Connectionist Models for Certain Tasks Related to Object Recognition

Jayanta Basak
Machine Intelligence Unit
Indian Statistical Institute
Calcutta - 700 035
India

A dissertation submitted to the Indian Statistical Institute
for the partial fulfillment of the degree of
Doctor of Philosophy
1994

To my parents.

Acknowledgements

At the outset, I take this opportunity of expressing my indebtedness to Prof. S. K. Pal for his guidance and encouragement during the course of this work. Without his thoughtful suggestions and ernest involvement, it would not have been possible for me to complete this work. I gratefully acknowledge Prof. D. Dutta Majumder for his advices and overall involvement during the initial stage of this investigation. I express my gratitude to one of my colleagues, Dr. Santanu Chaudhury, who actually introduced me to the field of computer vision. Throughout the investigation, I enjoyed the attachment with Dr. C. A. Murthy, whose mathematical excellence helped me a lot in venturing into the field of neural networks. I like to express my heartfelt thanks to Dr. N. R. Pal, who helped me significantly in the later part of this work. I acknowledge Dr. B. Chanda for his help in the earlier part of this research.

I express my regards to Dr. M. K. Kundu for his encouragement especially during the later stage of this work. I am thankful to many of my friends and colleagues, particularly, Dr. S. R. Das, Dr. P. Bose, Mr. D. P. Mukherjee, Mr. A. Mukherjee for rendering technical and nontechnical help and suggestions in different stages of this investigation. I acknowledge KBCS center, ISI, for partly supporting this investigation and providing computing facilities. The facilities provided by the Computer and Statistical Service Center of this Institute are also acknowledged. Thanks are also due to Mr. S. Chakraborty for preparing the diagrams.

I thank office staff and colleagues of Machine Intelligence Unit, ISI for their help. I also acknowledge the Indian Statistical Intitute, Calcutta for providing me the opportunities to carry out this research.

I am indebted to my parents, family members and relatives for their support and encouragement.

ISI, Calcutta
September, 1994.

Jayanta Basak

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Scope of the Thesis

1.1 Introduction

Recognition of objects in an image, according to Suetens et al. [1], refers to the task of finding and labeling parts of a two-dimensional image of a scene that correspond to the real objects in the scene. Object recognition is necessary in a variety of domains like robot navigation, aerial imagery analysis, industrial inspection and so on. Normally, different strategies for object recognition [1]–[5] involve establishing some model for each object, i.e., some general description of each object, and then labeling different parts of the scene according to the knowledge about the models.

Object models can have two-dimensional (2D) or three-dimensional (3D) descriptions. 2D descriptions are generated from viewer centered representations where each view is represented using shape features derived from graylevel or binary images of the prototype object models. On the other hand, 3D descriptions require view point independent volumetric representations those permit computations at an arbitrary viewpoint. Generation of 3D descriptions from the captured 2D images is a computationally difficult problem [6] (one approach is to generate $2\frac{1}{2}$D sketch from the 2D image [7]), whereas the 2D descriptions of the objects can be constructed more easily. Construction of 2D descriptions is straight forward for flat objects like hammer, spanner etc., and such descriptions are often used in in-

spection problems of flat industrial objects. Moreover, in several other tasks like character recognition, analysis of remotely sensed imagery etc, 3D information is neither necessary nor available, and information processing is to be performed only on the basis of 2D descriptions. The present thesis concerns with the tasks related only to 2D descriptions (2D object recognition). 2D object recognition involves mainly two stages, first, extraction of features from the captured image and the associated preprocessing tasks, and second, interpretation of the extracted feature set. These two stages are briefly explained below.

A : Feature Extraction and Related Preprocessing

In the image grabbing process, the light coming from the scene is projected onto a plane (image plane), and the image plane content is digitized into a two-dimensional array. Each location in the array specifies a position in the image plane, and the location contains an integer value specifying the intensity of the image at that position, i.e., the amount of light received from the scene after projection (at that location). In the case of a color image, the color information is also stored in each location of the array. To recognize the objects present in the image of the scene, the image is preprocessed in several stages, and consequently some characterizing features are derived from the image. These features are used for further interpretation task. Different stages involved in the feature extraction and allied preprocessing tasks are as follows :

Enhancement/noise cleaning : Often it is necessary to improve the signal-to-noise ratio in a grabbed image before any further processing is done. Sometimes the contrast of the image also needs to be enhanced. A widely used technique for noise cleaning is to convolve the image with some smoothing mask which basically takes the neighborhood information of a pixel and tries to restore the original information on that basis. Marr [8] has shown that Gaussian smoothing performs optimally in many cases. The contrast of the image can also be enhanced by graylevel modification. This includes various techniques like graylevel correlation, histogram equalization etc. Detailed discussion on this topic is provided in [9], [10].

Segmentation/edge or line detection and linking : After noise cleaning, the image is partitioned (segmented) into meaningful homogeneous re-

gions. The term 'meaningful' is problem dependent. Segmentation can be performed depending on different properties such as intensity value, texture color etc. Various techniques like histogram thresholding, clustering, and iterative pixel classification using classical approaches, fuzzy set theoretic approaches, and connectionist approaches are reported recently in a survey by Pal and Pal [11].

Once the image has been segmented, boundaries of the object regions can be easily found out. In many cases, the boundaries of the objects are extracted by finding out the edge/line information in the image. An edge, in an intensity image, is a border between two adjacent extensive regions where graylevel changes abruptly. On the other hand, a line (straight or curved) is found in an intensity image if graylevel along a thin strip is approximately uniform and distinctly different from its neighboring regions (lines are sometimes called roof edges) [9]. Although edge detection helps in finding out the boundaries in many situations with better efficiency, it may not provide closed contours for separating objects from the background. Lines are useful in many cases to identify the junctions between two different adjacent planes in 3D objects [12]. Lines are also useful in identifying different structures like roads, rivers etc. in remotely sensed imagery [13].

Continuity in the detected edge/line segments is desirable for extracting the boundary information or the characteristic features of the objects. But many times, continuity may not be preserved due to the presence of noise, imprecision in the image grabbing process etc. and significant edge/line points may be lost. It is therefore, often necessary to link or connect different edge/line segments in an image before extraction of features. Again, for adequate preservation of structural information in remotely sensed images or aerial images, edge/line linking is often necessary. Two major approaches for doing this task are edge/line following and relaxation labeling. The details will be discussed in Section 1.2.1.

Region/edge based feature extraction : After finding out object regions (often they are termed as blobs [6]) and/or edges, characteristic features are extracted. We refer these features which are directly extracted from the edge/line or region information as primary features. Primary features, extracted by region analysis, try to capture region information like grayvalue, texture, color, area, perimeter, compactness etc. These features essentially

represent some global characteristics of the object regions (blobs). On the other hand, boundary characteristics of the objects can be obtained from their edges or contours. These characteristics include global features like the components of the Fourier descriptor of the boundary information, or local features like curvatures at different boundary pixels, junction points (where more than one edge/line meet), corners (where the curvature changes abruptly), length of edges between adjacent corners, holes and so on [14]. Some of the local features which essentially carry some structural information of the objects are often termed as primitives.

Secondary feature extraction : The set of features required for interpretation should be able to provide some structural/relational information about the objects and should be invariant under different conditions [6]. To have such characteristics of the features, often secondary features are extracted depending on the nature of relative positions or groupings of the primary features. The kind of secondary features to be extracted is normally goal-driven and they should be able to provide the spatial description of the environment.

The secondary features include relations between different primitives or different subparts of an object which can be used to interpret an object (relational features). For example, the relations like LEFT, RIGHT, TOP, BOTTOM etc. may be used to specify the location of different subparts in an object with respect to each other [15], [16]. Note that, these relations are often used to specify the relational descriptions of the objects and they may not be referred as secondary features in some literature. However, for the sake of uniformity, we refer these relational descriptions as the secondary features. The distances between different primitives may also be considered as the secondary features.

The primitives may also be grouped according to human psychovisual characteristics or the way of perception (perceptual grouping). The perceptual grouping may be performed based on similarity, proximity, collinearity etc. For example, U-contours, small parallel segments etc. often carry structural information in aerial imagery [17], and these can be used as invariant features to interpret the scene. Details on such groupings are provided in [18], [19], [20].

B : **Interpretation of the Features**

After extraction, the set of features is used to identify the objects present in the scene. To perform this task some models (object description) are stored for each object in the knowledge base as mentioned before. The derived set of features are then matched against the selected attributes of the object models utilizing various techniques. In the matching process, all features may not have equal importance (viz., some objects may have some distinct features), and many times features are ordered according to some preassigned *priority* (feature ranking).

Note that, the feature extraction and interpretation (matching) processes are dependent on each other. In other words, extraction of the features is dependent on the type of objects to be found in the scene and consequently the interpretation process to be followed. Similarly, the matching strategy depends on the set of extracted features. For example, in the case of region based features producing some global characteristics of the objects, statistical or distance based decision rules perform better. On the other hand, for structural and relational features characterizing the local properties, computationally expensive algorithms based on relaxation labeling techniques, association and relational graph matching techniques, generalized Hough transform techniques, heuristic search techniques etc. [4] are necessary. The algorithms using the local and relational features have several advantages over those using only global features in many cases like interpreting more than one object simultaneously, dealing with occlusions.

In the task of interpreting the feature set, sometimes twofold processing is performed. One is hypothesis generation which is essentially a bottom-up process to generate the hypotheses about the presence of objects from the extracted feature set. The other is the hypothesis verification, which is a top-down process where the model attributes are matched against those of the image-derived features. However, depending on the model descriptions, the algorithms are widely different. A brief overview of different techniques for interpreting the derived feature set is presented in Section 1.2.2.

◇

The above discussion harps on the point that although 2D object recognition broadly involves two stages : feature extraction (and allied preprocessing tasks)

and interpretation of the feature set, various techniques can be followed in each step of these stages. Whatever methodologies be used for feature extraction or interpretation of the feature set, several requirements are to be satisfied for their applicability in real-life tasks. First, the methodologies should be robust and fast. Preferably, the algorithms should be implementable on parallel hardware. Second, in the task of interpreting the feature set, sometimes it is necessary to associate degree of importance (or weights) with the features. If the association of importance or weights with the features can be performed automatically depending on the environment then the methodology may prove to be more versatile.

The performance of the classical algorithms, in general, is not comparable with the real-time performance of the biological systems. A good comparison of the human visual system and the machine recognition systems has been provided in [21]. Although the objective of machine vision is not necessarily to emulate animate vision, but the latter is found to be capable of adapting in the environment and seems to be more robust in its behavior. Therefore, if some findings in the fields of neurophysiolgy and psychology regarding visual cognition can be taken into account in the development of artificial systems then the performance of these systems may plausibly be improved. As discussed by Skryzpek [22], the findings in neurophysiology may provide a bottom-up guideline, while the findings in psychology may provide a top-down guideline for such improvement. Since *artificial neural networks* (ANN) provide the closest computational framework of biological nervous systems, the neurobiological and psychological findings may possibly be incorporated into the artificial recognition systems in a better way using ANN models. Artificial neural networks are often referred as *connectionist models*[1] or *parallel distributed models* or *computational neural networks* or simply *neural networks*.

Artificial neural network models are massively parallel interconnected networks of simple processing elements (neurons), intended to interact with the real world in the same way as biological nervous systems do. This does not necessarily mean that artificial neural networks are able to emulate the behavior of biological systems, rather they sometimes resemble biological systems in a very naive manner. However, the neural networks (or connectionist models) having several basic advantages like robustness, scope for massive parallelism and capability of learning from examples (adaptivity and generalization capability) provide a tempting paradigm

[1]Sometimes connectionist models refer to more general kind of networks which have the capability of distributed knowledge representation.

for dealing with the real-life recognition tasks. For example, in the task of edge/line linking, neural networks may provide a massively parallel computational paradigm with very simple processing elements. Similarly, for the interpretation of feature set in a scene, they provide not only a paradigm for massively parallel computation, but also a robust, structured framework for learning the importance of different features depending on the environment. Various standard and application-specific artificial neural network models have been reported in the literature [23]–[31]. The effectiveness of neural networks has been tested in a variety of application-specific tasks like optimization [32]–[35], image segmentation [36]–[38], enhancement and restoration [39], [40], scene labeling [41], speech perception [42], [43], natural language processing [44], [45], motion analysis [46], [47], expert systems [48], [49], rule generation [50] and so on.

Objective of the Thesis

The objective of the present thesis is to provide some results of investigation demonstrating the effectiveness of connectionist approach for dealing with certain tasks related to object recognition, namely, the tasks of edge/line linking and interpretation of features. The investigation includes – the updating of edge/line point strengths based on neighborhood information for their linking, mapping relational graph homomorphism onto neural networks for structure matching, development of neural network model for mixed category perception (under both supervised and unsupervised modes), structural learning and recognition, and finally, development of a connectionist system for object recognition incorporating some of the psychological findings. Some new models are developed in this regard. Performance of these models are also compared with that of the existing models. Before going into the scope of the thesis let us present a brief overview of both classical and connectionist approaches to edge/line linking and 2D object recognition.

1.2 Overview of Classical Approaches

1.2.1 Edge/Line Linking

As mentioned before, an edge, in an intensity image, is a boundary between two adjacent extensive regions where graylevel changes abruptly. On the other hand, a

line (straight or curved) is found in an intensity image if graylevel along a thin strip is approximately uniform and distinctly different from its neighboring regions.

In real images, it is very difficult to detect edge or line points due to the presence of noise in a scene. The presence of noise creates randomness in intensity profile to a great extent. Moreover, the intensity in a natural image changes over a wide range of scale. Marr [8] has shown that there exists some inherent difficulty in detecting edge point locations exactly. The inexactness in detecting the edge point locations is governed by the uncertainty principle given as

$$\Delta x.\Delta\omega \geq \pi/4.$$

Here, $\Delta\omega$ is the variance in the edge filter's spectrum in the frequency domain, and Δx is the variance in edge location. In other words, if Δx is very small, i.e. edges are accurately located then $\Delta\omega$ would be large and a number of redundant or noisy edges would be detected. On the other hand, if $\Delta\omega$ is decreased (in order to decrease the effect of noise) then there would be smoothing effect and the edges will be dislocated. The uncertainty principle leads to the idea of optimal filtering where an intensity image is first convolved with a Gaussian mask and then the zero crossings are detected (as the edge points) from the Laplacian of the convolved image. The linearity of the Laplacian and the convolution operator gives rise to the symmetric filter, namely, Laplacian of Gaussian (LoG). Canny [51] has shown that the detection of the zero crossings in the second directional derivative (instead of using symmetric Laplacian operator) gives better results.

The line detection algorithms are also involved and there exist the same kind of problems as in the edge detection techniques. The main idea in line detection algorithms is to match suitable templates corresponding to the desired type of line segments in order to find out the similar line segment structures. In literature [52], quite a few line detection algorithms have been presented. In one approach of line detection using template matching [53], nine different 3×3 templates are proposed to measure the likeliness to linear features like edges and lines. Semilinear and nonlinear operators [54], [55] were designed incorporating various constraints in the linear features. In these approaches, maximum of the responses due to these templates is considered as index of existence of the line points. The responses are later thresholded to separate out the true line points from the non-line ones. Facet models were also proposed for surface fitting and consequently line points were detected [56],[57].

For the task of feature extraction, continuous edges and lines are almost always desired (this is necessary for extraction of local features). Some portions of the edge or line segments may be lost due to the presence of noise, shadowing and other reasons. Moreover, thresholding the output in line detection algorithms result in discontinuities or thickening of the lines. Disconnected line or edge segments may be connected to produce continuous lines or edges. A number of sequential and parallel algorithms have been proposed so far for the task of edge/line linking. Let us present a few of them in brief.

Hough transform was used to link various line or edge segments, where the image space is transformed into parameter space [58]–[63]. Each line or edge segment in the image space can be represented as a parametric equation. For example, in the case of a straight line, slope and intercept with one of the axes specify the parameter values. In the case of curved segments, more than two such parameters is necessary. Every point in the image is transformed into the parameter space where the points lying on a line/edge form a cluster. The location of the cluster depends on the parameters specifying the line segment (details on Hough transform is provided in Chapter 6 of this thesis). Even if the line segments are broken in the image, the formation of clusters is not affected so long as the parameter values of the broken segments are very close. Note that, use of Hough transform is limited by the fact that it does not yield any information about the exact extent of the line segments being considered. If the response values in the parameter space corresponding to the clusters are considered then it may provide some information about the length of the line/edge segments, but it would not be able to specify the starting and terminating pixels on the edges/lines). Secondly, the selection of threshold used in these techniques is also a nontrivial task and completely depends on the nature of the scene. Moreover, Hough transform works for edge/line linking only when the nature of edge/line segments approximates some known parametric form. However, this transform provides an efficient method (which has ample scope for massive parallel implementation) for extracting out the linear structures in an image.

Apart from Hough transform, there are several other techniques for performing this task. One of them is 'edge following' or 'contour tracking'. This method iteratively traces the possible edge/line points in an image (the edge/line points, to be linked with the current edge/line points, are searched for). Ballard and Brown [64] developed a sequential algorithm for edge following. A search technique

based on angular proximity and positional proximity was used for linking different line segments [13]. The technique was consequently used to detect linear structures from aerial photographs. Various other such techniques are mentioned in the survey of Suetens et al. [1]. Note that, selection of starting points is a problem with these techniques.

In a different approach [65], a line segment of predefined length was predicted to exist at some particular location depending on the response to some line detection operators. The prediction was then verified against statistical tests performed on that line. Symbolic processing based on closeness and orientation of small line segments has also been used to detect long continuous lines [66].

Relaxation labeling has also been used in linking edge/line segments. In this technique, each pixel is initially labeled indicating whether it is an edge/line point or not. The possibility of a pixel being an edge/line point is updated depending on some criterion function of the characteristics (like strength and orientation) of the neighboring edge/line pixels. Two kinds of relaxation techniques are normally followed for this task. One of them is probabilistic relaxation [67], [68], where the probability values of the pixels being edge/line pixels are updated. In the other kind, namely discrete relaxation [69], [70], the pixels are either flagged as edge pixels or non-edge pixels. There is another appealing approach [71], where a cover over the broken edge segments was formed, and broken lines and edges were linked by the edge induction.

