.5$.

2.6 Conclusions

In this chapter we developed a neural network framework to detect the corner points in binary images. The initialization of each node in the network has been done using the information about the spatial distribution of the object boundary pixels. The network used here have the structure of two dimensional lattice (Figure 2.5). Each node in this network gets information from its neighborhood and updates its state. Each node has a ramp transfer function. In order to satisfy the convergence of the network, the output of the nodes should be within the limits of saturation level of the ramp function. To satisfy this restriction the conditions for selecting the

network parameters are also derived which in turn helps in designing the network. The effectiveness of this network in finding the corner points is demonstrated in chapter 4.

Chapter 3

Corner Detection in Gray Images

In the present chapter we consider the case of gray images. In a gray image the image boundary may not be well defined as in the case of binary images. Therefore, the initialization of the cornerity vectors at each point is not straight forward. This problem can be taken care of by finding the edge strength and the direction of edge at each point of the gray image. Then the initialization of the corner vectors can be performed according to the edge strength of the points in a neighborhood of the point in concerned.

3.1 Edge Detection

For detecting edge points, we used the Sobel operator [6]which uses a mask as shown below

z_1	z_2	z_3
z_4		z_5
z_6	z_7	z_8

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

$$x_{strength} = (z_1 + 2 * z_4 + z_6) - (z_3 + 2 * z_5 + z_8) \tag{3.1}$$

and

$$y_{strength} = (z_1 + 2 * z_2 + z_3) - (z_6 + 2 * z_7 + z_8) \tag{3.2}$$

To limit the these values within unity these edgeg strengths are normalized.

3.2 Initialization of the Corner Values

The cornerity vectors are initialized as

$$\vec{C_i} = \sum_{j \in N(i)} |\vec{\mathbf{e_j}}||sin\theta_{ij}|\hat{\mathbf{p}}_{\mathbf{ij}} \tag{3.3}$$

where $\hat{\mathbf{p}}_{\mathbf{ij}}$ is the unit vector in the direction from pixel i to pixel j, $\vec{\mathbf{e}}_{\mathbf{j}}$ is the edge strength vector, perpendicular to the edge at pixel position j, and θ_{ij} is the angle between the edge vector $\vec{\mathbf{e}}_{\mathbf{j}}$ and $\hat{\mathbf{p}}_{\mathbf{ij}}$. The equation (3.3) based on the fact that an edge gives maximum induction along the direction perpendicular to its edge strength vector direction [2]. 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 parameters k has same meaning as in the binary case (chapter 2)

3.3 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 locatins. This is because of the fact that unlike binary images, in a gray image we can not get well defined boundary 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

$$\frac{du_j}{dt} = \sum_{i \in N(j)} w_{ji} c_{x_i} e_i e_j - w_s c_{x_j} \tag{3.4}$$

or

$$\frac{dv_j}{dt} = \sum_{i \in N(j)} w_{ji} c_{y_i} e_i e_j - w_s c_{y_j} \tag{3.5}$$

and the output of the node j is

$$c_{x_j} = g(u_j)$$

or

$$c_{y_j} = g(v_j)$$

where $g(.)$ have the same meaning as in binary case (Equation (2.3) and (2.4)).

3.4 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 choosen as

$$E = -\frac{1}{2}\sum_i \sum_{j \in N(i)} w_{ij} e_i e_j c_{x_i} c_{x_j} + \frac{1}{2} w_s \sum_i c_{x_i}^2 - \frac{1}{2}\sum_i \sum_{j \in N(i)} w_{ij} e_i e_j c_{y_i} c_{y_j} + \frac{1}{2} w_s \sum_i c_{y_i}^2 \tag{3.6}$$

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 2.4.

3.5 Conclusions

In this chapter we concentrated on the extension of the network for binary images to accept gray images. For this we initialized the cornerity vector at each point

in the image considering the edge strength vectors of the neighboring pixels. The performance of the network depends on the performance of the edge detection and the initialization procedure. The bounds on the parameters is same as that in the case of binary images. The effectiveness of this network in identifying the corner points is demonstrated in chapter 4.

Chapter 4

Experimental Results

Here first we provide outline of the experimental simulation of the networks. Then the experimental results for binary and gray images are provided in section 4.2 and 4.3 respectively.

4.1 Simulation of the Network Model

To simulate the methods developed in chapter 2 and chapter 3, we use linear approximation of the equations (2.1), (2.2), (3.4), and (3.5). The approximated updating rules for the binary images are

$$u_i(t+\triangle t) \leftarrow u_i(t) + \left(\sum_{i \in N(j)} w_{ji} c_{x_i}(t) - w_s c_{x_j}(t) \right) \triangle t \tag{4.1}$$

$$v_i(t+\triangle t) \leftarrow v_i(t) + \left(\sum_{i \in N(j)} w_{ji} c_{y_i}(t) - w_s c_{x_j}(t) \right) \triangle t \tag{4.2}$$

and the same for the gray images are

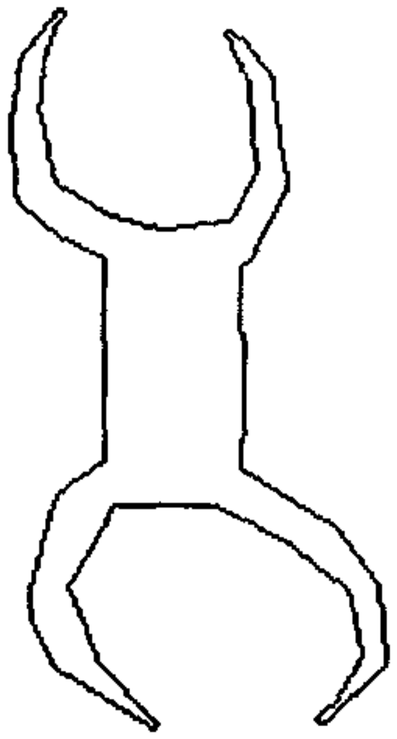
$$u_i(t+\triangle t) \leftarrow u_i(t) + \left(\sum_{i \in N(j)} w_{ji} c_{x_i}(t) e_i e_j - w_s c_{x_j}(t) \right) \triangle t \tag{4.3}$$

$$v_i(t+\triangle t) \leftarrow v_i(t) + \left(\sum_{i \in N(j)} w_{ji} c_{y_i}(t) e_i e_j - w_s c_{x_j}(t) \right) \triangle t \qquad (4.4)$$

respectively. $\triangle t$ is a small parameter denoting the increment in time. The parameter $\triangle t$ has to be small enough in order to get good approximation of the original differential equations. To perform the experiments we used the the value of $\triangle t$ as 0.06 for the binary images and 0.05 for the gray images. The other parameter values are given along with the corresponding figures in the following two sections.

4.2 Results for Two-tone Images

The results for two-tone images are shown in Figures 4.1-4.5. The detected corner points are marked by black dots. The parameters $(k, b, u_0, \theta w_1, w_2)$ are given along with each results.