◇

We have briefly presented an overall idea of the various edge/line linking techniques which are often necessary in the process of feature extraction. Let us now present a brief review of different classical algorithms developed for 2D object recognition. Here the stress is given mainly on interpretation of features.

1.2.2 2D Object Recognition

In literature, there exist several excellent reviews [1]–[5], [14] on object recognition. Wallace [4] has classified different existing techniques according to their approaches. We have presented the review in a similar line as described by Wal-

lace. As mentioned in Section 1.1, the selection of algorithms for interpretation of features depends on the type of features. If the features reflect some global characteristics of the objects, statistical or distance based decision rules perform better. On the other hand, for structural and relational features characterizing the local properties, computationally expensive AI-based techniques [4] are necessary.

The global feature based methods essentially derive some global properties from the image like gray value, color, area, perimeter, compactness, number of holes etc., and based on these properties the feature vector is classified into different categories using statistical or distance based classification rules [72]. Fourier descriptors derived from the object boundary are also used to derive the feature vector. Since these kinds of features reflect global characteristics of objects, it is difficult to deal with the problems of occlusion by this approach. The local (structural/relational) feature based techniques include syntactic classification, graph matching, generalized Hough transform based methods, relaxation labeling, heuristic search, boundary correlation etc.

The syntactic approach to pattern recognition involves the representation of a pattern by a string of concatenated subpatterns called primitives. These primitives are considered to be the terminal alphabets of a formal grammar whose language is the set of patterns belonging to the same class. Recognition, therefore, involves a parsing of the string. Fu [73] has presented a nice introduction to a variety of techniques based on this approach. Various applications of this approach include character recognition, chromosome analysis, identification of skeletal maturity from X-ray images , machine part recognition, shape analysis and recognition [73]–[81]. Concept of fuzzy sets has also been incorporated in syntactic approach to increase the generating power of a grammar [82]. Recently, this approach is not being well attended by the researchers as compared to other approaches discussed subsequently.

Besides the aforesaid conventional pattern recognition techniques, there are some AI-based techniques which are commonly used for 2D object recognition. Let us now describe them.

Association graph and Relational graph based techniques

In association graph based technique, a graph is formed by representing the acceptable matches between the scene and model features as the vertices, and connecting

the compatible associations by edges. Compatibility can be determined on various constraints. After formation of the association graphs cliques are found in order to obtain the most compatible matches between the objects in the scene and the model objects.

Yachida and Tsuji [83] used a simple kind of feature graph representation and a hierarchical model involving coarse and fine representations for object recognition. Chen et al. [84] used object models in the form of local feature graphs. Bolles and Cain [85] developed a sophisticated 2D object recognition procedure, called the local feature focus method using two types of local features, namely, corners and holes. Object models were formed by analysis of the corresponding CAD models of the objects.

Koch and Kashyap [86] used association graphs to find matches for polygonally approximated 2D objects under occlusion. Corners are used for the detection and localization of the objects. Polygonal moments of the corners are used to find similarity between scene and model corners. An association graph is formed considering the matches and compatibility constraints, and the maximal clique in the graph is used as a hypothesis. The generated hypothesis is then verified by checking the region consistency. Wen and Lozzi [87] formed some subpolygons by drawing lines which are parallel to the sides of the model and scene objects. The invariant moments of the subpolygons have been used for the purpose of matching. Han and Jang [88] also used association graph to represent the compatibility between the model and scene features (which were curvature points of the 2D objects). A heuristic search (discussed later) was then used to enhance the process of finding out the cliques of the association graph.

Another graph based technique for object recognition is based upon the relational descriptions of the objects [15], [16]. The knowledge base of the recognition system consists of structural description of prototype objects. The structural description of the objects consists of a set of primitives corresponding to various parts of the objects and a set of n-ary relations defined over the set of primitives. Matching rule between the objects is defined in terms of relational homomorphisms, monomorphisms and isomorphisms. Relational homomorphism maps the primitives of the test objects to a subset of the primitives of a prototype where all the interrelationships among similar primitives are preserved. When the set of candidate primitives is comparatively smaller, this is equivalent to finding a small object as a part of a large object. To tolerate noisy and erroneous environment, a concept

of ϵ-homomorphism has been formulated. ϵ-homomorphism is a mapping such that sum of the weights of the relations between the primitives in the candidate which are not satisfied in the corresponding subset of the prototype primitives is less than the threshold ϵ. Hence, matching is a problem of finding a relational homomorphism between the shapes. This can be solved by general constraint satisfaction tree search. Searching operation can be speeded up by using look-ahead, forward-checking and/or relaxation operator.

Eshera and Fu [89] developed an image understanding system using attributed relational graph and used it for locating objects in a multiobject scene. Grimson and Lozano-perez [90] developed a recognition system, called RAF (Relational Attitude Finder), that locates 2D and 3D objects from noisy and occluded data. Wallace [91], [92] proposed a heuristic search based method for recognition of occluded objects using both local and relational features.

Generalized Hough transform based techniques

Concept of Hough transform has been discussed in Section 1.2.1. It can be used to extract simple structures like lines, curves of known form, circles etc. [59], [61]–[63]. It was extended to generalized Hough transform (GHT) [93] to match arbitrary contours. In GHT, each edge point in the image is aligned with the model edge points and accordingly, the position of the model object in terms of (x, y, θ) is calculated. Thus for each constituent edge point in each model object, (x, y, θ) values are computed which indicate the plausible locations of model objects determined locally by the edge points. A four-dimensional accumulator array $A[N][X][Y][\Theta]$ is maintained where N is the number of model objects, and X, Y, and Θ are the quantized values of x, y, and θ respectively. Corresponding to each (x, y, θ) value computed for each model object, accumulator value is incremented. After considering all edge points in the image, the peaks in accumulator space are found out, which essentially represent the objects' identities and locations. Different variations of GHT have been devised to recognize the objects from a scene. Instead of considering all edge points, some characteristic features in the model objects can also be considered. Some algorithms also try to reduce the space requirement to maintain the accumulator array at the cost of inherent parallelism embedded into GHT. Let us describe some of the techniques employing GHT for 2D object recognition.

Turney et al. [94] used subtemplate matching scheme in generalised Hough trans-

form space for recognition of occluded objects. The method uses a database of object models containing boundary templates for every stable position of the objects and a set of its salient subtemplates. The matching of subtemplates of the models and the boundary in the image is performed in θ-S space, because it is important to place the subtemplate in correct orientation. The slope θ and the arc length S are determined during boundary extraction.

This method critically depends on the saliency values for different subtemplates. An algorithm was also proposed which determines the saliency of boundary segments of one object with respect to those of a set of other objects. The algorithm adjusts the weights of subtemplates so that different templates correlate poorly. Correlation between the templates is considered in terms of the contribution to the accumulators in the Hough space.

Stockman et al. [95] proposed a method for object detection which accumulates local evidence by pose clustering. In this method, models of 2D objects are organized into a set consisting of (i) real vectors describing boundary segments and (ii) abstract vectors linking primitive features (e.g., a vector connecting two holes). The elements of the set are defined in an object centered coordinate system. For a matched pair of image and model features, a transformation vector consisting of rotation, scaling, and translation parameters is computed. A cluster of match points is formed in the transformed space, and the cluster indicates a consistent set of matching features. This method is expected to be robust because clustering procedure integrates all local information before any recognition decision is made. At the same time, heavily occluded objects cannot be recognized because they would not form large clusters. Also, stray matching points can influence the cluster centre and/or merge two neighboring clusters.

Segen [96] described another pose clustering approach for recognition of occluded objects. Here, a unique search method is used to find viable pose candidates. Initially, objects are allowed to have three degrees of freedom (*rotation*, *x position*, y position). This freedom is reduced by one dimension at a time through a series of one dimensional Hough transforms. This method avoids the need for large multidimensional arrays. Arbuschi et al.[97] also applied Hough transform technique for recognizing mechanical parts.

Bhanu and Ming [98] proposed a method for matching polygonal approximations of partially occluded 2D objects. Here, disparity values (rotational and translational

disparities computed between compatible segments) are clustered using k-means algorithm in the transformed space. A confidence value has been defined for each cluster and the cluster with highest confidence level is selected as the final transformation cluster. Although the method is robust and reliable, it cannot elegantly deal with the cases of severe occlusion. In the technique proposed by Ullmann [99], object locations are guessed depending on the visible edge segments. Then depending on the votes in accumulator space, the invisible (due to occlusion) edge segments are replaced in the image and their consistency are checked with the model segments.

Heuristic search techniques

In these techniques, some estimated positions and orientations are found from the matched set of features. Then a tree search technique based on heuristic reasoning is employed to find out the match for other features. The nodes in the tree normally represent matches between model primitives and scene primitives. Each node is associated with some weight indicating the measure of similarity of the scene primitives with that of the model. Different variations of A* search algorithms are usually employed in the heuristic search process.

Knoll and Jain [100] proposed a technique for occluded object recognition using feature indexed hypotheses (features are fixed length boundary segments). In this technique, desirability of a feature is defined by its benefit-to-cost ratio. The benefit of using a feature depends on the number of matches the feature makes, and the cost is the sum of costs of matching the feature and testing the number of hypotheses the feature generates. Feature selection is performed automatically using a 'greedy' algorithm which chooses features according to their desirability values. Based on the match between the model and image features, a set of hypotheses about the presence and location of the objects is generated and they are verified using object templates.

Ayache and Faugeras [101] developed a working system, named, "HYPER" for 2D object recognition. In this system, hypotheses about identity and position of the objects are generated by comparing privileged segments (segments of longer lengths) of the model descriptions with those in the image. The generated hypotheses are ranked according to the estimated error. Some of the best hypotheses are then evaluated by matching the nonprivileged segments. The matching process is terminated when the number of hypotheses evaluated is large or when a very high

quality measure (quantified by the estimated error) is reached by a hypothesis.

Tropf [102] developed a heuristic search based recognition system for partially overlapped workpieces. The approach is based upon semantic labeling of the primitives. There are several other heuristic tree search based systems including those developed by Rummel and Beutel [103] and Chaudhury et al. [104] for planar shape recognition.

Relaxation labeling based techniques

Various algorithms for object recognition have been developed based on the concept of relaxation labeling (Section 1.2.1). In these techniques, the features derived from the scene are initially matched (and then associated) with the model features according to some similarity measures. Then the labelings (associations) are updated on the basis of some compatibility function which takes care of the labelings of other scene features. The updating process continues till a suboptimal quality of match (may be quantified on the basis of compatibility function) is reached.

Bhanu and Faugeras [105] proposed a probabilistic relaxation technique for matching polygonal approximations of 2D objects (aerial imagery and industrial components). Here, the probability of a match between a model segment and a scene segment is computed based on the attributes like length, intervertices distance, angle between two segments etc. The probability values are updated according to the local consistency which is essentially the consistency of matches of the adjacent segments. During matching, special constraints are added so that the same scene segment does not match with more than one model segment.

Henderson and Samal [106], [107] proposed a method for shape analysis using multiple semantic constraints. The approach is based upon discrete relaxation using both local and global constraints. Medioni and Nevatia [108] used a discrete relaxation approach to match aerial images with map-based models (using a fixed scale) on the basis of linear features. Initial labels are set depending on the line segment orientations. Arc consistencies are effectively determined depending on the distances between the pairs of features in the image and the model. An interpretation is given once a certain percentage of matches is found.

Grimson [109] developed a scheme for recognition of occluded industrial parts using linear and circular features which is similar to consistent labeling or discrete

relaxation labeling technique. In this technique, initial matching is performed with the help of generalized Hough transform technique. Then a constrained search in the interpretation tree is performed with these initial labelings. Recently, another method has been proposed by Umeyama [110] (which is similar to that of Grimson) in order to take care of the flexible objects, i.e., objects having movable subparts (e.g., scissor). Here, the points on the object boundaries are parametrically modeled, and constrained optimization tree search is performed for finding out the best match for some suitable parameter values.

Other techniques

Apart from using the techniques mentioned before, several other algorithms have been developed for 2D object recognition. Price [111] used a boundary correlation approach in which linear sequence of boundary segments are used to represent the object contours. The compatibility between the model and scene segments is found by comparing the lengths, intersegment angles etc., and the orientation differences are stored in a disparity array. The orientation difference in the longest sequence of match is used to compute the transformation from model to scene. This transformation is then used to find the final set of matched segments.

The method developed by Gorman and Mitchell [112] considers polygonal approximation of objects. Fourier coefficients are computed for each triplet (considering three consecutive vertices). A norm squared distance between the Fourier coefficients for each segment in the model and scene is measured, and an intersegment distance table is formed. A minimum distance path in the table that has resulted from a diagonal transition is considered to be the desired match between the model and scene object. Ansari and Delp [113] used a similar technique where a new local shape measure, namely sphericity, was proposed as the similarity measure.

◇

Some Remarks : The relative merits and demerits of the abovementioned techniques are provided in [4]. Although the algorithms have been classified according to the techniques they have adopted, some of these algorithms use more than one technique to enhance the recognition capability of the system. For example, in the scheme developed by Grimson [109], the initial choice of match between the

model and scene features was guided by generalized Hough transform technique. The match quality was then further improved by using constraint satisfaction tree search.

From the discussion, it becomes apparent that the task of recognition of objects is a kind of optimization (constraint satisfaction) problem. A suitable solution can be approached by heuristic search, relaxation labeling, finding out cliques from association graphs, or finding the homomorphism between relational graphs.

Note that, generalized Hough transform may be used to obtain some initial guess about the presence of objects. But in order to get the desirable matching performance, the most prominent peak in the Hough space needs to be detected. This, in turn, is a nontrivial task. Again, in the implementation of GHT, space requirement is very high. However, one definite advantage of GHT is that it can be directly implemented on a parallel machine. Also, the saliencies (degrees of importance) of different feature-object pairs can be effectively used in GHT.

1.3 Overview of Connectionist Approaches

We have presented a brief overview of classical algorithms dealing with the tasks of edge/line linking and object recognition. Apart from classical approaches, there is another approach, namely, connectionist approach (which has recently drawn the attention of researchers) for the development of computational strategies for these tasks. As mentioned before, connectionist models do not necessarily emulate animate vision, but they provide a structured computational framework with robustness, capability of learning (adaptivity and generalization), and a scope for massive parallelism. Moreover, it might be easier to take into account the psychological and neurological findings within the connectionist framework than with classical algorithmic models.

In this section, we are going to present a brief overview of some basic neural networks followed by some models for mixed category perception. Then applications of neural networks to the tasks of edge/line linking and object recognition are discussed. It may be mentioned here that the literature on connectionist approaches to the tasks of edge/line linking and object recognition is not that rich as compared to its classical counterpart.

1.3.1 Neural Network Models

The research in the field of neurocomputing started with the emergence of a simple feed-forward model called *Perceptron* [114]. It has only two layers, namely input layer and output layer. The number of nodes in the input layer is the same as that of the dimensionality of the input feature vector. The number of nodes in the output layer is equal to the number of desired output classes. Perceptron essentially generates linear separation boundaries to separate out the linearly separable decision regions in the feature space. The weights of the links between the input and the output layer determine the nature of linear separation boundaries.

Later, Minsky and Papert [115] have shown that the perceptron model cannot be used for implementing a simple problem of exclusive OR, where a single linear separation boundary cannot separate the decision regions. To circumvent such a problem additional layers of neuron (hidden layers) in between the input and the output layers were introduced [116], [28]. This enables the network to generate nonlinear (or piecewise linear) separation boundaries. Since the network is an extension of perceptron model with several intermediate or hidden layers, it is called multilayered perceptron (MLP). A learning rule, namely back propagation was designed to train MLP under supervised mode. But for a given application, the selection of appropriate number of hidden layers and nodes for producing a desired performance is still an open problem. Several modifications of the backpropagation rule have been proposed to speed up the learning process and also to guide the network to adjust the number of hidden nodes [117]–[120].

In perceptron and multilayered perceptron, the nodes generally have some sigmoidal transfer functions or step transfer function. Another class of feed-forward networks [121] have emerged employing generalized radial basis functions in the hidden nodes. Radial basis functions produce maximum output for some given input, and the output decreases as the input activation increases or decreases. In the learning process of radial basis function (RBF) networks, sometimes the weights from input to hidden layer are separately trained with some a priori knowledge about the input data. However, backpropagation learning rules can also be used in RBF networks [30], [29].

Apart from feed-forward networks, there exist another class of networks, called recurrent networks. In feed-forward networks, information can flow only from the

input to the output layer. On the other hand, in the recurrent networks activations flow in both the directions. Recurrent back-propagation networks [122] can be used for time-series prediction.

Hopfield [123],[124] proposed a model for retrieving output patterns from noisy input patterns. Hopfield net is a recurrent network with all neurons connected to each other via weighted links. The neurons can produce either binary or continuous valued outputs. Each unit, in addition to receiving the inputs from other neurons, can receive external input or bias. With a given input pattern, the activation value of each neuron is iteratively updated depending on the given input bias and the weighted signals coming from other neurons. The output of the neurons reaches a stable state if the weights are symmetric (i.e., the weight w_{ij} of the link between any two neurons i and j is the same as that of the link between j and i (i.e., w_{ji})). Hopfield network is able to function as an associative memory where for a given partial information, the corresponding stored pattern is retrieved (auto-associative memory). Hopfield model with continuous state dynamics has also been employed to obtain suboptimal solutions to constraint satisfaction problems [32], [33]. Kosko [125], [126] developed a network for heteroassociative memories where associations between pairs of binary patterns are formed. Simpson [127] designed a higher-order intraconnected associative memory network where relationships between autocorrelators and heterocorrelators have been studied. The higher order associative memory incorporates multiplicative connections between nodes.

Stochastic network models were also developed in the trend of research on neurocomputing. Hinton et al. [128] developed a learning algorithm for Boltzmann machine (which is a conceptual amalgamation of simulated annealing with Hopfield type of models) for retrieving patterns from partial information.

The networks mentioned so far deal with classification and/or content addressability. The content addressability is the capability of retrieving the total information from a clue of partial information. It is seen that autoassociative and heteroassociative models are very good in performing the task of information retrieval. Multilayered perceptron and other similar models are very good in performing the task of classification. However, apart from content addressability, biological systems also exhibit self-organizing capability, where the entities are grouped into categories automatically without the help of any external teacher. Several investigations have been made on the self-organizing behavior of connectionist models.

There are several self-organizing networks employing the competitive learning process. The mostly referred models are Kohonen's self-organizing feature map [129], and adaptive resonance theory [130]. Kohonen's model can automatically produce topologically correct maps of the features of observable events. Adaptive resonance theory (ART) was developed by Carpenter and Grossberg for classifying the input patterns into different categories. Both supervised and unsupervised modes of learning are possible in the adaptive resonance theory of network models. ART was developed to form stable category codes for individual patterns employing a parallel search mechanism. A new 2/3 rule was used to match the input pattern with the top-down template pattern. A vigilance factor was defined to monitor the belongingness of an input pattern to a particular category, and this controls the process of formation of a new category. Both fast and slow learning processes have been incorporated in the self-organizing model.

The adaptive resonance theory was extended to categorize analog input patterns to form stable category codes [131]. The characteristics of such networks for additive and subtractive noise have also been studied [132], [133]. Later another model, namely, ARTMAP [134], was developed which uses two different ART modules (one of them acts as a predictive system) linked by an associative learning network. An internal controller ensures autonomous system operation in real time. ARTMAP acts like a self-organizing expert system that calibrates the selectivity of its hypotheses, based upon the predictive success. Recently, the ARTMAP has been extended to fuzzy ARTMAP [135] in order to handle the imprecise or vague information in input patterns.

Amari [136], [137] developed a self-organizing model for concept formation and its orthogonal and covariance learning techniques. Amari and Maginu [138] formulated the statistical neurodynamics of the associative memory. Anderson et al.[139] have developed the brain-state-in-a-box model to produce the association between the input and the output patterns and applied it for categorical perception.. They also discussed on probability learning techniques in this model. Chua and Yang [140] have proposed a cellular neural network model for performing several image processing tasks.

It may be mentioned here that the abovementioned networks dealing with problems like classification, self-organization, and content-addressability accept patterns from a single category at a time while performing these operations. But in many situations, it may be necessary to perceive mixed categories (i.e., more

than one category overlapping each other) simultaneously. Such cases may arise in the problem of recognizing multiple objects in a scene at a time or perceiving music coming from more than one source or even in prediction of disorders from the symptoms of a patient.

In literature, there exist several investigations on mixed category recognition. Peng and Reggia [141] developed a model for prediction of multiple disorders for a given set of manifestations. Here, the weights of the links were preassigned based on the probability of co-occurrence of manifestations and disorders. Later Cho and Reggia [142], [143] developed a learning scheme for automatically assigning these weights through competition and cooperation process. The competitive and cooperative process was mathematically formulated and an error backpropagation learning rule was derived. However, the learning rule operates in supervised mode only. Marshall [144]–[146] developed a scheme for mixed category recognition where the competition process was confined within the nodes of similar nature, i.e. getting activations from inputs which have sufficient overlap. Cohen and Grossberg [147], [148] used the concept of masking field for the recognition of mixed categories. The use of masking field enables the system to find out embedded patterns. It also allows multiple non-overlapping patterns to be classified simultaneously. Later, Nigrin developed a neural network model, namely, SONNET [149]–[152] which is able to self-organize in the presence of mixed categories and form stable codes. A concept of competition between the links for associating a category with a feature was introduced.

1.3.2 Edge/Line Linking

The task of line or edge linking can be dealt with *neural networks* or *connectionist architectures* which are often considered to be suitable for cooperative processing based on neighborhood information as needed for the task of edge/line linking. However, as mentioned before, the literature on connectionist approaches for this task is not much rich as compared to its classical counterpart. Some of the connectionist approaches dealing with the problems related to edge/line linking are described below.

Grossberg and Mingolla [153] have formulated a theory based on neural dynamics of visual processing to analyze real and illusory contour formation, contour

and brightness interactions, and filling-in of disconnected illuminants. Two different processes namely, boundary contour process and feature contour process are considered. The concept of brightness perception and perception of features and boundaries have also been discussed in this context [154]. The boundary contour process and feature contour process are based on short-range competitive and long-range cooperative processes. For boundary contour process several orientation sensitive masks are used, which respond to the amount and the orientation of the contrast but not to the direction of contrast. The boundary completion is performed by the long-range oriented cooperative process. For feature contour process, several masks sensitive to the direction (not to the orientation) and amount of contrast are used. Diffusive filling-in process is used for the feature contour process. An approach based on these processes has been reported in [155] for establishing neural dynamics theory for perceptual grouping phenomenon. This theory explains the perceptual grouping phenomenon by considering the interactions between boundary contour system, feature contour system and object recognition system. Feedback between the competitive and cooperative stages synthesizes a global context sensitive segmentation with many possible groupings of feature elements.

Linsker [156],[157] analyzed the underlying theory of self-organization and applied it to model the development of early visual processes in retina. He has shown the emergent properties of orientation sensitivity through cooperative and competitive processes. An unsupervised learning scheme similar to the Hebbian learning rule has been applied here. It has been shown that the cooperative and competitive processes enable the system to grow orientation selective columns, i.e., self-organize into banded patterns of cells of similar orientation [158]. This study provides the very basic levels of tasks in the process of early vision.

Zucker [71] tried to formulate a biologically plausible theory for linking broken edge and line segments. A concept of diffusion according to the direction of the edge or line points has been used to form an edge cover in order to guide the linking process. The process of diffusion according to edge/line orientations has also been analyzed theoretically.

It is to be noted here that several theories on neural dynamics have been proposed to explain the phenomenon of edge/line detection and linking, and the related visual processes in the retinal cortex. However, any structured neural network model which is able to perform the task of edge/line linking has not yet been

reported. Although a dynamical model for early vision based on neural network has been proposed by Bengtsson [159], it does not deal particularly with the task of edge/line linking. Also, the multipurpose neural processor, proposed by Knopf and Gupta [160] deal with several low-level processing tasks related to machine vision, but the specific problem of edge/line linking has not been dealt with.

1.3.3 Object Recognition

Connectionist approaches for visual pattern recognition can be broadly classified into two categories. In one approach, the features are extracted using classical methods, and then these features are used for the recognition task with one of the conventional networks like Hopfield, Kohonen, or multilayered perceptron. In the other approach, some new application specific models are built up which take the binary or graylevel images and recognize the visual patterns. The process of feature extraction gets embedded into the models. However, in many cases the new models also incorporate the principles of the conventional neural networks.

Hopfield model has been used in many object recognition algorithms by posing the task as an optimization problem (note that the capability of Hopfield model to deal with optimization problems was first shown in [32]). Here, an energy function is defined for the network such that the minima of the function corresponds to optimal match between the image features and the model features. The dynamics of the network is set in such a way that the network settles to a local minima of its energy function through iterative updating of the states of its units, and the state vector corresponding to the local minima provides a suboptimal match between the image and model features. Despite the fact that the Hopfield nets are not guaranteed to provide the optimal solutions, rapid computational capability of the network provides an important mechanism for tackling the computational complexity involved in dealing with object recognition problems. However, a limitation of using Hopfield model is that the importance levels of different feature-object pairs cannot be automatically associated, i.e., the weights cannot be learned. Rather, these weights are preassigned before any matching task is performed. Moreover, the problem of mixed category perception has not been dealt with this model.

Nasrabadi and Li [161], [162] developed a scheme for object recognition with the help of a Hopfield model where the polygonal approximations of 2D objects (both

the model and scene objects) are represented as graphs. A Hopfield network with two-dimensional array of neurons (number of rows represents the number of scene features and number of columns represents the number of model features) is used to match the model graphs (one at a time) with the scene graph. The best matching subgraphs correspond to the objects present in the scene. However, this particular approach deals with only symmetric relations between features, i.e., matching between only undirected graphs is performed. Features using 'sphericity' property of objects [163] have also been used to form the graphs in this scheme.

The task of 3D object recognition using Hopfield network was performed by Lin et al. [164]. Here, the network is used for matching prototype objects with the stored model objects in two stages : feature-wise in the first stage and surface-wise in the second stage. Here also, only the symmetric relations between features have been considered.

Multilayered perceptron has also been used for object recognition by posing the task as a classification (supervised) problem. Here, feature vectors are derived for each object model and an MLP is trained with these feature vectors under supervised mode. The trained network then accepts the features extracted from the image and classify these features accordingly. Note that, the problem of simultaneously recognizing more than one object cannot be dealt with this approach. This is due to the fact that the feature set extracted from an image of more than one object (occluding each other) is essentially an overlap of the feature sets corresponding to different objects. As a result, the input feature vector falls widely apart from the true decision regions formed by the learned parameters.

In the method proposed by Tsang et al. [165] for object recognition using MLP, the features were extracted as follows. The entire range of angles ($0^\circ - 360^\circ$) was divided into a number of slots of equal size, and each slot has been treated as a feature. The feature value is determined by the number of corners whose angles fall within the corresponding slot. Besides this, another feature vector was formed in a similar way, where a feature value is equal to the number of arc changes (between two successive corners) by an angle within the corresponding slot. The union of these two feature vectors was then presented to the network for learning and classification.

There is another system developed by Tsang and Yuen [166] for the recognition of partially occluded objects. The system consists of three stages. In the first stage,

object boundaries are detected and then some salient features are extracted from a feature codebook. In the second stage, presence of some possible objects are hypothesized using a nonlinear elastic matching technique. Finally, the presence of each possible object is verified using the corresponding salient features with the help of an MLP. However, in this system, MLP is used only in the final stage, and hypotheses about the presence of the objects are made in the nonlinear elastic matching process itself.

Bebis and Papadourakis [167] an MLP based recognition system where some invariant features were extracted by using the cumulative angular and curvature representations of the object boundaries. They also investigated the effectiveness of Kohonen model with the same features for performing this task. Although Kohonen model has been found to self-organize properly with the feature sets derived from single objects, it would face the same difficulty like MLP in the case of mixed category perception.

Adaptive resonance theory has also been used for shift and orientation invariant visual pattern recognition. Srinivasa et al. [168] used an invariance network in conjunction with an ART1 module for this purpose. Here, different rotations (0^o, 90^o, 180^o, 270^o) and shifts (in discrete steps) are explicitly coded in the invariance network which cooperatively interacts with the output layer of ART1 module. Although shift and rotation invariance were achieved for simple form of visual patterns, real-life objects were not considered as input. Moreover, ART1 module also has the same difficulty like MLP, in the perception of mixed categories.

Note that, the connectionist methods described so far, exploit the capability of some standard neural networks like Hopfield model, MLP, Kohonen model, or ART1 for performing the task of object recognition. It is further to be mentioned that these approaches consider only the problem of object identification, and do not deal with the pose estimation or localization problem with the network models. There is another kind of approach (as mentioned before) where some application-specific models are developed for this task. Some of these models also deal with the problem of localization or pose estimation. These are described below.

. Cognitron [169] was developed to categorize input patterns by employing *competitive learning* techniques, but it fails to recognize patterns suffering from positional shifts. To incorporate the property of position and scale invariance, a multilayered model, namely, neocognitron [170]–[173] was developed. The model uses two kinds

of cells, namely, S and C cells arranged in alternate layers. S-cells extract the features at various stages, while the C-cells ensure position and scale invariance. The shift invariance is achieved by tolerating the positional shift a little in each layer of C-cells at a time. The network was designed to possess the self-organizing capability where the connections between two maximally activated cells (within a predefined vicinity) are reinforced. Complex transfer functions were used for the cells in this model. The main contribution of this model is that it introduces the concept of adjusting the positional shift or deformations incrementally within a group of cells in each layer.

The model was extended to incorporate the property of selective attention [174]–[177] by using feedback pathways from the output layer to the input layer. With the help of feedback signals, the network is able to attend a particular pattern even when a mixture of more than one pattern is presented to the network. It was tested for numeral recognition problem. The effectiveness of neocognitron has also been tested for character recognition problem [178], [179]. Recently, the model has been extended to segment and recognize cursive scripts [180] with the help of a "search controller" which assists to select a particular search area.

The power of neocognitron and its variation lies in the fact that the models are capable of tolerating error due to positional shift, scale change or deformation. But the model is not capable of recognizing more than one object simultaneously. Whenever a mixture of patterns is provided to the network, it always recognizes one (most prominent one) of them. Moreover, the model does not consider the structural relationships between the features and the objects. The model is also incapable of tolerating the rotational variance. Recently, a variation of the model has been used to achieve rotation invariant object recognition [181]. Minnix et al. [182] used the underlying concept of neocognitron [175], and developed a self-organizing model for translation-invariant object recognition with better ease.

Hinton [183], [184] extended the idea of generalized Hough transform to *dynamic Hough transform*. In DHT, the problem of scale, position and orientation invariance are dealt with by considering a reference frame for the object and describing the constituent features with respect to the object. Using cooperative and competitive computation, the reference frames of the object models are determined. In DHT model, the relative importance of the features are not considered. A concept of part-whole hierarchy was provided by Hinton [185] for efficient storage of and effective computation on the object models. Although the concept has been

illustrated with examples of letters, it is also applicable to recognition of other objects.

Mozer [186] developed a word perception model (MORSEL) incorporating the selective attention mechanism. The model is able to learn and recognize multiple letters along with their relative positions. The phenomenon of neglect dyslexia in psychological patients has also been explained with the help of this model [187]. However, the performance of the model is dependent on the orientation of the objects and therefore it may be difficult to apply it for industrial object recognition. Again, like neocognitron, no structural relationship between the features and the objects was considered and the performance of the network was also not tested on industrial objects.

The structural relationship between features and objects was considered in the connectionist system (TRAFFIC) proposed by Zemel [188], [189]. The model uses the knowledge of transformation from the feature reference frame to the object reference frame. Here also, it is not possible to recognize multiple objects simultaneously.

There exist several other relevant investigations which need to be mentioned in the context of object recognition using neural networks. Poggio and Edelman [190]–[193] developed a three-layered network for producing a standard viewpoint representation of the 3D objects from any given viewpoint. The input layer consists of the coordinate values of the features in the image plane. The hidden layer consists of several nodes with generalized radial basis functions (Gaussian in nature). The centers of the transfer functions code the input viewpoint representation from the coordinate values. The output nodes have linear gain functions. The activations in the output layer represent the coordinates of the features in the standard reference frame. The performance of the model was tested with wire-frame models of 3D objects.

Wechsler and Zimmerman [194] considered distributed associative memory (matrix associative memory) for recognizing 2D objects with rotational and scale invariance. They used a conformal complex-log mapping to transform the rotation into an additive constraint. From the transformed image the scale and rotations are estimated, and retrieval of the corresponding stored model is performed with the help of distributed associative memory. The model was further extended [195] in conjunction with a re-projection system to tolerate some nonmetric transforma-

tions due to change in camera positions (the nonmetric transformations include foreshortening, self-occlusion etc.).

Hirai and Tsukui [196] presented a scheme for position independent pattern matching with neural networks. The proposed model contains three layers, namely, pattern matching layer, minimum distance layer and a recognition layer. It uses a supervised mode of learning. The effectiveness of the network is tested with very simple synthetic patterns. However, the model does not deal with the problems of rotation invariance and simultaneous recognition of multiple objects.

Chan [197] developed a model for object recognition with translation invariance. It uses one hypernetwork which generates the target response along with some secondary response values. The next stage consists of a confirmatory network which analyses the target response and the secondary response values to decide which particular object has appeared in the scene. The model achieves translational invariance by replicating the weights of the links over different positions. But it is not able to recognize orientationally invariant objects and also the behavior for mixed objects has not been studied.

There exist several other recently developed connectionist systems for object recognition. Li and Nasrabadi [198] used a cascade of restricted coulomb energy (RCE) network to classify between the objects. The boundary/shape information of the objects was coded into feature vectors of fixed length, and then cascaded RCE networks were used to progressively resolve the overlapping complex decision regions. However, in this technique, the problems of mixed object recognition and localization have not been considered. Higher-order neural networks employing multiplicative (pi-) connections had also been employed for position, scale and rotation invariant object recognition [199]. However, in this technique, the problem of localization within the connectionist framework has also not been dealt with.

Feldman presented the principles of connectionist computing, principle of stable coalition formation and winner-take-all network in [26], [27], [200]. Note that, the neural networks mainly involve the study of emergence of activations of the cells and weights of the links on the basis of mathematical modeling. On the other hand, AI based techniques mostly deal with inference representation. For solving the recognition problem in visual domain, representation of knowledge in spatial domain needs to be considered. This leads to the concept of structured connectionist models for developing visual recognition system. Sabbah [201] also

used structured connectionist framework for origami object recognition.

◇

Some Remarks : It is apparent from the aforesaid discussion that a few theoretical frameworks explaining the phenomenon of edge/line detection and linking, and the related low-level processes in the retinal cortex have been provided, but no structured connectionist model for performing these tasks on real-life images has been reported. Development of such a model is therefore necessary as a first step in designing a stand-alone connectionist system for object recognition.

In the algorithms for object recognition using basic models like Hopfield, MLP, or Kohonen, the objects are found to be identified but not localized, i.e., pose estimation problem is not solved with the network. Although Hopfield model has been used for multiple object recognition by matching model graphs one at a time with the scene graph, the problem of mixed category perception (i.e., simultaneous recognition of multiple objects) has not been dealt with. Moreover, the degree of importance of the features cannot be learned. On the other hand, feature importances can be learned with MLP or Kohonen model, but the problem of mixed category perception cannot be handled.