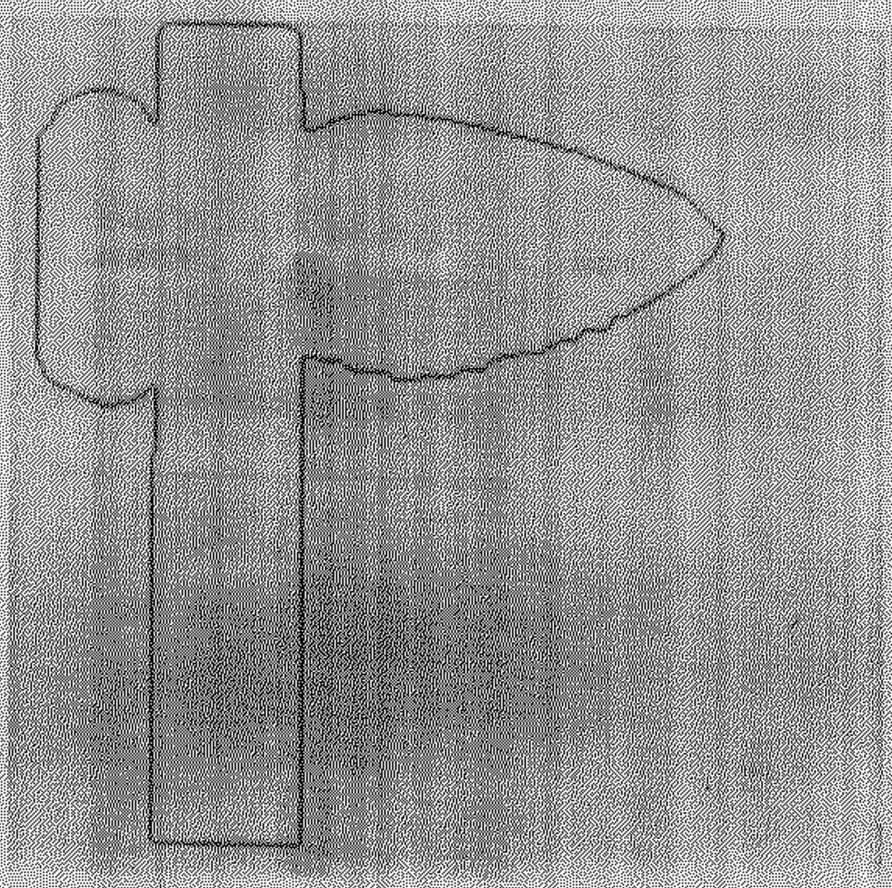


(a)

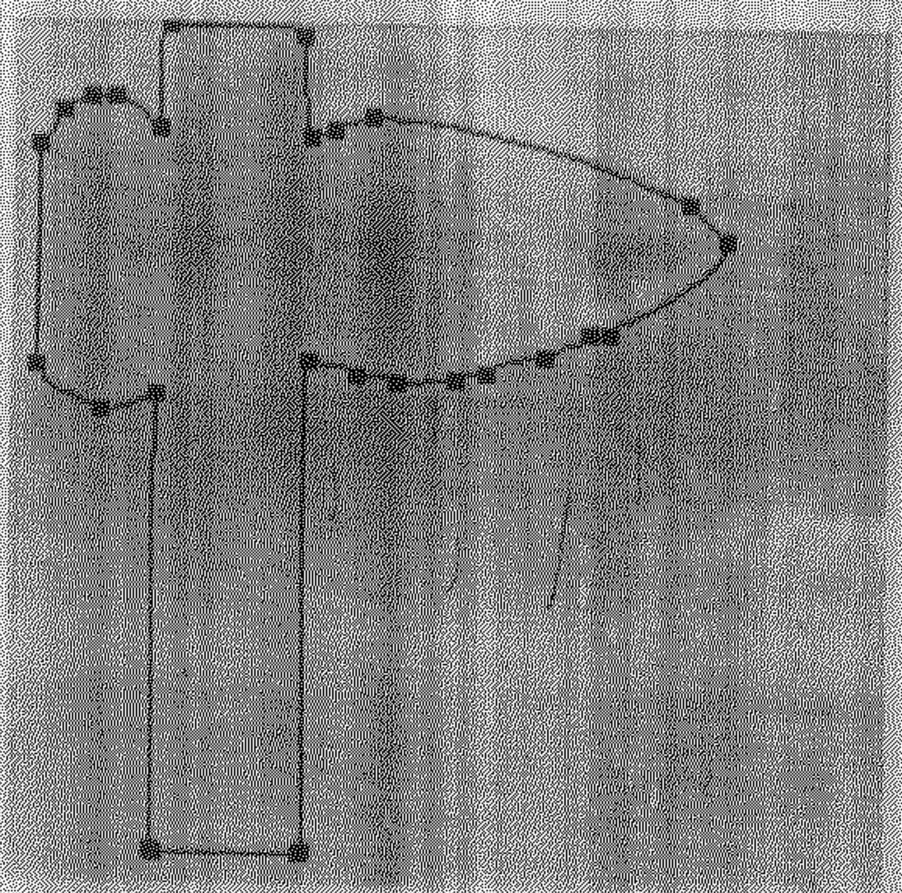
(b)

(c)

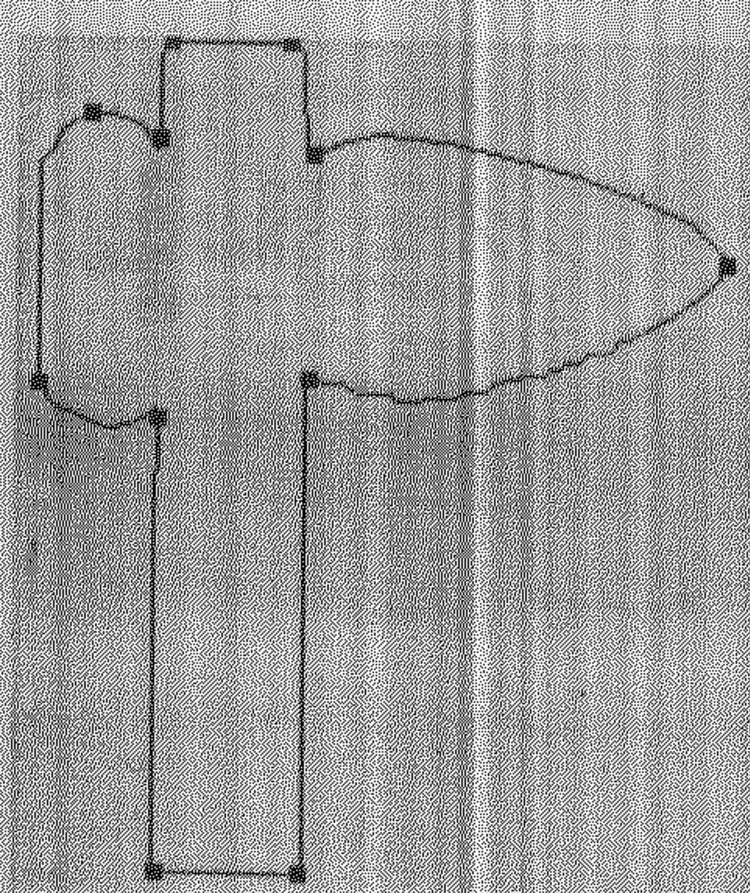
Figure 4.1: (a) Original binary image. (b)Detected corner points (k=5, b=1, $u_0 = 0.25, \theta = 0.08$, $w_1 = 0.100, w_2 = 0.141$) (c) Detected corner points(k=10, b=1, $u_0 = 0.1, \theta = 0.035$, $w_1 = 0.025, w_2 = 0.045$)



(a)