In the case of application-specific models like Neocognitron, MORSEL, TRAFFIC etc., pose estimation problem is taken care of by some of them using selective attention mechanism. But simultaneous recognition of multiple objects (except MORSEL) is not considered. Although psychological studies reveal the fact that the object identification and localization occur in two separate parts of the visual cortex [202], [203], this particular finding has not yet been incorporated in any of the connectionist models developed, so far, for object recognition.

AI techniques (classical approaches) have proved to be better in their performance of recognizing industrial objects while the connectionist models have more generality. It has been claimed that there is a dilemma between generality and performance [5] of any system. Thus it seems that a middle path needs to be followed where object recognition system should be designed within a structured connectionist framework (with richer knowledge representation scheme) incorporating some of the psychological findings for better architectural efficacy, and the system should be able to learn and recognize (both identify and localize) multiple objects simul-

taneously.

1.4 Scope of the Thesis

In this thesis, we have presented some results of investigation demonstrating the effectiveness of connectionist approaches for certain tasks related to object recognition, namely, edge/line linking and feature set interpretation. For performing edge/line linking in graylevel images, two new models are developed which employ cooperative and competitive processes within the neighborhoods of neurons.

The study on feature set interpretation has two parts. In the first part, a basic network, namely Hopfield model, has been used to develop an algorithm for matching relational descriptions of prototype structures with the candidate structures for object recognition.

The second part, on the other hand, concentrates on the development of some new application-specific models for learning and simultaneous recognition of multiple objects. For this purpose, a model (X-tron) is initially designed for the perception of mixed categories (under both supervised and unsupervised modes). A method is also developed in this regard, based on the principle of X-tron, for simultaneous extraction of multiple linear structures in an image. Finally, this X-tron is employed to build a connectionist system (PsyCOP) which has two separate channels for identification and localization of the objects. Structural relationships between the features and the objects are also considered in the design of PsyCOP. The effectiveness of the proposed models has been demonstrated on both synthetic and real life data.

The rest of the thesis is organized under the following chapter headings.

1.4.1 Edge/Line Linking [204,205]

Two new connectionist models dealing with the task of edge/line linking in graylevel images have been presented in **Chapter 2**. The processing elements (PE) are arranged in the form of a two dimensional lattice in both the networks. These networks accept the strengths and the corresponding directions of the fragmented

edges (or lines) as the input. The state of each processing element is updated by the activations received from the neighboring processing elements. In one of the models, each neuron interacts with its eight neighbors, while in the other model, each neuron interacts over a larger neighborhood. After convergence, the output of the neurons represent the linked edge (or line) segments in the image. The first model directly produces the linked line segments, while the second model produces a diffused edge cover. The linked edge segments are obtained by finding out the spine of the diffused edge cover. Although the networks are designed for edge/line linking, they may be employed with suitable modifications for performing other low-level operations which require local cooperative and competitive processing.

1.4.2 Structure Matching using Hopfield Network [206]

The problem of matching structural descriptions of objects (descriptions of parts and spatial relations between them) in a connectionist framework has been dealt with in **Chapter 3** of the thesis. A Hopfield net based matching technique is developed which is general enough to take care of partial mismatch between the individual parts and spatial constraints between these parts. In addition, a transformation of the shape descriptions has been suggested with which shape descriptions containing asymmetric spatial constraints between the parts can be matched using symmetric interconnection weights for the Hopfield net. Note that, unlike the earlier attempts [161], [163], [164] which use only symmetric relations, we have considered both symmetric and asymmetric spatial relations. The effectiveness of the method has been demonstrated for recognition of handtools and symbols.

1.4.3 X-tron : Supervised Mixed Category Perception [207,208]

The recognition scheme, described in Chapter 3, employing the basic network (Hopfield) cannot take care of learning the degree of importance of different input features and is also incapable of simultaneous recognition of multiple objects. A three-layered connectionist model (X-tron) with supervised learning scheme has been presented in **Chapter 4** for simultaneous recognition of multiple categories/objects. The input layer accepts numerical values representing the degree of confidence

about the presence of the features. The output layer produces the degree of confidence about the presence of categories/objects, and the hidden layer corresponds to the feature-object associations. The learning rules for X-tron are defined in such a way that the weights of the links asymptotically reach some predefined probabilistic measures. During learning, the network automatically adjusts the number of nodes in the hidden layer. A theoretical investigation has also been performed on the expected number of nodes in the network. The performance of the model has been experimentally demonstrated on overlapped binary strings and visual patterns.

1.4.4 X-tron : Unsupervised Mixed Category Perception [209–211]

X-tron, described in Chapter 4, is extended for categorization or self-organization (*even in the presence of multiple or mixed patterns*) in **Chapter 5**. Here also, during self-organization, the network automatically adjusts the number of nodes in the hidden and output layers, depending on the complexity or nature of overlap between the patterns. An ambiguity measure is given based on how well the features are being interpreted by the network. From the ambiguity measure a certainty factor about the decision of the network is derived. A *vigilance threshold* is used to decide whether the decision of the network is correct or not. Functionally the network consists of two parts, one of them categorizes the incoming patterns, the other monitors the performance of categorization. The characteristics of the model has been demonstrated experimentally on both one-dimensional binary strings and image patterns. The effect of noise on the categorization performance has also been investigated both theoretically and experimentally.

1.4.5 Primitive Extraction, Structural Learning and Recognition [212–216]

In **Chapter 6**, a three-layered network is developed using the principle of X-tron for simultaneous extraction of multiple linear structures (primitives) from an image. The network basically implements Hough transform where multiple peaks (corresponding to the linear structures) are detected simultaneously using iterative

verification scheme. The network also uses the method of line linking, as described in Chapter 2, for smoothing out the line directions in the hidden layer.

This network is then cascaded with a three-layered MLP so that the resulting six-layered network can be used for learning and recognition of structures. The effectiveness of this system has been demonstrated on handwritten Bengali character recognition.

1.4.6 PsyCOP : Object Identification and Localization [217,218]

Chapter 7 presents a connectionist system (PsyCOP) which is built using the principle of X-tron, for learning and simultaneous recognition of multiple flat industrial objects like hammer, spanner, plier etc. The system is designed by integrating the generalized Hough transform technique with the psychological finding – identification and localization occur in two separate regions in visual cortex. The system also uses the mechanism of selective attention for initial hypotheses generation. Several new structured connectionist modules are also introduced in this regard. Special two stage training paradigm has been developed for learning the structural relationships between the features and objects, and learning the degrees of importance of the features with respect to the objects. The principle used for learning the importance values of the features is the same as that used by X-tron in Chapter 4. The noise tolerance capability has been theoretically investigated, and preventive measures are taken against the instances of false alarming. The performance of the system has been tested on real life data both for single and mixed (overlapped) instances of object categories.

1.4.7 Conclusions and Scope of Further Research

Chapter 8 concludes over the thesis and presents some scope of further investigation.

Chapter 2

Edge and Line Linking

2.1 Introduction

We mentioned in Chapter 1 (Section 1.1 and Section 1.2.1) that edge and/or line detection and linking is an important stage in the the feature extraction process. An edge, in an intensity image, is a border between two adjacent extensive regions where graylevel changes abruptly. On the other hand, a line (straight or curved) is found in an intensity image if graylevel along a thin strip is approximately uniform and distinctly different from its neighboring regions (a line is often referred to as a roof edge). Neural networks, being a robust, massively parallel computational framework, and suitable for local processing may prove to be a useful paradigm for dealing with such tasks. Several theories, based on neurodynamical computation, explaining the phenomenon of edge/line linking are briefly described in Section 1.3.2.

In this chapter, we present two new connectionist models dealing with the task of line and edge linking in graylevel images [204], [205]. The first one is structurally similar to the cellular network models [140] where each processing element (PE) receives input only from its eight neighborhood. However, the processing elements used here are functionally more versatile than those used in the cellular network models. The second model, on the other hand, depends on the edge induction process where each PE interacts with other PEs over larger neighborhood.

The rest of the chapter is organized as follows. The underlying concept of edge/line linking is presented in Section 2.2. Section 2.3 describes a new model for linking line segments considering the interactions between the processing elements over eight neighborhood. Two different updating rules are described in this section. Section 2.4 presents another network for edge linking based on the concept of edge induction, where the processing elements interact over larger neighborhoods. The experimental results on graylevel images are presented in Section 2.5. The relative merits and demerits of the models are described in Section 2.6.

2.2 Overall Methodology

An edge has a step- (or ramp-) like intensity profile. This suggests the simplest and most popular edge detection technique : detection of local maxima of the first spatial derivative (gradient) of the intensity values. Similarly, the description of a line indicates that it has a spike- (or roof-) like intensity profile, i.e., a step- (or ramp-) like gradient profile (sometimes a line is referred as a roof edge). Consequently, line can be detected as the local maxima of the second directional derivative of the intensity values. In discrete domain, only sixteen different types of line segments can appear (Fig. 2.1) over a 3×3 neighborhood. For each kind of line segment a particular type of template can be designed. When the template is convolved with the input image, the output indicates the strength of the line, and the template type indicates the direction of the line. In linking, generally, a set of edge (or line) segments which are believed to belong to the same curve, are linked to form the continuous line or edge in the image. But, finding out the subset of edge (or line) segments which do belong to the same curve is the basic problem. If the process of reinforcement of the edge/line points based on the neighborhood information is considered then edge (or line) segments may be iteratively linked with the neighborhood segments without any prior selection of appropriate subset. In the present approach of edge or line linking two assumptions are made which are as follows :

i : True edge and line points on a curve in a scene follow some continuity pattern, whereas the noisy pixels do not follow any such continuity.

ii : The strengths of the true edge or line pixels are greater than those of the noisy pixels.

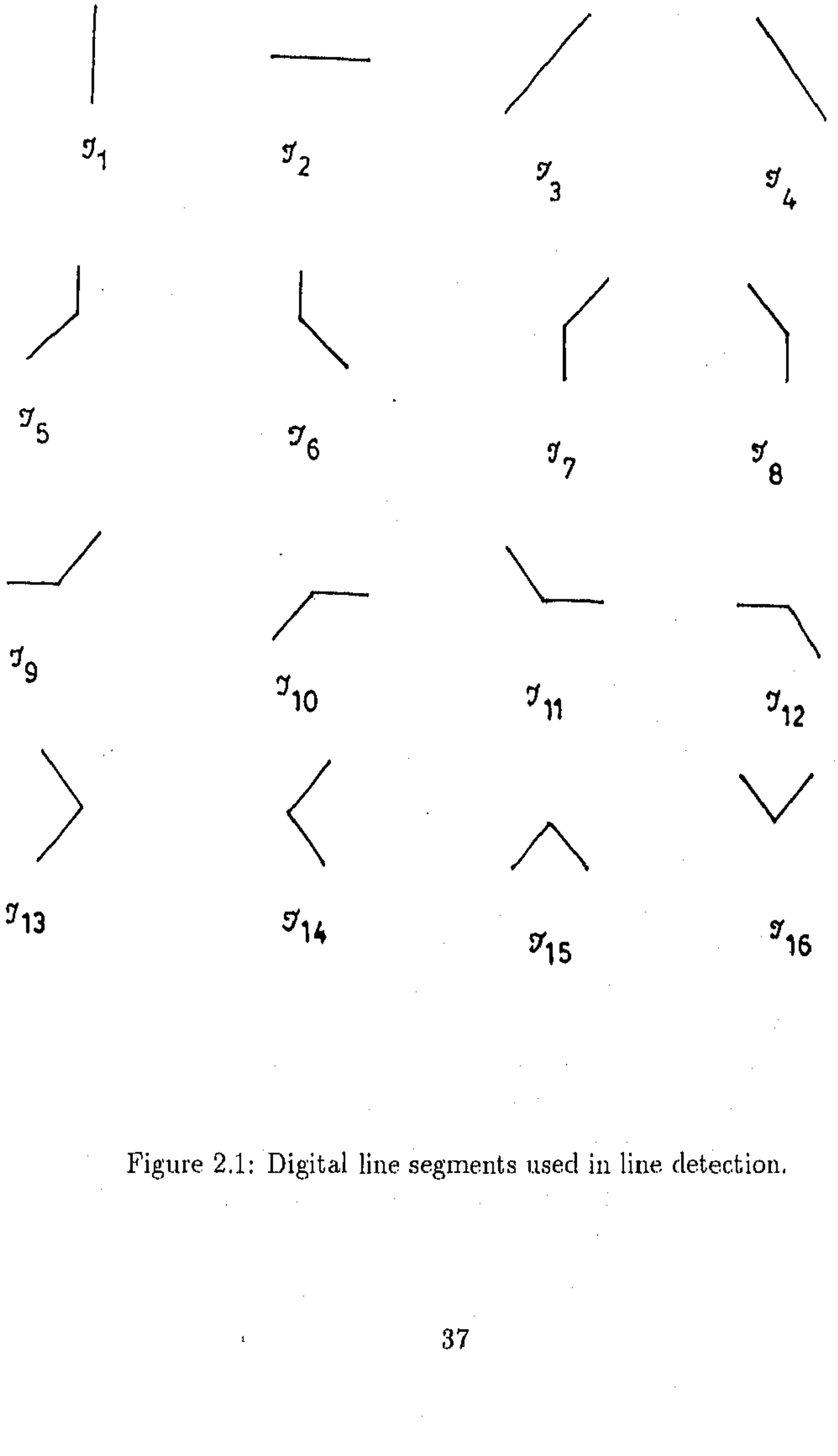


Figure 2.1: Digital line segments used in line detection.

Although the strength of some noisy pixels may be more than that of true edge or line pixels, the above assumption holds good for average strength of the edge/line pixels over the entire image [1].

In the process of reinforcement, an edge (or line) segment should reinforce the neighborhood segments depending on its direction. Suppose, image contains an edge segment APB, where P is the center pixel (Fig. 2.6). In that case maximum reinforcement should occur along the extensions of PA and PB. On the other hand, minimum (or zero) reinforcement should occur in the perpendicular direction of PA and PB. Again, the edge segment APB will be reinforced mostly by the segments close to the extensions of PA and PB having similar directions. If two neighboring edge segments reinforce each other then the edge strengths of the true edge pixels (which are recursively reinforced) would increase, and noisy edge pixels would retain their strengths. In order to stabilize the strength of the true edge (or line) pixels, and to remove noisy edge (or line) pixels, each pixel should inhibit its own edge (or line) strength.

The proposed connectionist models for edge (or line) linking in an image of $m \times n$ pixels, consist of $m \times n$ neurons (or processing element or PE), arranged in a two-dimensional array where each neuron corresponds to a single pixel. Each neuron is connected over a neighborhood, and each one has a negative self-feedback. The self-feedback helps in removing the noise in the scene during the process of iterative edge (or line) linking. The models accept the initial edge/line strength and the information for edge (or line) direction at each pixel (the initial edge/line strengths and directions are found by using some suitable convolution operator). The output of each neuron represents the edge (or line) strength after linking at that particular location, i.e., the output of the connectionist model represents the edge (or line) map of the image after linking of the segments.

Initially, the states of the neurons are clamped according to the input (the edge (or line) strengths and the directions) from the graylevel image. The state of the neurons are then updated with the neighborhood information. In the process of updating, the external input (i.e. input from the image) does not affect the states of the processing elements any more. The edge strengths and the directions extracted from the graylevel image are used only to clamp the state of the neurons at the very initial stage. During updating, the nature of the output of a neuron will depend

[1]The 'strength' of an edge/line point indicates the output of the convolution operator.

on the cumulative support received from the neighborhood. Therefore, how the neighborhood neurons are connected will play a key role in the linking process.