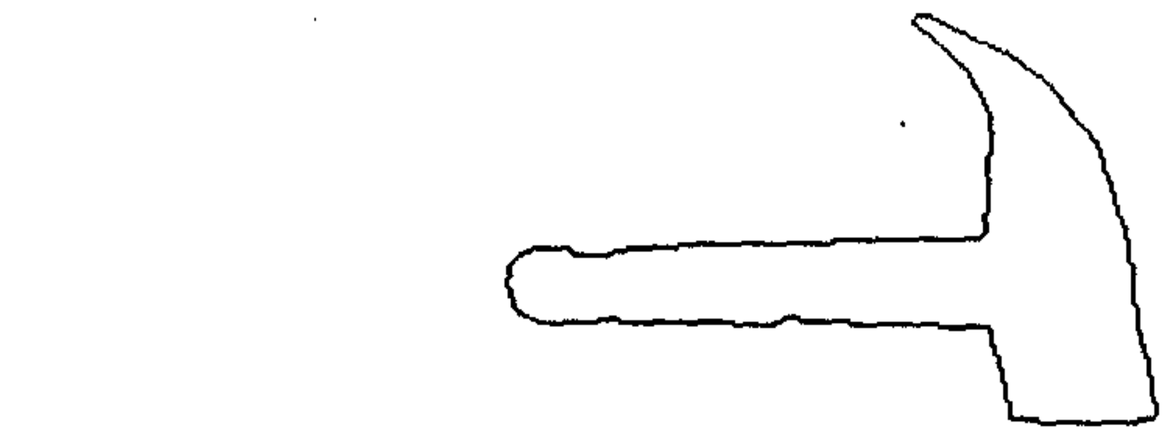


(b)

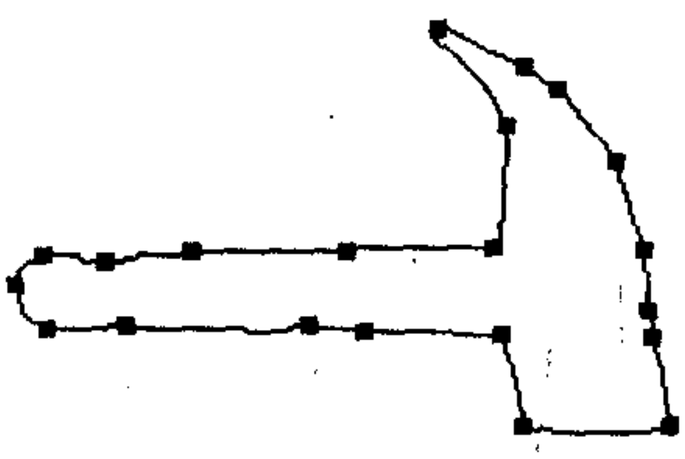


(c)

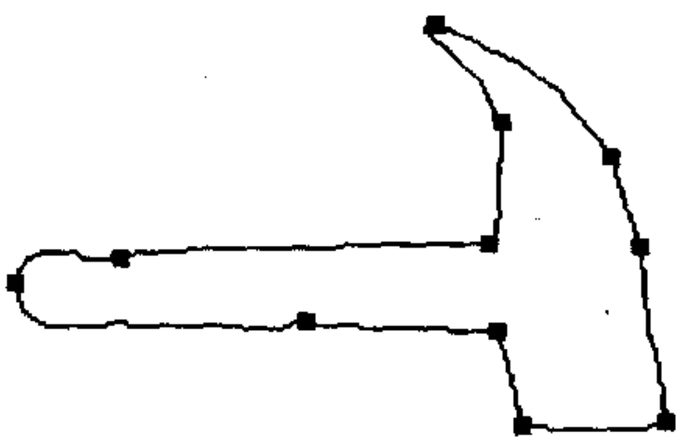
Figure 4.2: (a) Original binary image. (b)Detected corner points (k=5, b=1, $u_0 = 0.25, \theta = 0.08$, $w_1 = 0.080, w_2 = 0.141$) (c) Detected corner points(k=12, b=1, $u_0 = 0.1, \theta = 0.04$, $w_1 = 0.017, w_2 = 0.030$)



(a)

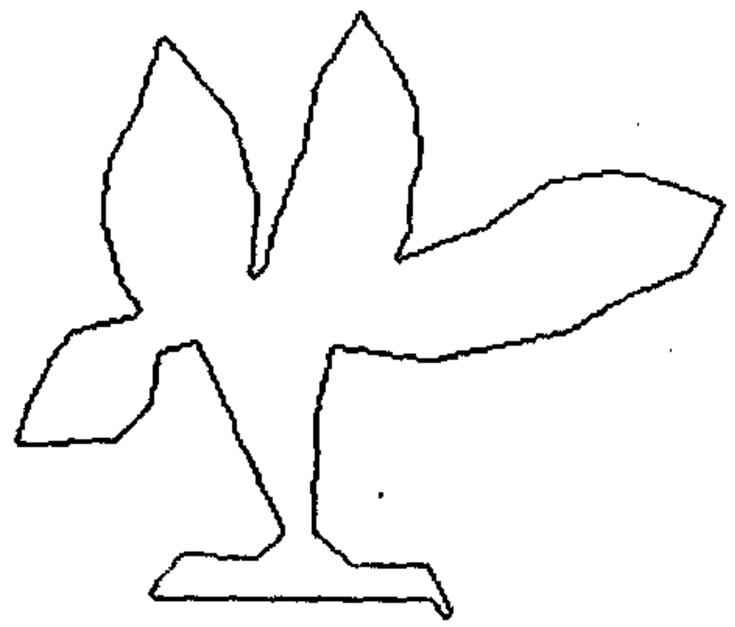


(b)

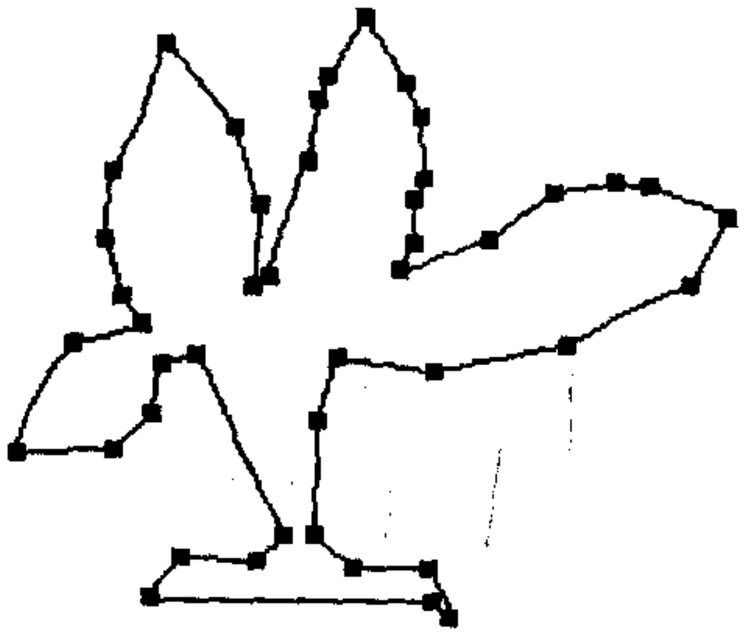


(c)

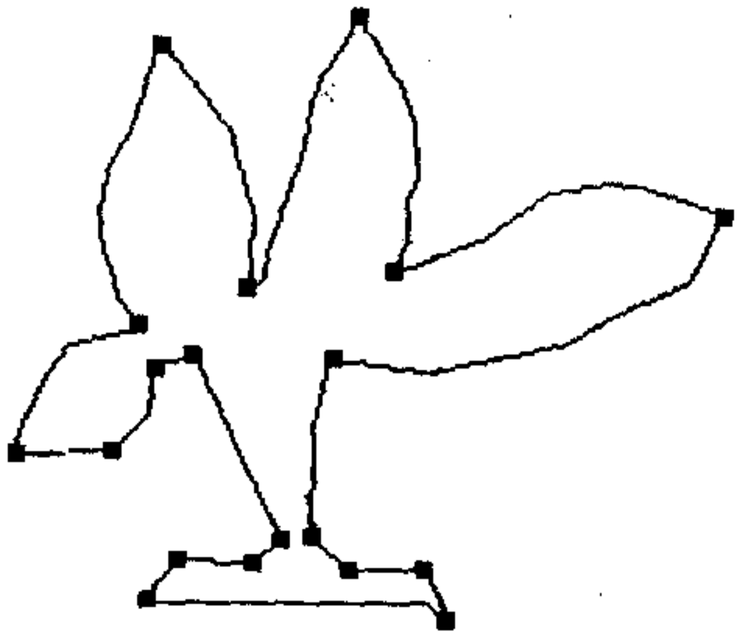
Figure 4.3: (a) Original binary image. (b)Detected corner points (k=8, b=1, $u_0 = 0.1, \theta = 0.05$, $w_1 = 0.039, w_2 = 0.042$) (c) Detected corner points(k=12, b=1, $u_0 = 0.09, \theta = 0.05$, $iw_1 = 0.017, w_2 = 0.020$)



(a)

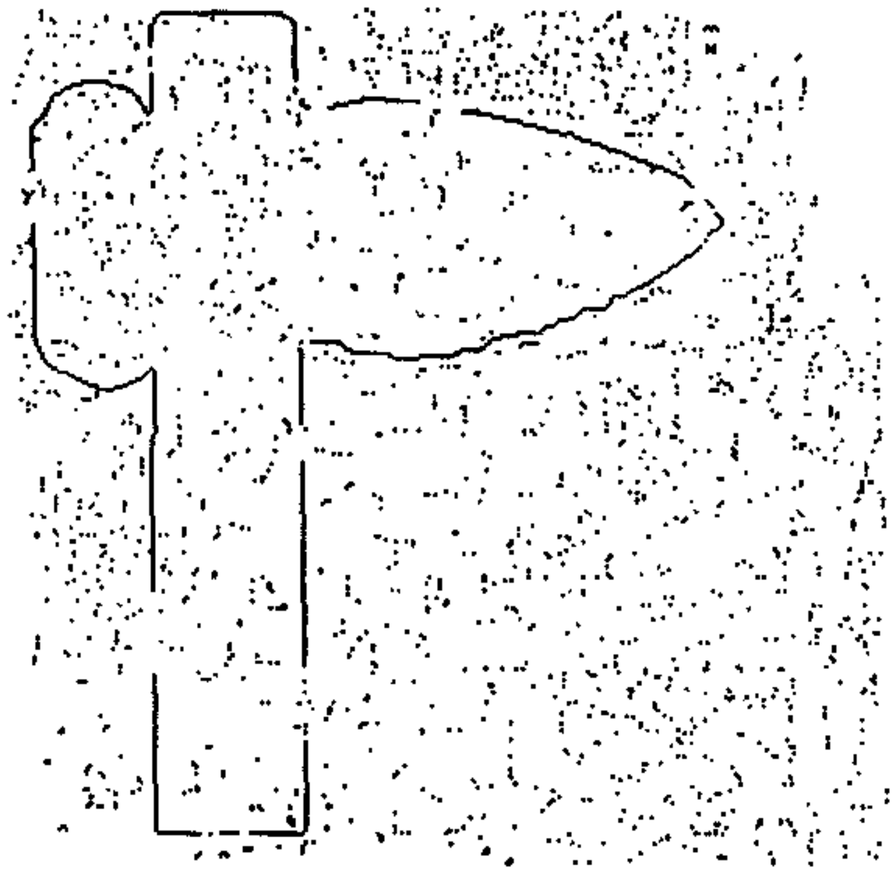


(b)



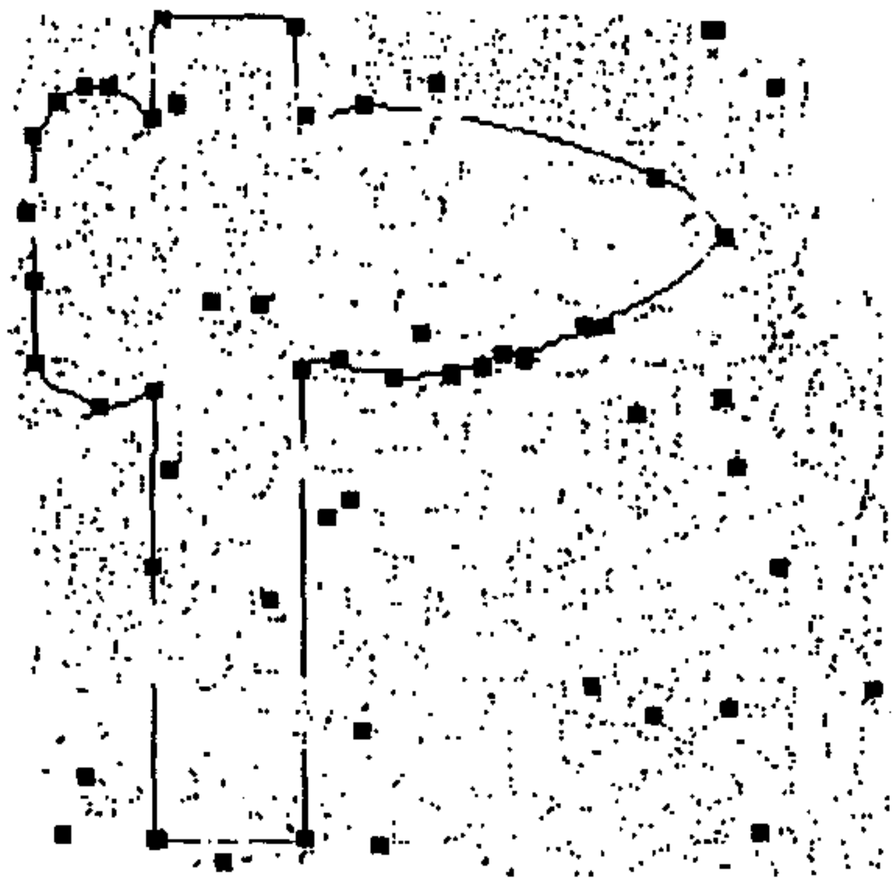
(c)

Figure 4.4: (a) Original binary image. (b)Detected corner points (k=3, b=.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)