2.3 8-Neighborhood Connection

2.3.1 Line Detection Operator

As stated before, line strength at any pixel can be computed by matching the templates representing the second directional derivative operator. The strength and corresponding type are considered only when some conditions are satisfied. For example, for the first line segment ($\mathcal{T}_1$) in Fig. 2.1, let

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

and

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

where $I(x,y)$ is the grayvalue of the pixel (x,y). In this case, $(d_1+d_2)/2$ is taken as the line strength corresponding to the template type $\mathcal{T}_1$, if

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

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

2.3.2 Network Model

Each neuron in the *first model is connected* to its eight neighbors as shown in Fig. 2.2 and Fig. 2.3. The proposed structure has a similarity to that of

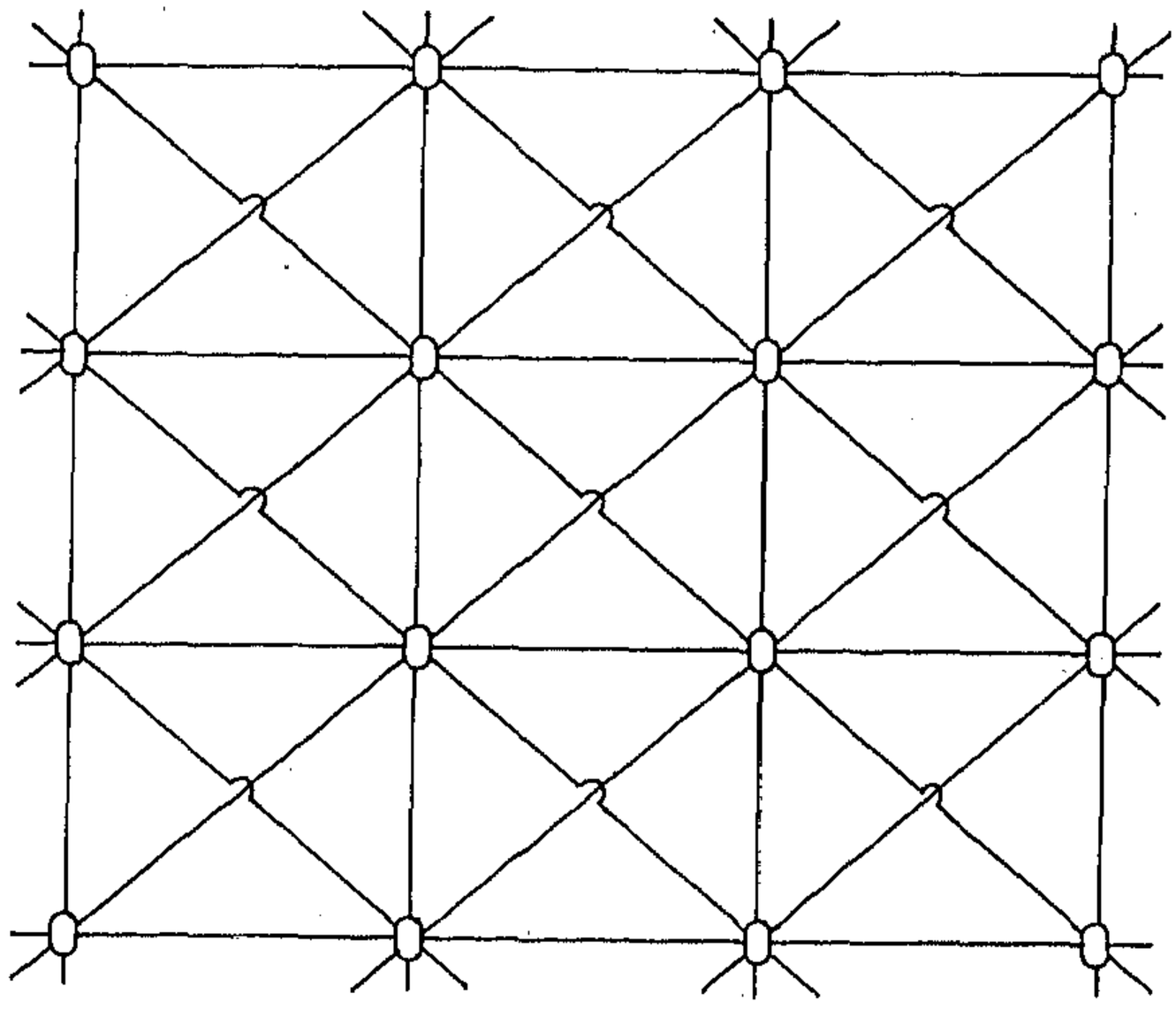


Figure 2.2: connections among the processing elements in a cellular connectionist model.

cellular model [140]. Each processing element in this connectionist model has some special functions apart from that in the conventional neural network models. In conventional neural network models [23], [24], [140], [123], [124], each neuron takes a number of inputs in the form of numerical values and produce a single output. In the present model, each neuron has a number of independent state elements which are updated by the external inputs depending on the type of the input. A part of the output of each neuron represents the activation level and the other represents the type of the output. Similarly, the input to a neuron has two parts, one of them is the numerical value representing the activation level and the other represents the type of activation. Note that, the type of activation (for input and output) actually depends on the template type matched at that location. Mathematically,

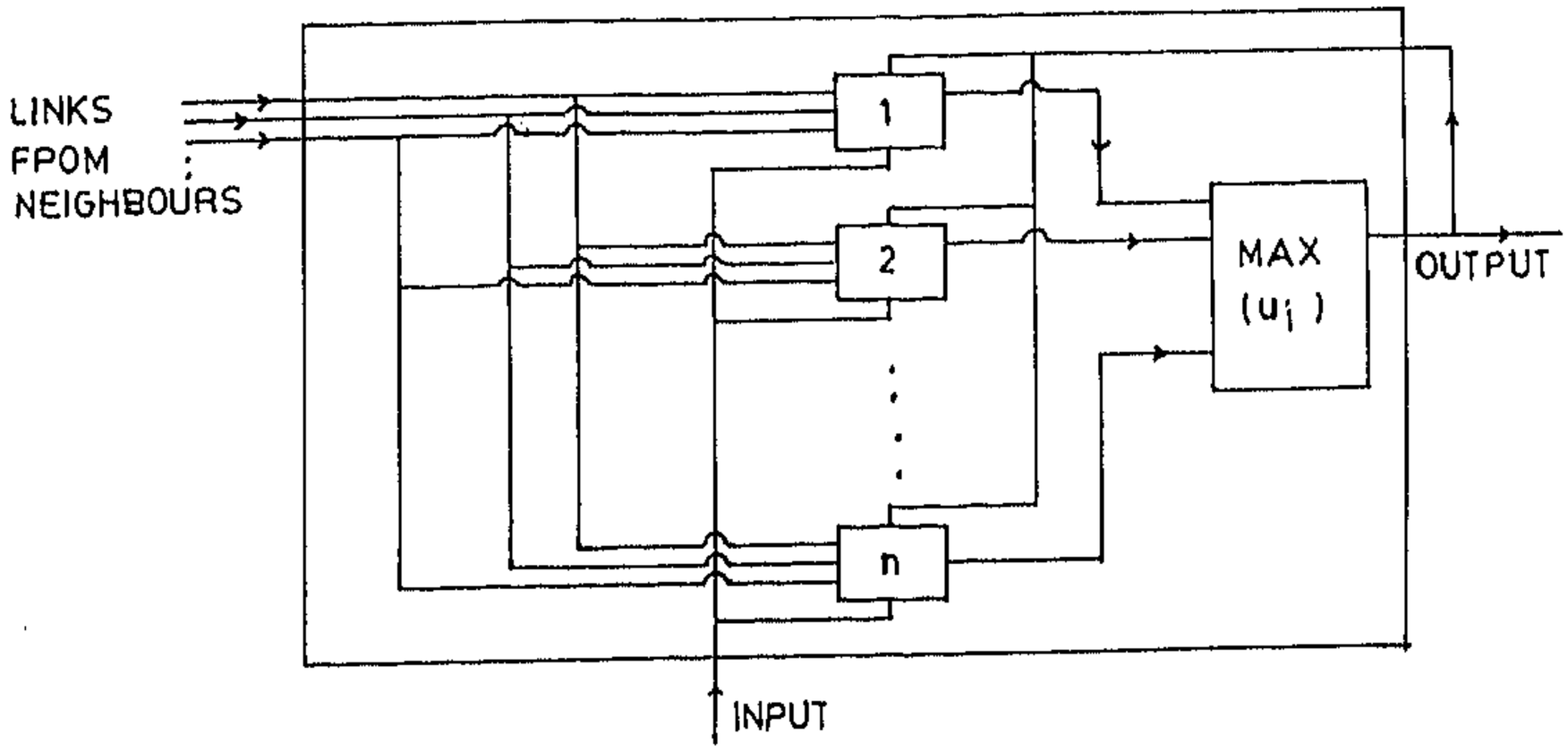


Figure 2.3: Schematic diagram of a processing element (PE) for line linking.

the output of a neuron can be written as

$$\mathbf{v}_i = [v_i, T_i^o],$$

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

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

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

$$\begin{aligned} v_i &= g(\max_{1 \le k \le n}(u_{ik})) \\ T_i^o &= T_{il} \quad \text{when } u_{il} \ge u_{ik} \quad \forall k, 1 \le k \le n. \end{aligned}$$

The transfer function $g(.)$ can be chosen as a sigmoid function [2].

[2]Note that although $g()$ is used as the transfer function of the neurons in this chapter, some other notation may be used in later discussion if necessary. However, if not mentioned explicitly, $g()$ will refer to the transfer functions of the neurons.

The states of each processing element is updated as

$$\frac{du_{ik}}{dt} = \sum_{i} \sum_{j \in N(i)} w_{ij} D_{ij}^{k} v_j \tag{2.1}$$

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

$$D_{ij}^{k} = \mathcal{N}_{ji}(T_j^o, T_{ik}) \tag{2.2}$$

where $\mathcal{N}$ is the neighborhood function.

The neighborhood function ($\mathcal{N}$) can be described in the following way. The sixteen different template types can be presented as an ordered pair (ζ_1, ζ_2) as explained below. The pair is defined by considering each line segment in Fig.2.1 as a collection of two segments joined at the center pixel. For example, the type at the output of j^{th} neuron can be defined as

$$T_j^o = (\zeta_{j1}^o, \zeta_{j2}^o).$$

ζ_{j1}^o and ζ_{j2}^o represent two labels within the eight neighborhood of the processor i, as shown in Fig.2.4. Similarly, the type of k^{th} state element of i^{th} neuron can be

	x_{i-1}	x_i	x_{i+1}
y_{i-1}	-4	-3	-2
y_i	-1	0	1
y_{i+1}	2	3	4

Figure 2.4: Relative positions of the processing elements in the network.

defined as

$$T_{ik} = (\zeta_{ik1}, \zeta_{ik2}),$$

where ζ_{ik1} and ζ_{ik2} also represent two labels within the eight neighborhood (Fig.2.4). For example, the template type $\mathcal{T}_1$ can be represented as $\mathcal{T}_1 = (-3, 3)$ or $(3, -3)$. Similarly, $\mathcal{T}_2$ can be represented as $\mathcal{T}_2 = (-1, 1)$ or $(1, -1)$. It is to be noted that any template (ζ_{i1}, ζ_{i2}) is same as (ζ_{i2}, ζ_{i1}). Let the positions of the processing elements i and j be (x_i, y_i) and (x_j, y_j) respectively in the lattice structure of the

network. Let a variable ζ be defined as

$$\zeta = (x_j - x_i) + 3(y_j - y_i)$$

In that case, if j is in the 8-neighborhood of i then ζ will take the same set of values as shown in Fig.2.4. The output type of PE **j** will affect the type of k^{th} state element of PE i when either of the following conditions hold :

$$\text{A} : \zeta = \zeta_{ik1} \bigwedge (\zeta^o_{j1} = -\zeta_{ik1} \bigvee \zeta^o_{j2} = -\zeta_{ik1})$$

$$\text{B} : \zeta = \zeta_{ik2} \bigwedge (\zeta^o_{j1} = -\zeta_{ik2} \bigvee \zeta^o_{j2} = -\zeta_{ik2})$$

For example, let the k^{th} state element of PE i be $\mathcal{T}_5$ or (2, -3). In that case, according to condition (a), PE j should be placed in such a way that $\zeta = 2$, i.e., PE j should be placed at the lower left corner of PE i, and T^o_j should be $(-2, *)$. Similarly, according to condition (b), PE should be placed in such a way that $\zeta = -3$, i.e., PE j should be just above PE i, and T^o_j should be $(3, *)$. If either condition (a) or condition (b) is satisfied then only PE j will affect the state element of PE i. Mathematically, it can be written as

$$\begin{aligned} \mathcal{N}_{ji}(T^o_j, \mathcal{T}_{ik}) \;=\; & \max\{1 - \left(|\zeta - \zeta_{ik1}| + |(\zeta^o_{j1} + \zeta_{ik1})(\zeta^o_{j2} + \zeta_{ik1})|\right) \\ & \left(|\zeta - \zeta_{ik2}| + |(\zeta^o_{j1} + \zeta_{ik2})(\zeta^o_{j2} + \zeta_{ik2})|\right), 0\} \end{aligned} \tag{2.3}$$

The updating rule described in Equn. 2.1 has been derived on the basis of continuity of the line (or edge) points. The image after detection of the line (or edge) points is mapped onto the array of processors, such that the output of each neuron represents the strength of the line (or edge) at the corresponding point. Ideally, after convergence of the network, the output of a neuron corresponding to a line (or edge) point should be unity and zero otherwise. On a continuous line there should be two line (or edge) points on the opposite sides of a point (except for the end points). Therefore, the neurons corresponding to a line (or edge) point will receive activations from the neighboring neurons. On the other hand, the noisy pixels in an image do not follow any continuous pattern. As a result, the neurons corresponding to the noisy pixels do not receive proper support from the neighbors, and consequently their activation levels remain unchanged. Therefore, the resultant effect is an enhancement of the line (or edge) pixels without any reduction of noise. To reduce the effect of noise, each neuron has a negative self- feedback. Usually, for a pixel on a continuous line, the activations received by the corresponding neuron

from the neighborhood is greater than the negative feedback. The updating rule with negative feedback can be written as

$$\frac{du_{ik}}{dt} = \sum_{j \in N(i)} w_{ij} D_{ij}^{k} v_j - w_s v_i. \tag{2.4}$$

The strength of the self-feedback (w_s) is the same for all neurons in the network.

2.3.3 Convergence of the Network

To prove the convergence and stability of the network, we have to show that both the output types and the output values converge and remain stable. First, let us prove the stability of the output types.

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

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

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

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

It can be shown that under the updating rule given by Equn. 2.4, the output of the network stabilizes.

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

$$E(t) = -\frac{1}{2}\sum_i \sum_{j \in N(i)} w_{ij} D_{ij} v_i v_j + \frac{1}{2}\sum_i w_s v_i^2 \tag{2.5}$$

where the term D_{ij} is given as

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

From Equn. 2.4, it is clear that the updating of the output activation can be written as

$$\frac{du_i}{dt} = \sum_{j \in N(i)} w_{ij} D_{ij} v_j - w_s v_i \tag{2.6}$$

where u_i represents the state value of the mostly activated state element of processing element i, i.e., $u_i = g^{-1}(v_i)$. The Equn. 2.6 is true because if some neuron j does not update the mostly activated state element of neuron i then the change in output of neuron i due to j will be zero.

From Equn. 2.5, it is clear that the energy function is bounded for some finite network structure. More over between two neighboring neurons,

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

In other words,

$$D_{ij} = D_{ji}. \tag{2.7}$$

Moreover, in the design of the network, w_{ij} was selected to be symmetric, i.e., $w_{ij} = w_{ji}$. Therefore, from Equn. 2.5

$$\frac{dE}{dt} = -\sum_i \left(\sum_{j \in N(i)} w_{ij} D_{ij} v_j - w_s v_i \right) \left(\frac{dv_i}{dt} \right) \tag{2.8}$$

Substituting the value of $\frac{du_i}{dt}$ from Equn. 2.6, the Equn. 2.8 becomes

$$\frac{dE}{dt} = -\sum_i g^{-1'}(v_i) \left(\frac{dv_i}{dt} \right)^2 \tag{2.9}$$

where $g^{-1'}$ is the 1st derivative of the inverse of $g(.)$ which is an increasing function, and as a result, $g^{-1'}$ is positive in $[0,1]$, and evidently, $(\frac{dv_i}{dt})^2$ is always nonnegative. Therefore,

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

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

2.3.4 Modified Updating

In the network, the output of a state element of a neuron changes if it gets support from even one of its neighboring PEs according to Equn. 2.4. In a continuous line, a line point (except the end points) would get support from two of its neighbors, depending on the direction of the line at that point. Again for noisy pixels the chance of getting support from both the neighbors simultaneously is much less than that from one neighbor. Therefore, to reduce the noise level in the output, the state vector of each processing element can also be updated considering supports from both the neighbors simultaneously. But, this process may degrade the continuity of the line to some extent.

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

$$\frac{du_{ip}}{dt} = w \sum_j \sum_{k \in N(i)} D^p_{ijk} \sqrt{v_j v_k} - w_s v_j \tag{2.10}$$

where the weights of interconnections in the network are considered to be uniform and equal to w. $D^p_{ijk} = D^p_{ij} \wedge D^p_{ik}$, i.e., the state element (u_{ip}, T_{ip}) is updated only if it gets support from both the PEs j and k in its neighborhood. In the process of updating, the output type is the type of the mostly activated state element. It can be shown as before (Section 2.3.2) that the output type of any processing element in the network always stabilizes, i.e., type of each PE can change only a finite number of times. The outputs of the PEs can be shown to converge in the modified updating process also.

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

$$\frac{dE}{dt} = -\sum_i \left(\frac{dv_i}{dt}\right)^2 g^{-1'}(v_i) \tag{2.11}$$

i.e.,

$$E(t) = -\int_0^t \frac{dE}{d\tau} d\tau \tag{2.12}$$

where τ is a dummy variable. If it can be shown that $E(t)$ is bounded for any value of t then the network can be claimed to converge. This is true because from Equn. 2.11, it is clear that

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

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

Proof : It is evident that

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

Again,

$$D^p_{ijk} = D^p_{ij} \bigwedge D^p_{ik}$$

Therefore,

$$D^p_{ijk}\sqrt{v_j v_k} \leq \frac{1}{2}(D^p_{ij} v_j + D^p_{ik} v_k) \tag{2.13}$$

The updating rule (Equn. 2.10) can be written as

$$\frac{du_{ip}}{dt} \leq w \sum_{j \in N(i)} D^p_{ij} v_j - w_s v_i \tag{2.14}$$

Let us assume

$$\left(\frac{du_{ip}}{dt}\right)_1 = w \sum_{j \in N(i)} D^p_{ij} v_j - w_s v_i \tag{2.15}$$

In other words,

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

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

$$\frac{dv_i}{dt} \leq \left(\frac{dv_i}{dt}\right)_1, \quad \forall i \tag{2.16}$$

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

$$v_j v_k \leq \sqrt{v_j v_k}$$

Let

$$\left(\frac{du_{ip}}{dt}\right)_2 = w \sum_j \sum_{k \in N(i)} D^p_{ijk} v_j v_k - w_s v_i \tag{2.17}$$

In that case,

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

As a consequence we have,

$$\left(\frac{dv_i}{dt}\right)_2 \leq \left(\frac{dv_i}{dt}\right), \quad \forall t \tag{2.18}$$

Therefore, from Equn 2.16 and 2.18.

$$\left(\frac{dv_i}{dt}\right)_2 \leq \left(\frac{dv_i}{dt}\right) \leq \left(\frac{dv_i}{dt}\right)_1 \tag{2.19}$$

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

$$E_1 = -\frac{1}{2}w \sum_i \sum_{j \in N(i)} D_{ij} v_i v_j + \frac{1}{2} \sum_i w_s v_i^2 \tag{2.20}$$

and

$$E_2 = -\frac{1}{2}w \sum_i \sum_j \sum_{k \in N(i)} D_{ijk} v_i v_j v_k + \frac{1}{2} \sum_i w_s v_i^2 \tag{2.21}$$

In these equations D_{ij} has the same meaning as before. $D_{ijk} = 1$, if $D^p_{ijk} = 1$, $D^q_{jik} = 1$ and $D^r_{kij} = 1$ where $v_i = g(u_{ip})$, $v_j = g(u_{jq})$ and $v_k = g(u_{kr})$. From Equn. 2.20 and 2.21, it is clear that

$$\frac{dE_1}{dt} = -\sum_i g^{-1'}(v_i) \left(\frac{dv_i}{dt}\right)_1^2 \tag{2.22}$$

and

$$\frac{dE_2}{dt} = -\sum_i g^{-1'}(v_i) \left(\frac{dv_i}{dt}\right)_2^2 \tag{2.23}$$

Therefore, from Equn. 2.16, 2.18, 2.22 and 2.23 it is evident that at any particular output state of the network,

$$\frac{dE_1}{dt} \leq \frac{dE}{dt} \leq \frac{dE_2}{dt} \tag{2.24}$$

Again, the energy functions E_1 and E_2 are bounded (from Equn. 2.20 and 2.21). Therefore, the energy function $E(t)$ is also bounded [219]. □

2.4 Larger Neighborhood

Now, let us consider the neighborhood size to be larger than the 8-neighborhood. If a larger neighborhood is used, it is difficult to determine which neighborhood pixels should reinforce each other depending only on the template types. Therefore, instead of using the template types the edge (or line) directions are stored here.