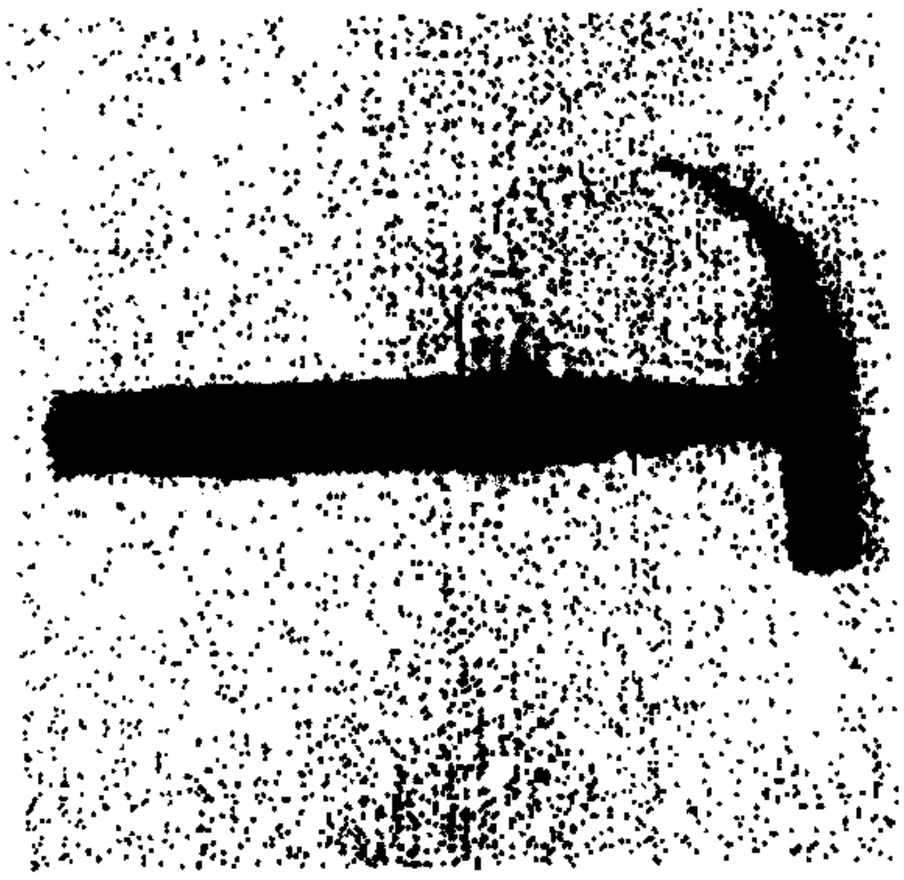


(c)

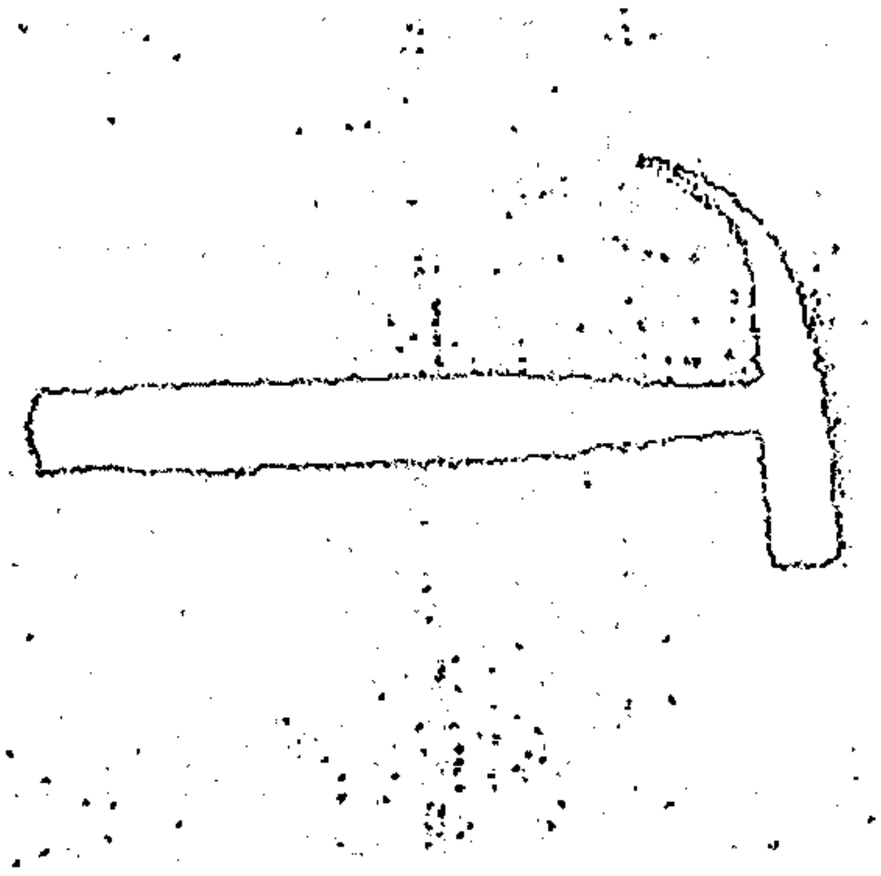
Figure 4.5: (a) Original binary image. (b)Detected corner points (k=10, b=0.2, $u_0 = 0.1, \theta = 0.05$, $w_1 = 0.020, w_2 = 0.030$) (c) Detected corner points(k=12, b=1, $u_0 = 0.006, \theta = 0.003$, $w_1 = 0.012, w_2 = 0.023$)

4.3 Results for Gray Images

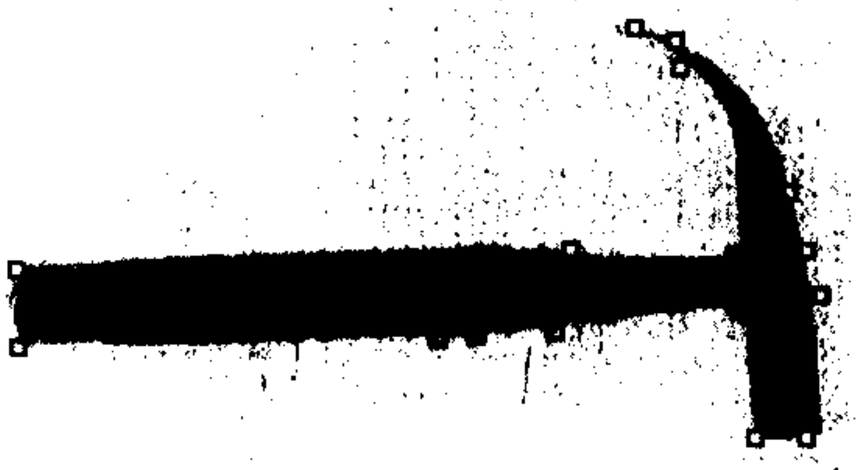
The results for gray images are shown in Figures 4.6-4.8.



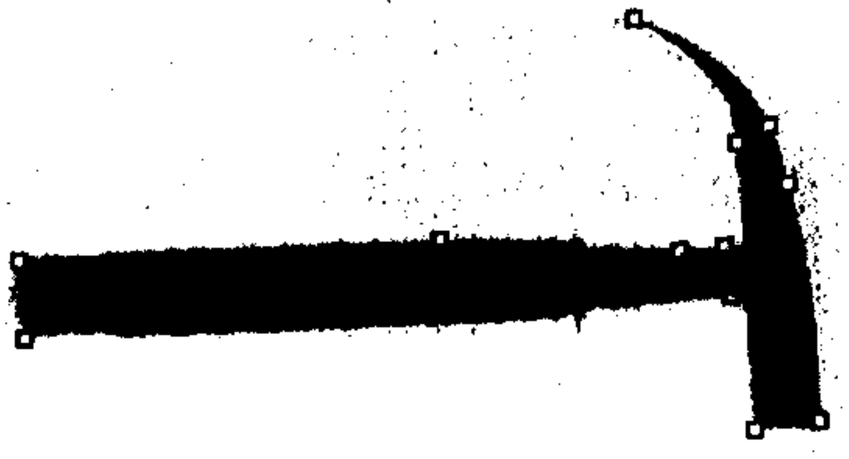
(a)



(b)

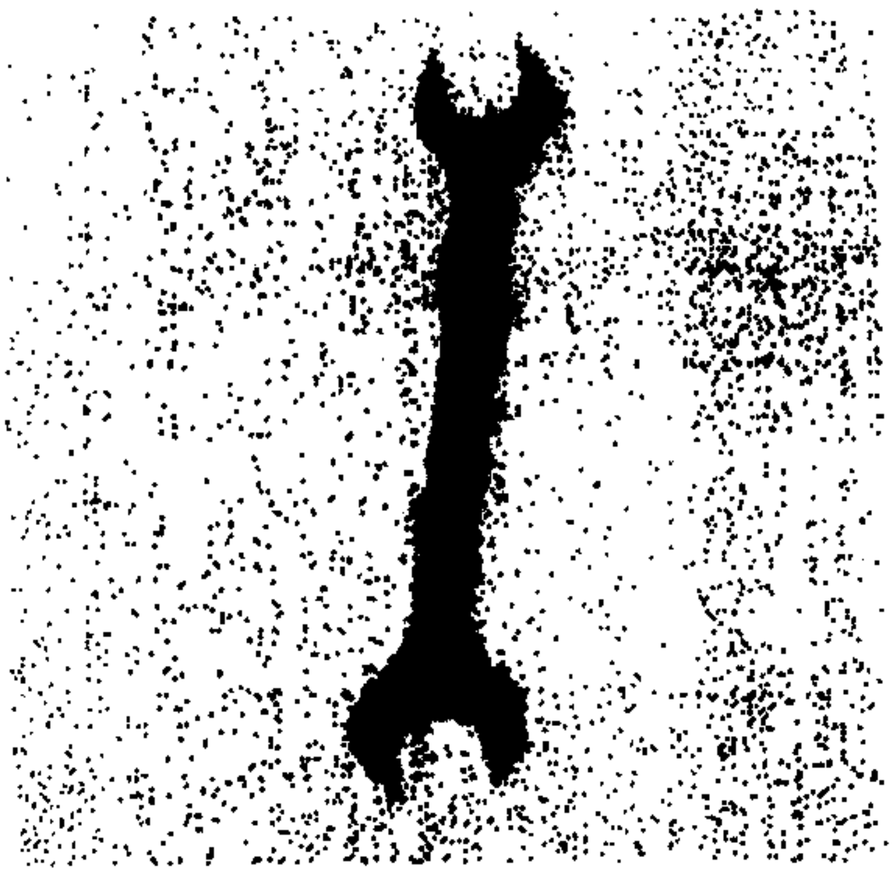


(c)

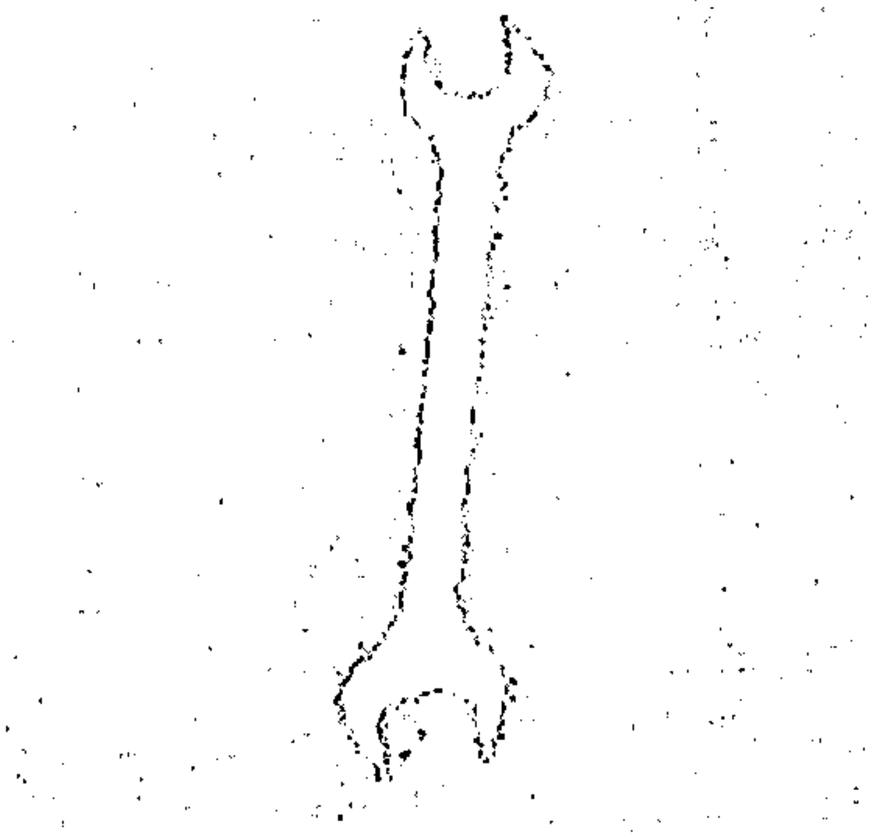


(d)

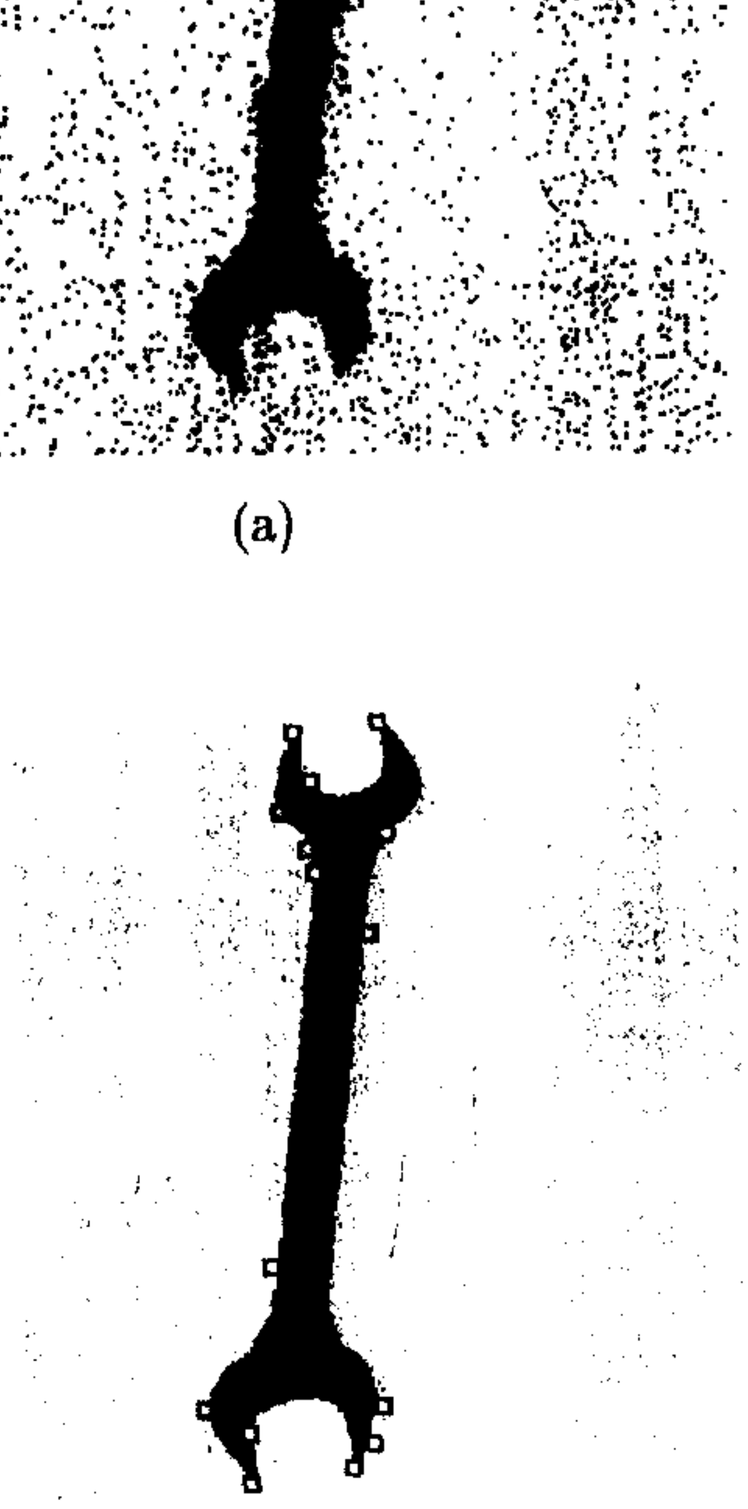
Figure 4.6: (a) Original graylevel image. (b)Edge Image (c)Detected corner points (k=5, b=0.2, $u_0 = 0.5$, $\theta = 0.25$, $w_1 = 0.015, w_2 = 0.034$)(d)Detected corner points (k=9, b=1, $u_0 = 0.24$, $\theta = 0.1$, $iw_1 = 0.022, w_2 = 0.088$)