2.4.1 Overview of the Algorithm

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

Consider any edge point P (Fig.2.5) with the edge vector $\mathbf{e} = (e, \alpha)$, where e is the magnitude and α is the direction of the vector. N_P is a circular neighborhood of P (of radius R_P) over which P induces edge points. In rectangular coordinate system the edge vector can be written as (e_x, e_y), where e_x and e_y are the components along x- and y- axes respectively, i.e. $e_x = e\cos\alpha$ and $e_y = e\sin\alpha$. From the continuity principle (assumption (i) in Section 2.2), edge points are expected to appear along a line AB which is perpendicular to the edge direction. Therefore, P should try to induce edge vectors at every point on AB, and should not affect any point on CD. If the effect of P is considered to be smoothly distributed over its neighborhood then it can be stated that the effect of P on the points on PQ increases as Q approaches B and decreases as it approaches C. If P induces an edge at any point on PQ then it is reasonable to assume that the direction of the induced edge vector is perpendicular to PQ. Let the effect of P on any point P' in N_P be $\mathbf{e}' = (e', \phi')$, where e' and ϕ' are the magnitude and direction of the induced vector. Then

$$\phi' = \begin{cases} \pi/2 + \phi & \text{if } \alpha > \phi \\ \phi - \pi/2 & \text{otherwise} \end{cases} \tag{2.25}$$

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

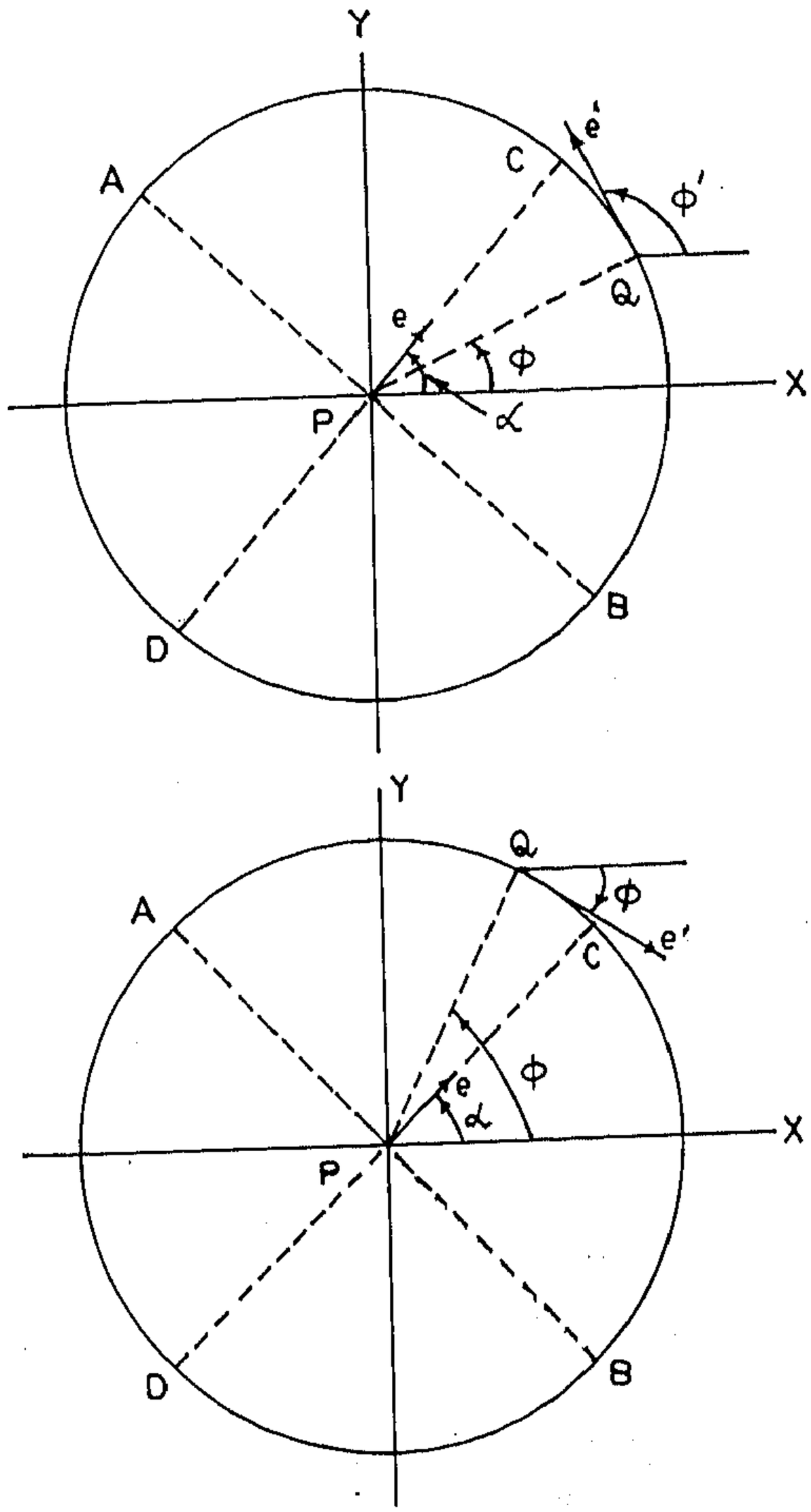


Figure 2.5: Edge induction on a circular neighborhood; AB is perpendicular to CD. **e** is the actual edge strength, $\mathbf{e}'$ is the induced edge vector.

strength is given as

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

The induced edge vector e′ can also be written as (e'_x, e'_y) in 2D rectangular coordinate system. It can be proved that (Appendix A.1)

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

In the process of edge point induction. spreading of edge points will occur due to the presence of any edge point. If there exists some broken edge (or line) segments and the length of the missing segment is approximately same or less than the diameter of the region of induction, then the broken segments are linked by induced edges. The noisy edges also spread their activity and will create regions of diffused edges. In order to decrease the effect of noise, each edge point should inhibit its own strength, just like the feedback in the previous algorithm (Section 2.3). In the process of reinforcement of edge strengths by edge point induction, a ridge of edge responses forms along a true edge (or line), where maximum response occurs at the true edge points.

If each edge point on an edge segment spreads its activity then an edge point is induced near the edge segment such that the normal distance of the point from the segment is less than the radius of the neighborhood of activity. Let P be a point near an edge segment AB, such that $PA = PB = L$, where L is the maximum distance from which a point on AB can induce P (Fig.2.6). It can be shown that (Appendix A.2) the induced edge strength e' at point P is

$$e' = 2eL\left[1 - \frac{tan^{-1}(L/D)}{(L/D)}\right] \tag{2.26}$$

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

$$e' = 2e\sqrt{R^2 - D^2}\left[1 - \frac{tan^{-1}\left(\sqrt{(R/D)^2 - 1)}\right)}{\sqrt{(R/D)^2 - 1}}\right] \tag{2.27}$$

It is clear from the above expression that $e' = 0$ when $D = R$, and is maximum $(e' = 2eR)$ when $D = 0$. The value of e' at a point decreases as the point moves

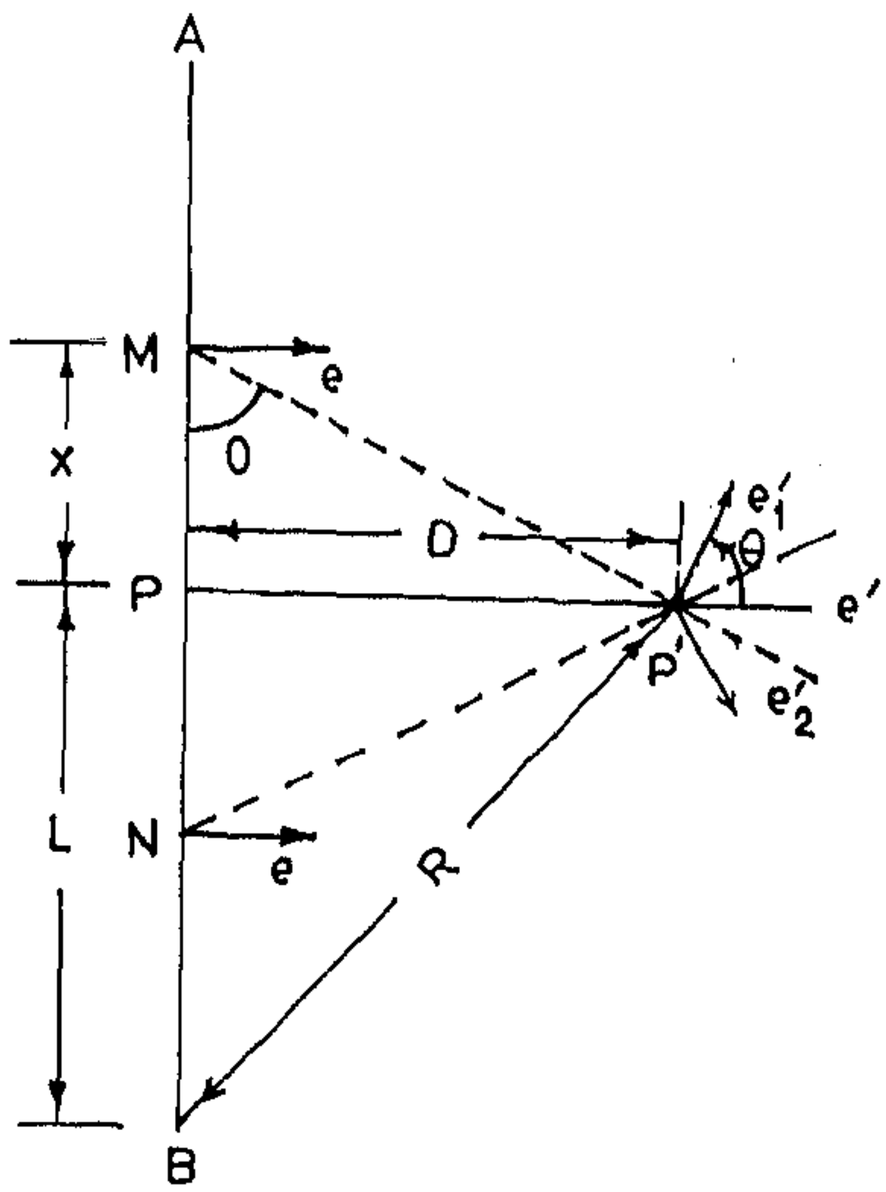


Figure 2.6: Edge induction at a point P' due to the edge segment AB. PP' is normal to AB. The edge strength at every point on AB is considered to be equal to e.

farther from the true edge segment. Therefore, the maximum values occur at the true edge points with a gradual decrese in edge strengths on either sides of the edge. If lateral nonmaximum suppression is performed then the actual edge points can be found out. The algorithm is implementable on a network model which is discussed below.

2.4.2 Network Model

The network model is composed of processing elements arranged in a two dimensional lattice similar to the model used in the previous method (Section 2.3). Here each PE is connected over a large neighborhood unlike the eight-neighborhood in the previous model. The model is designed to have a massively parallel implementation of the algorithm described above. The interconnection between any two processing elements is shown in Fig.2.7. Each neuron has two different intermediate outputs. Outputs v_{i1} and v_{i2} represent the edge responses along x- and y-directions (i.e. e_x and e_y) respectively at a location corresponding to processing element i. The weights of the interconnections depend on the spatial distribution of the processing elements which is governed by Equn. 2.28. For any two PEs i and j, if $i \in N_j$ (N_j is the neighborhood of j) then the weights of the links from j to i can be mathematically written as

$$\begin{aligned} w_{ij}^{11} &= w \sin^2 \phi_{ij} \\ w_{ij}^{22} &= w \cos^2 \phi_{ij} \\ w_{ij}^{12} &= -w \sin \phi_{ij} \cos \phi_{ij} \\ w_{ij}^{21} &= -w \sin \phi_{ij} \cos \phi_{ij} \end{aligned} \tag{2.28}$$

Here, ϕ_{ij} is the angle subtended with the x-axis by the imaginary line joining PE j to PE i (in the lattice structure). w_{ij} represents the weight of the link from processing element j to processing element i. The distribution of weights is shown in Fig.2.8 Each processing element i has negative self-feedback w_{ii}^{11} and w_{ij}^{22} as shown in Fig.2.8.

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

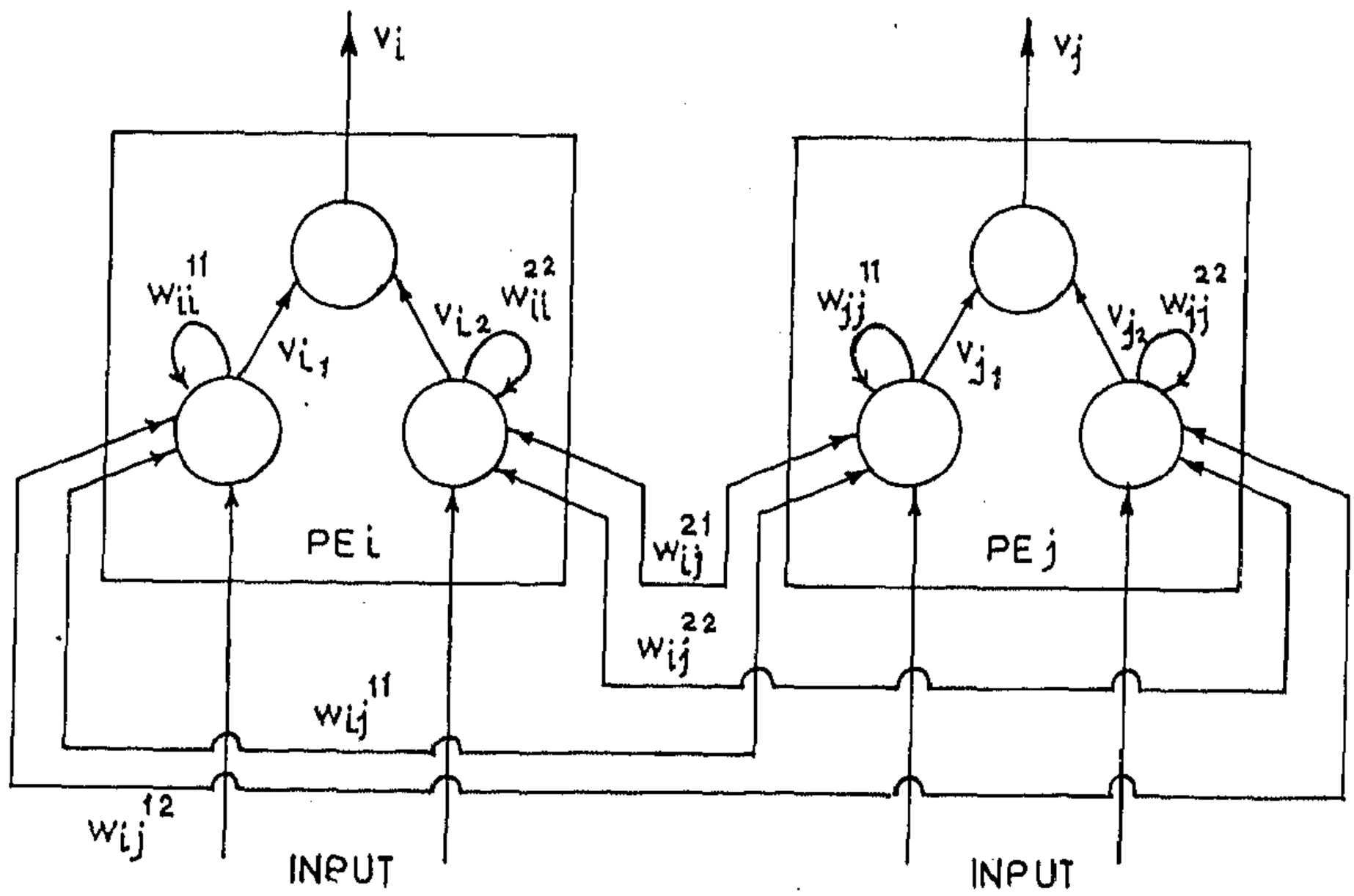
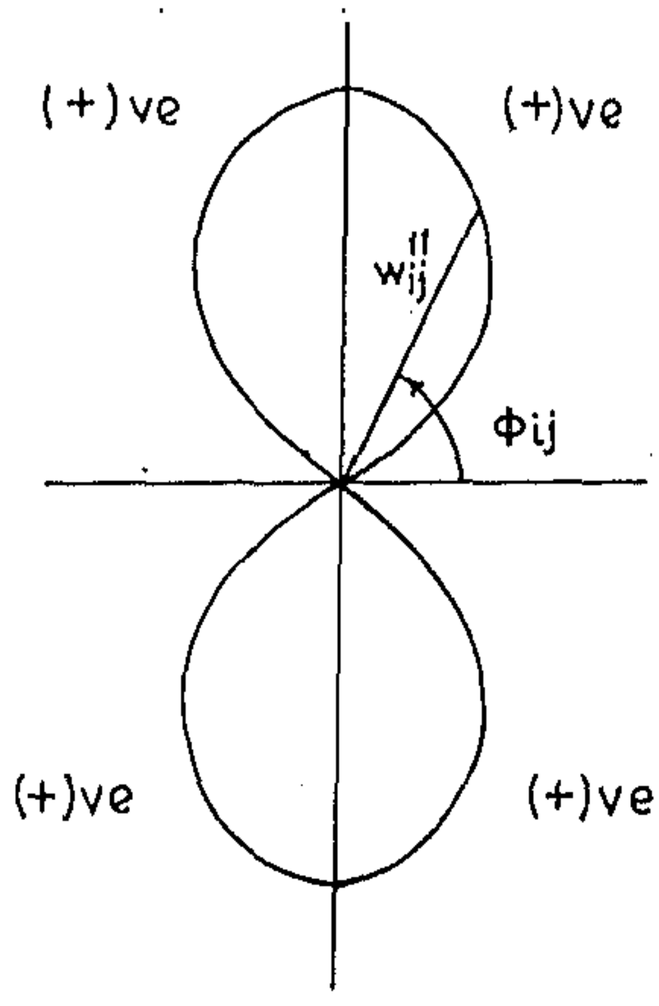
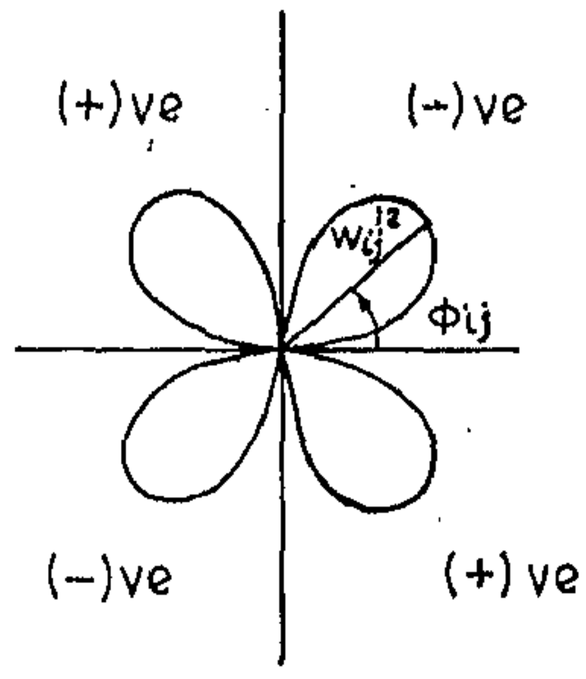