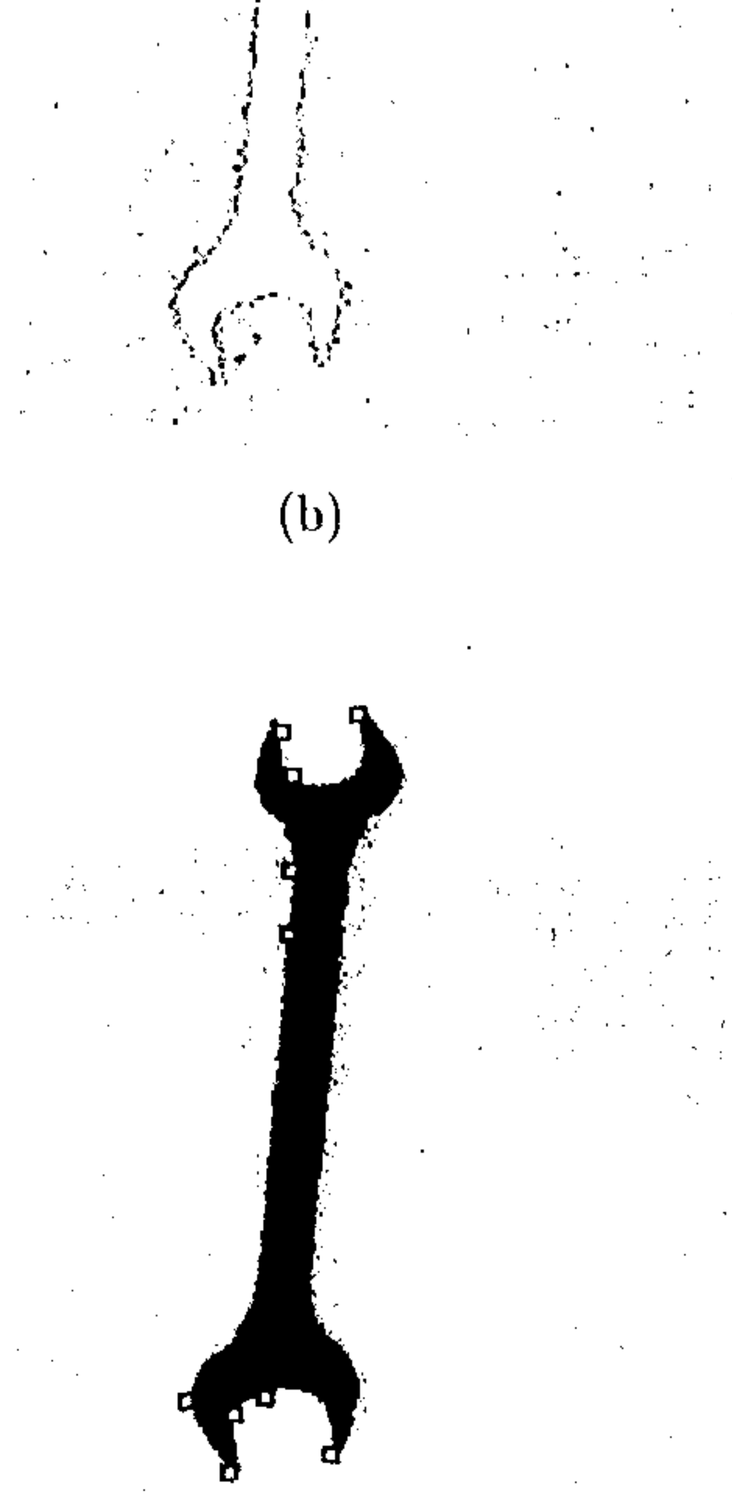
(a)



(b)

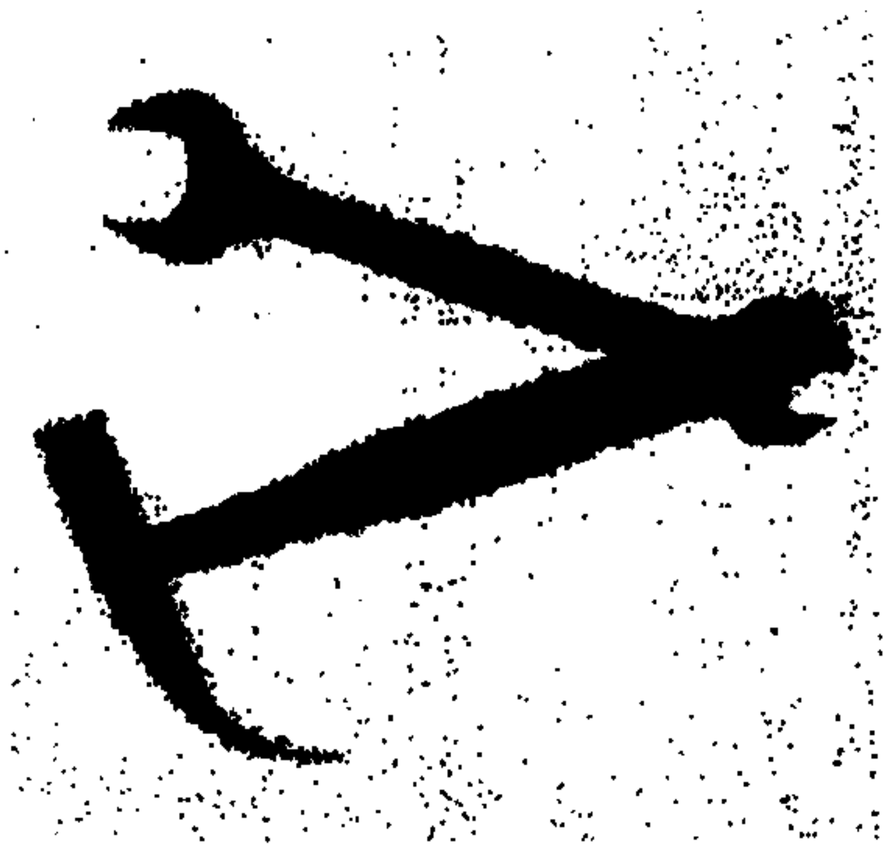


(c)