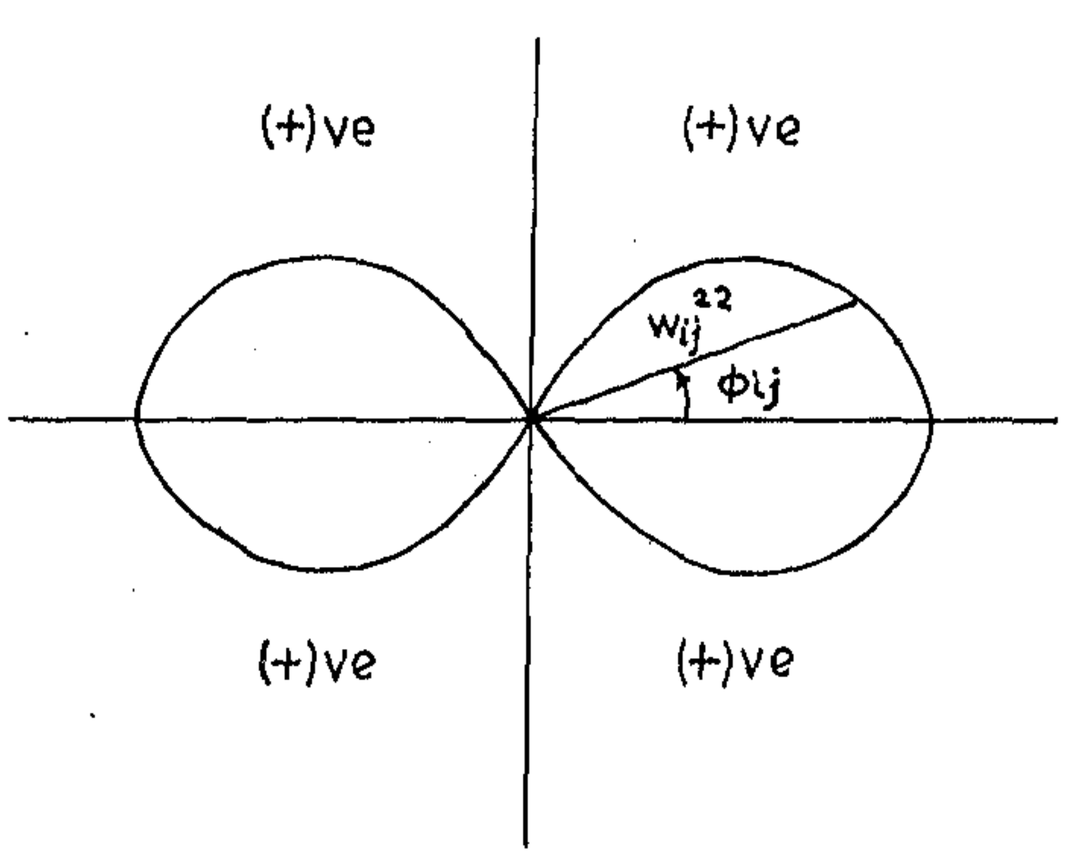
Figure 2.7: Connections between two processing elements i and j in the network. Each processing element has three active elements.



(a)



(b)



(c)

Figure 2.8: Distribution of weights of the interconnection links in polar coordinates. The radius vector denotes the absolute value of the weight : (a) distribution of w_{ij}^{11}, (b) distribution of w_{ij}^{12}, (c) distribution of w_{ij}^{22}.

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

$$\begin{aligned} w_{ii}^{11} &= -w_s R \\ w_{ii}^{22} &= -w_s R \end{aligned}$$

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

The output of any processing element i is given as

$$v_i = \sqrt{v_{i1}^2 + v_{i2}^2} \tag{2.29}$$

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

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

and

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

u_{i1} and u_{i2} are the instantaneous inputs to the processing element i as shown in Fig.2.7. The transfer function $g(.)$ can be chosen as a sigmoid function. In the present case, $g(.)$ is chosen as

$$g(x) = \begin{cases} 0 & \text{if } x < -1 \\ x & \text{if } -1 \leq x < 1 \\ 1 & \text{otherwise} \end{cases} \tag{2.30}$$

The dynamics of the network can be written as

$$\frac{du_{i1}}{dt} = \sum_{j,i \in N_j} w_{ij}^{11} v_{j1} + \sum_{j,i \in N_j} w_{ij}^{12} v_{j2} - u_{i1} + w_{ii}^{11} v_{i1} \tag{2.31}$$

and

$$\frac{du_{i2}}{dt} = \sum_{j,i \in N_j} w_{ij}^{21} v_{j1} + \sum_{j,i \in N_j} w_{ij}^{22} v_{j2} - u_{i2} + w_{ii}^{22} v_{i2} \tag{2.32}$$

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

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

and

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

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

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

$$\begin{aligned} E_1 &= -\frac{1}{2}\sum_i\sum_j w_{ij}^{11} v_{i1} v_{j1} - \frac{1}{2}\sum_i\sum_j w_{ij}^{12} v_{i1} v_{j2} \\ &\quad + \int_0^{v_i 1} g^{-1'}(\xi) d\xi - \frac{1}{2}\sum_i w_{ii}^{11} v_{i1}^2 \end{aligned} \tag{2.33}$$

$$\begin{aligned} E_2 &= -\frac{1}{2}\sum_i\sum_j w_{ij}^{21} v_{i2} v_{j1} - \frac{1}{2}\sum_i\sum_j w_{ij}^{22} v_{i2} v_{j2} \\ &\quad + \int_0^{v_i 2} g^{-1'}(\xi) d\xi - \frac{1}{2}\sum_i w_{ii}^{11} v_{i1}^2 \end{aligned} \tag{2.34}$$

The energy function of the system is given as

$$E(t) = E_1(t) + E_2(t) \tag{2.35}$$

Therefore,

$$\frac{dE}{dt} = \frac{dE_1}{dt} + \frac{dE_2}{dt} \tag{2.36}$$

Using the first derivatives of E_1 and E_2 (in Equn. 2.33 and 2.34), the expression for $\frac{dE}{dt}$ can be written as

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

$$-\frac{1}{2}\sum_i\sum_j\left[w_{ij}^{12}v_{j2}\left(\frac{dv_{i1}}{dt}\right)+w_{ij}^{21}v_{i2}\left(\frac{dv_{j1}}{dt}\right)\right]$$
$$-\frac{1}{2}\sum_i\sum_j\left[w_{ij}^{21}v_{j1}\left(\frac{dv_{i2}}{dt}\right)+w_{ij}^{12}v_{i1}\left(\frac{dv_{j2}}{dt}\right)\right] \qquad (2.37)$$

Since the weights are symmetric the Equn. 2.37 can be written as

$$\begin{aligned}\frac{dE}{dt} &= -\sum_i\left[\sum_j w_{ij}^{11}v_{j1}+\sum_j w_{ij}^{12}v_{j2}-u_{i1}+w_{ii}^{11}v_{i1}\right]\left(\frac{dv_{i1}}{dt}\right)\\ &\quad -\sum_i\left[\sum_j w_{ij}^{21}v_{j1}+\sum_j w_{ij}^{22}v_{j2}-u_{i2}+w_{ii}^{22}v_{i2}\right]\left(\frac{dv_{i2}}{dt}\right) \qquad (2.38)\end{aligned}$$

From Equn. 2.31 and 2.32

$$\frac{dE}{dt}=-\sum_i g^{-1'}(v_{i1})\left(\frac{dv_{i1}}{dt}\right)^2-\sum_i g^{-1'}(v_{i2})\left(\frac{dv_{i2}}{dt}\right)^2 \qquad (2.39)$$

Since $g(.)$ is an increasing function, $g^{-1'}(.)$ is positive. Therefore, it is evident that $\frac{dE}{dt}\leq 0$ for all $t>0$. Since $E(t)$ is bounded (from Equns.2.33,2.34)

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

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

As the radius of the neighborhood of activity of each processing element decreases, the energy of the system changes. At any point of time the change of energy is finite, and as a result, the rate of energy change reduces to zero as the neighborhood shrinks to a point. The output of the network depends on the rate at which the radius of the neighborhood decreases. If the radius decreases too fast then the true edge points may not be linked properly. On the other hand, if radius is decreased too slowly then flat bars of edge responses along the true edge points is formed. Let the radius of neighborhood of any processing element is decreased linearly at a rate γ, i.e. $\frac{dR}{dt}=-\gamma$. In that case to prevent the saturation of the edge strength near a true edge point, it is required that (from Appendix A.3)

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

In this expression, e is the edge strength at any point on a true edge, and $R(0)$ is the initial radius of the neighborhood. The relation is proved without considering

the self-inhibition of any edge point and the recursive effect from the induced edge points. For any image, γ is considered as $\lambda w R^2(0)$, where λ is a constant. Considering the inhibitory effect of self-feedback and excitatory effect from the induced edge points, the lower and upper bounds for γ can be fixed. In real images considering average edge responses, the value of λ is selected as 0.6.

2.5 Results and Analysis

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

2.5.1 Results from Model 1

The possible line pixels are found by matching all the templates corresponding to sixteen different line segments described in Section 2.3.1. At each pixel only the maximum and the second maximum responses are considered for subsequent processing. These two response values and the corresponding types (of the templates) found at a pixel are fed to the corresponding neuron. The state vector of each processing element consists of two state elements, i.e. n = 2. It is observed that the third maximum responses for most of the pixels in an image are much less than the first and second ones. Therefore, if the number of state elements are considered to be three or more, then there would be negligible change in the output of the network.

The network was simulated for different interconnection weights and different

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

In the first type of updating (Equn. 2.4) it is found that the effect of noise is greater than that in the second type of updating (Equn. 2.10). On the other hand, in second type of updating, discontinuities at some points are not resolved. Therefore, to have both the effects of linking and noise reduction, two methods are interleaved. Note that, the different ratio of the intermixing the updating rules result in outputs of different quality. The results have been presented in Fig.2.9 and Fig. 2.10.

In Fig.2.9(a) a portion of an airport in a satellite image is presented. The bright lines are detected in the image using the line detecting templates as mentioned before. The lines have been linked with different interleaved ratio, and the results are presented in Fig.2.9(a),2.9(b),2.9(c). The runway in satellite image has been detected.

Image of a girl is presented in Fig.2.10(a) The gradient response of the image was found using Sobel operator. The bright lines have been detected from the gradient image using the line detecting templates. The lines were linked and the result is presented in Fig.2.10(a),2.10(b),2.10(c).

It is evident from the results that for a ratio of 1:5 (Fig.2.9(b)) of the updating rules (Equn. 2.4 and 2.10) the linking of the line segments is better, but at the same time quite a few redundant line segments appear. For a ratio of 1:20 (Fig.2.9(d)), the noise decreases to a great extent while the linking is not as good as that in Fig.2.9(b). The results for a ratio of 1:10 (Fig.2.9(c)) presents an output of intermediate quality where a trade-off between the quality of linking and the noise degradation is achieved.

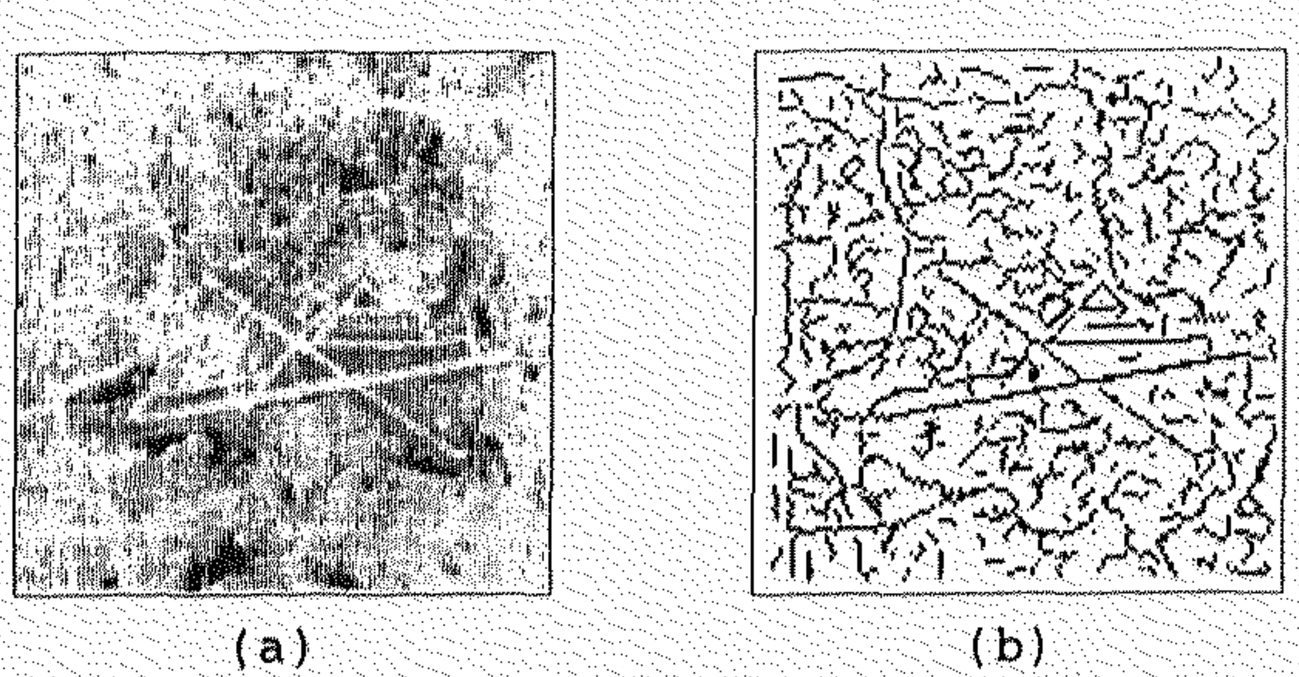


Figure 2.9: (a) Satellite image showing a part of an airport (128 × 128). (b) Lines detected using the interleaved ratio $s : p = 1 : 5$. (c) Lines detected using the interleaved ratio $s : p = 1 : 10$. (d) Lines detected using the interleaved ratio $s : p = 1 : 20$. Here, s is the number of iterations using the first updating rule and p is the number of iterations using the second updating rule. Parameters are selected as $w = 100$, $w_s = 50$, and time step = 0.005.

(a)

(b)

(c)

(d)

Figure 2.10: (a) An image of a girl (125×125). Lines detected from the gradient image using interleaved ratio (b) $s : p = 1 : 10$, (c) $s : p = 1 : 15$, (d) $s : p = 1 : 20$. Here, s is the number of iterations using the first updating rule and p is the number of iterations using the second updating rule. Parameters are selected as $w = 100$, $w_s = 50$, and time step $= 0.005$.