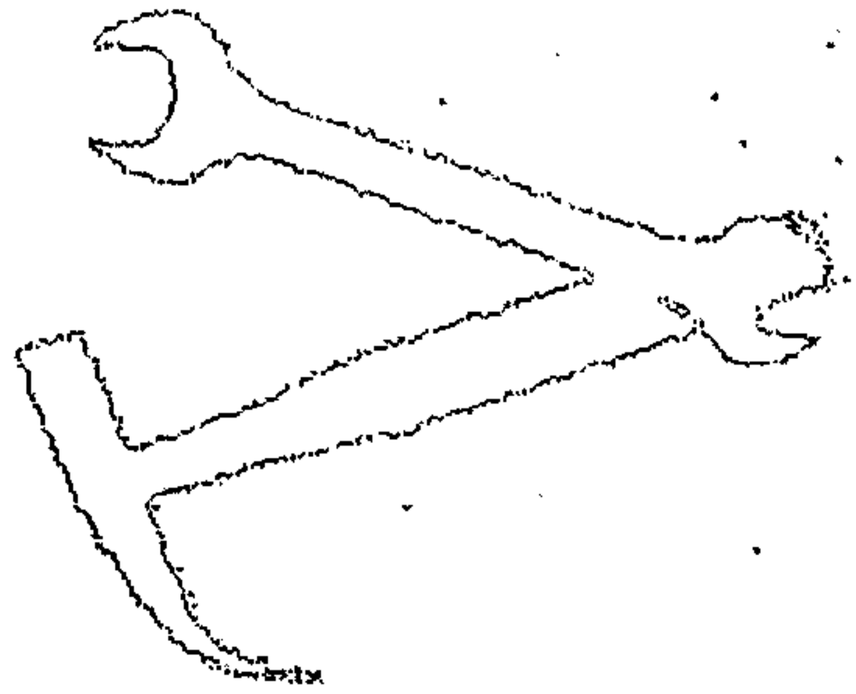


(d)

Figure 4.7: (a) Original graylevel 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, $u_0 = 0.1$, $\theta = 0.04$, $w_1 = 0.021, w_2 = 0.093$)

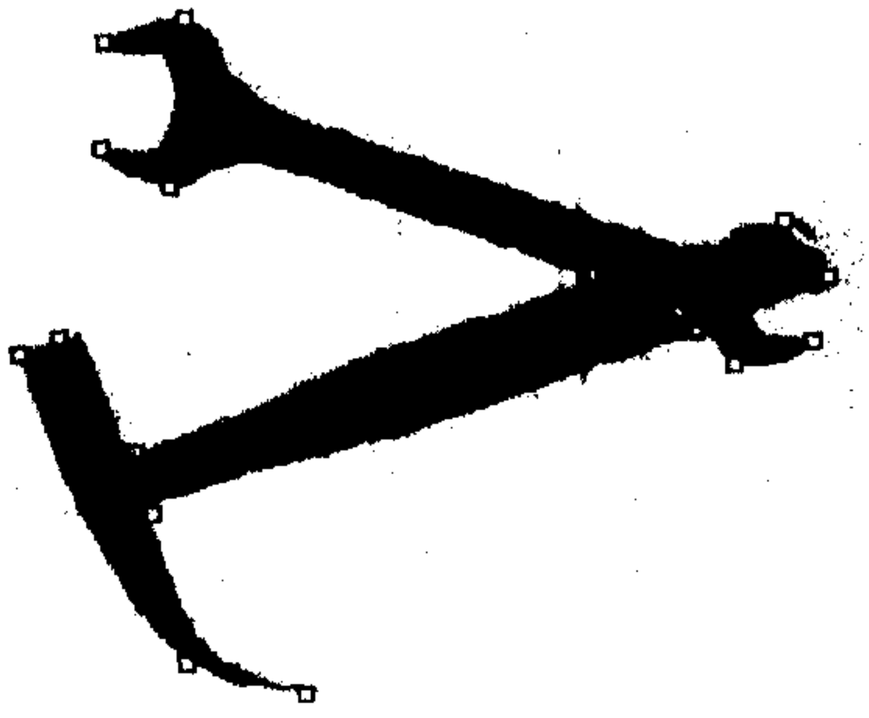


(a)



(b)

(c)



(d)

Figure 4.8: (a) Original graylevel 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, $u_0 = 0.24$, $\theta = 0.1$, $w_1 = 0.022, w_2 = 0.088$)

4.4 Discussion and Conclusion

The results are dependent on the initial size of the neighborhood. Smaller the neighborhood size, less the interaction among the distant points. This results in finer detail of the boundary segment. The finer detail disappear as we increase the neighborhood size. The method performs in noise environment and open boundary segment. The performance of the network for gray images depends on the proper detection of edge strength vectors.

Chapter 5

Conclusions and Scope of Future Work

We developed a corner detection algorithm in a connectionist framework. This framework has the advantage of having local computation for the nodes in the network. The convergence of the network has been proved with some restrictions on the link weights, maximum initialization value for 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 initial selection of radius of neighborhood. For a smaller initial size of the neighborhood, a finer description of the dominant points can be obtained. On the other hand, for a larger neighborhood, a coarse description will be performed. Hence the methodology provides us a flexibility to obtain different resolution depending on the subsequent higher level visual task to be performed. The algorithm also performs in a noisy environment with open object boundaries.

The network used here does not use any learning procedure. In gray images, the performance of the algorithm depends on the edge detection scheme. We used the edge strength vectors over a neighborhood to initialize the cornerity vector at a

Bibliography

[1] I. M. Anderson and J. C. Bezdek, "Curvature and tangential deflection of discrete arcs: A theory based on the commutator of scatter matrix pair and its application to the vertex detection in planar shape data" IEEE Trans. Pattern Anal. Machine Intell., vol. PaMI-6, pp. 27-40, 1984.

[2] J. Basak, B. Chanda, D. D. Majumder, "On edge and line linking with connectionist model", IEEE Trans. Systems. Man. Cybernetics., vol. 24. No.3, 1994.

[3] J. Canny, "A computational approach to edge detection", IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-8, pp.679-698, 1986.

[4] J. G. Dunham, "Optimum uniform piecewise linear approximation of planar curves," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-8, pp. 67-75, 1986.

[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] R. C. Gonzalez and R. E. Woods, *Digital image processing*, 1992.

[7] M. H. Han, D. Jang, and J. Foster, "Identification of cornerpoints of two-dimensional images using a line search method," Pattern Recognition, vol. 22, pp. 13–20, 1989.

[8] T. Pavlidis, "Algorithms for shape analysis and waveforms," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-2, pp. 301-312, 1980.

[9] A. Rosenfeld and E. Johnston, "Angle detection on digital curves," IEEE Trans. Comput., vol. C-22, pp. 875-878, 1973.

[10] 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.

[11] P. V. Sankar and C. V. Sharma. "A parallel procedure for the detection of dominant points on a digital curve

[12] D. M. Tasi, "Boundary based corner detection using neural networks", Pattern Recognition, vol. 30, pp. 85-97, 1997.

[13] J. Wang, Wu, Huang, and Wang, "Corner detection using bending value," Pattern Recognition Letters, vol. 16, pp. 575-583, 1995. e." Comput. Graphics Image processing, vol. 7, pp. 403-412, 1978.

[14] A. Rosenfeld and A. C. Kak *Digital Picture Processing*, Academic Press,æNew York, 1982.

[15] T. Kohonen *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 1988.

[16] 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

[17] J.Hertz, A. Korgh, R. G. Palmer, *Introduction to the theory of Neural Computation*, Addison-Wesley, California, 1991.

[18] D. H. Ballard and C. M. Brown, *Computer Vison*, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1982.

[19] S. Haykin, *Neural Networks, A Comprehensive Foundation*, Prentice-Hall, Upper Saddle River, New Jersey, 1994.