2.5.2 Results from Model 2

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

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

and

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

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

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

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

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

The weights of the interconnection between the processing elements are set according to Equn. 2.28. The value of w determines the speed of convergence of the network. A low value of w would cause a slow convergence and a very high value of w would result in rather unwanted oscillations. The value of w was selected experimentally and is chosen as 100. In fact, the value of w depends on the nature of the image also. The self-feedback, as discussed before, is taken to be proportional to the radius of the neighborhood. To prevent the noise to be detected as edge pixels, the self-feedback should be large enough so that the strength of the redundant edge pixels can reduce to zero with the emergence of the states of the processing elements. On the other hand, the amount of self-feedback should be small enough to ensure that the response of the true edge pixels are not reduced

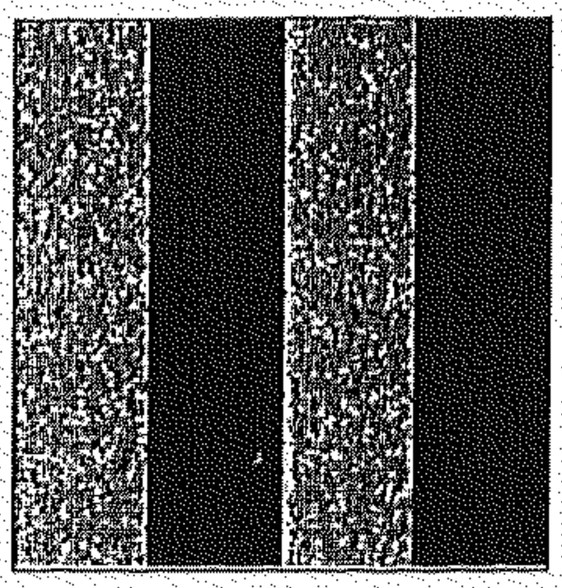
causing discontinuity. In ideal case, any edge point on a true edge will get support from 2R edge points (this is true when the edge is aligned with x- or y- axes). The edge point receives slightly less support, if the edge is not aligned with either of x- or y- axes. The self-feedback can be chosen slightly less than $2wR$. In the present system, the value of self-feedback was selected to be 180R.

The network was simulated to find out the edges of two different images as shown in Fig.2.11 and Fig. 2.12. The results show the edges linked for different radii of neighborhood (radius 2 and 3). It is evident from the results that the diffusion of edges is more with higher radius, and consequently more thick edges appear (Fig.2.12(b) and 2.12(d)). This causes comparatively more smoothing with radius 3 and also better linking which also supports the theory of edge induction as posed here.

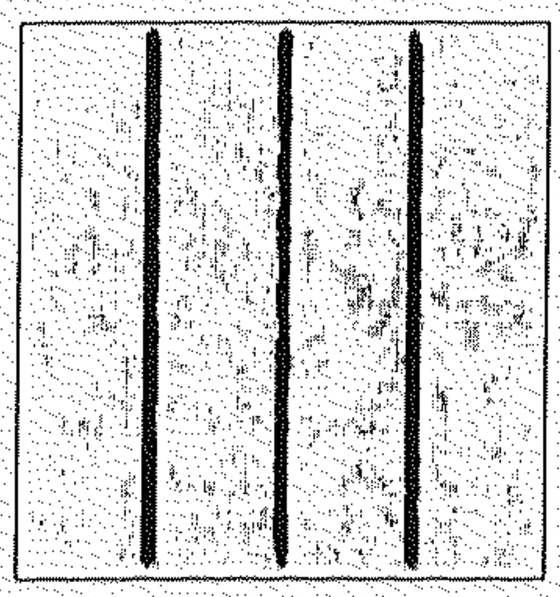
2.6 Conclusions and Discussion

In this chapter, two different connectionist models have been presented to deal with the task of edge/line linking by exploiting the cooperative and competitive processing capability of the neural architectures. The models have been designed for finding out the continuous line and edge segments in a graylevel image. The state updating rules and the corresponding convergence proof of the network have also been presented. Using these models the disadvantage of seed point selection in sequential line or edge linking is avoided.

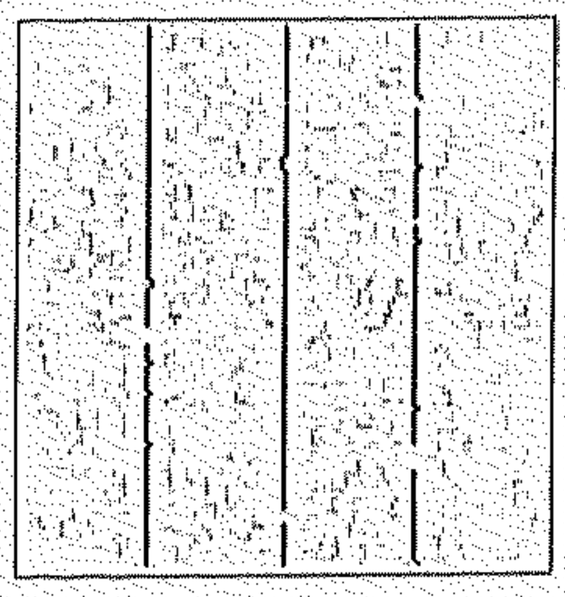
Both the network models, proposed here, work on the basis of neighborhood information. The first model relies on the information from eight neighborhood. Two different updating rules have been provided for the first model. Due to the presence of noise, two consecutive line or edge pixels on a segment may be lost. Using the second updating rule (Equn. 2.10) it would not be possible to restore the lost segment because a line point would be restored, in this case, only when it gets support from both the neighbors on either side. The first updating rule (Equn. 2.4) would be able to restore such kind of defects easily. On the other hand, the noise reduction capability of the second updating rule is much better than that of the first updating rule. Therefore, both the updating rules have been interleaved in linking the line segments. In the second model, larger neighborhood has been considered, and as a result, larger line or edge segments may be restored.



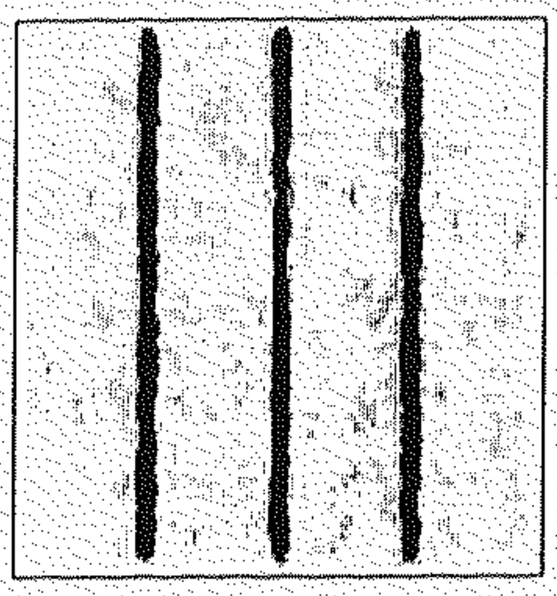
(a)



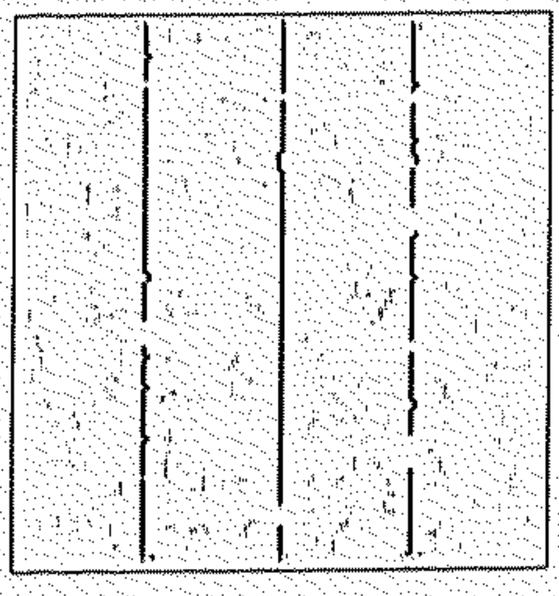
(b)



(c)



(d)



(e)

Figure 2.11: (a) Image (125 × 125) of noisy vertical bars. (b) Diffused edge cover (initial neighborhood, R(0) = 2). (c) Edges detected by lateral nonmaximum suppression from (b). (d) Diffused edge cover (initial neighborhood radius, R(0) = 3). (e) Edges detected by lateral nonmaximum suppression from (d). [$w = 100$, $w_s = 180$, $\lambda = 0.6$].

(a)

(b)

(c)

(d)

(e)

Figure 2.12: (a)Image (125 × 125) of a girl. (b) Diffused edge cover (initial neighborhood, R(0) = 2). (c) Edges detected by lateral nonmaximum suppression from (b). (d) Diffused edge cover (initial neighborhood radius, R(0) = 3). (e) Edges detected by lateral nonmaximum suppression from (d). [$w = 100$, $w_s = 180$, $\lambda = 0.6$].

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

In the first model, the open ended edge or line segments do not suffer lengthening. The output of a processing element, in this case, is updated only if it receives support from two neighbors (using second updating rule, Equn. 2.10). As a result, the states of the processing elements corresponding to the end points of the lines and edges are not updated (since they receive input only from single neighbors). Due to the presence of self-feedback, the output of PEs corresponding to end pixels decrease to some extent. If the first updating rule (Equn. 2.4) is used then the line segments can increase in length at their ends to an extent. Since the first and second updating rules are interleaved, they compensate each other, and as a result, the open ends of the line segments may suffer little change in length.

In the second model, the edge or line segments are lengthened with reduced strength at their open ends. This is due to the fact that the edge points at the open end induce edge vectors over a large neighborhood, the induced edge points further induce edge vectors and so on. As a result, the open ended edge segments may suffer an iterative lengthening. The amount of lengthening depends on the rate at which the radius of neighborhood is decreased. In real life images, open edge segments are less frequent than the closed loops of edges.

In the second connectionist model, the selection of initial radius of the neighborhood depends on the particular image. It is more or less dependent on the average separation of the edge segments in an image. Moreover, in the second model, at the junction of different edges, a gap may be created due to cross induction (i.e., simultaneous induction in different directions due to different edges). This phenomenon does not happen in the first model because in this case information about more than one type of line segment is present in each processing element. In the second model, the final result is obtained by selecting the points corresponding to the spine of the induced edge cover. There are many different techniques, available in literature [52] perform this selection. Here, non-maximum suppression technique is used for sake of simplicity.

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

Although the methods developed here are conceptually analogous to the relaxation labeling algorithms [67], [68], [71], they provide a new direction of implementing the cooperative processes (required for the linking) onto neural network models. Secondly, the present investigation comes up with a couple of new connectionist models which may also be applied to some other low-level operations apart from line and edge linking. Moreover, the present investigation shows the ability of connectionist models to embed the geometrical constraints into their architecture itself. Although the characteristics of preattentive vision were extensively studied by Grossberg and Mingolla [153], [155] the realization of the neural modeling was lacking there. The current investigation may find its use as a step towards establishing the bridge between the classical image processing techniques and cognitive psychology. ♠

In the following chapters, we will be concentrating on the task of feature interpretation using connectionist models. Chapter 3 presents a method using one of the basic networks (Hopfield model) whereas Chapters 4–7 deal with some application-specific models.

Chapter 3

Structure Matching using Hopfield Network

3.1 Introduction

An object can be described in terms of the description of its parts and spatial relations between the parts (as mentioned in Section 1.1 and Section 1.2.2). Such a description explicitly represents the spatial organization of the primitives forming an object. Structural descriptions are useful for many computer vision problems like shape matching [15],[220], stereo matching [221], etc. Structural object descriptions can be equivalently represented in terms of directed or undirected graphs in which the set of nodes represent the primitives and the arcs represent the spatial relations between these primitives. These graphs are commonly referred to as attributed relational graphs [222], [89]. Recognition schemes based on inexact matching of the graphs are developed in the classical approaches (described in Section 1.2.2). The number of features or primitives involved in such a matching scheme would be, in general, much smaller than the traditional feature (viz. edge points, corners) based methods. However, for construction of a structural description of the objects in terms of the known primitives and spatial relations, a priori knowledge of the problem domain is necessary.

Structural descriptions of the prototype and candidate objects can be matched by finding out a mapping between the primitives such that the spatial constraints

are preserved between them. In the case of inexact matching, a mapping is to be found out in order to maximize the match between the common portions of the two structural descriptions [15]. In other words, the problem is basically a *constraint optimization* problem. Hopfield nets [123],[124],[32] provide an efficient computational paradigm for solving these kinds of the problems [223],[162],[164].

In the present chapter, we demonstrate the effectiveness of a method which is developed using Hopfield network for finding out a suitable match for a candidate structure between a set of prototype structures [206]. In the formulation, the Hopfield network is considered as a two-dimensional array where the output of $(i,j)^{th}$ unit represent the degree of similarity between the i^{th} primitive and the j^{th} primitive of two shapes. Weights associated with the links indicate the compatibility between the possible matches represented by the corresponding units. These weights are preassigned on the basis of the spatial constraints between the primitives. This architecture is similar to that presented by Lin et al. [164]. But, in [164] asymmetric spatial relations like bottom, right, left, etc were not considered. The other related investigations [223, 162] also have not considered the asymmetric constraints. If asymmetric spatial relations are directly mapped onto the network, weights of the links will not remain symmetric and consequently convergence is not guaranteed under all possible conditions. A transformation for the structural descriptions has been proposed to circumvent this problem. With this transformation all types of spatial relations can be used in the structural descriptions. An energy function of the network has been formulated in a general fashion to take care of inexact matching between the descriptions. Also, the technique has been extended to find out the best match for a candidate structure between a set of prototype structures.

The rest of this chapter is organized as follows. The matching strategy between a candidate structure and a single prototype structure is described in Section 3.2. A graph theoretic transformation is proposed in Section 3.3 which can take care of the asymmetric spatial constraints. An energy function is formulated in this section to map the asymmetric constraints. The formulation of the energy function is extended in Section 3.4 in order to find the match for a candidate structure between a set of prototype structures. Section 3.5 presents the experimental results demonstrating the effectiveness of this method in matching the descriptions of handtools and characters. Finally, Section 3.6 presents the conclusions.

3.2 Matching Problem

In this section, we have considered the problem of finding a match between a candidate shape and the prototype so that the correct correspondence can be established between similar primitives of the two. Individual primitives of the shapes are considered to be low level geometric tokens, like lines, arcs, subparts of different shapes etc., which are characterized by a set of property-value pairs. Correspondence can be established between those primitives of the two shapes which possess approximately equivalent set of property-value pairs. In addition, spatial constraints between the primitives in one shape must be preserved among the matched primitives in the other shape. Spatial constraints or the relational features between the primitives can be also characterized by a set of property-value pairs and accordingly matching criterion can be formulated. Also, in many cases two shapes do not match completely, and only a portion of the given shapes may match. The neural network based matching scheme proposed here has been motivated by the above mentioned considerations. For matching structural description of the two shapes, having N primitives each, an artificial neural net with $N \times N$ neural units have been used. Each neural unit represents a possible match between individual primitives of the two shapes under consideration. This representation is natural because each primitive of the candidate shape can be mapped on to any one of the primitives of the prototype. The lowest energy state of the network, in accordance with an appropriately defined energy function, will represent the best mapping between the primitives of the candidate and prototype shapes which satisfy all the problem dependent constraints.

Let us assume that the energy function has the following form:

$$\mathcal{E} = \mathcal{E}_1 + \mathcal{E}_2 + \mathcal{E}_3 + \mathcal{E}_4 + \mathcal{E}_5 \tag{3.1}$$

Each term on the right hand side of the equation (3.1) represents contribution due to individual constraints. The term $\mathcal{E}_1$ ensures that any primitive in the candidate shape cannot match with more than one primitive in the prototype and vice-versa. Hence, $\mathcal{E}_1$ is defined in the following form:

$$\mathcal{E}_1 = A/2 \sum_p \underbrace{\sum_i \sum_j}_{i \neq j} v_{ip} v_{jp} + B/2 \sum_i \underbrace{\sum_p \sum_q}_{p \neq q} v_{ip} v_{iq}, \tag{3.2}$$

where A and B are positive constants; v_{ij} represents the state of the $(i,j)^{th}$ neuron (v_{ij} can take values between 0 and 1). The first sum in $\mathcal{E}_1$ is zero if and only if

there is no more than one nonzero entry in each column. The second term is zero if the same condition holds good for each row.

The term $\mathcal{E}_2$ ensures that the total number of primitives matched is equal to N. Therefore, $\mathcal{E}_2$ has been chosen as :

$$\mathcal{E}_2 = C/2(\sum_i \sum_p v_{ip} - N)^2, \tag{3.3}$$

where, C is a positive constant. When two shapes containing N primitives are perfectly matched this term becomes zero. If the shapes contain different number of primitives, say X and Y, the parameter N must be set to $min(X, Y)$ because the network can find match for maximum of $min(X, Y)$ primitives. Even in case for partial match, minimum value of this term ensures maximum possible number of matches between the primitives. This problem can be also tackled by including dummy primitives in the descriptions containing lesser number of primitives.

As mentioned before, primitives of the candidate and prototype shapes are characterized by a set of possible attributes or properties and values of these attributes. Two primitives can be considered to be matched if they have nearly identical set of attributes and similar values for those attributes. This criterion is captured by the $\mathcal{E}_3$ term of the energy function which is given by

$$\mathcal{E}_3 = D/2 \sum_i \sum_p v_{ip} \sum_{a=1}^{\mathcal{A}} \sum_{b=1}^{\mathcal{B}} W_i(a)(\mathcal{P}(i,a,b)(1-\mathcal{C}(p,a,b)) + \mathcal{C}(p,a,b)(1-\mathcal{P}(i,a,b))), \tag{3.4}$$

where, D is a positive constant

$W_i(a)$ is the priority or weight of the attribute a for the primitive i (this is assumed to be known a priori).

a is the variable representing individual attributes assumed to be indexed from 1 to $\mathcal{A}$.

b is the index for individual elements of the possible value sets of the attributes. The value sets have been assumed to be finite and discrete. Cardinality of the largest set is assumed to be $\mathcal{B}$.

$\mathcal{P}(i,a,b)$ is the selector function which can be either 1 or 0.

$\mathcal{P}(i,a,b) = 1$ if the primitive i of the prototype possess attribute a with the value b,

$\mathcal{P}(i,a,b) = 0$ otherwise

$\mathcal{C}(p,a,b)$ is the corresponding selector function for the primitives of the candidate shape.

This term will become zero when property-value pairs for the matched primitives of the candidate as well as the prototype shapes are exactly identical. In case of inexact match, if we expect partial match of the primitives of the prototype shape among the primitives of the candidate, the term $\mathcal{E}_3$ can be modified to

$$\mathcal{E}_3 = D/2 \sum_i \sum_p v_{ip} \sum_{a=1}^{A} \sum_{b=1}^{B} W_i(a)(\mathcal{P}(i,a,b)(1-\mathcal{C}(p,a,b))). \tag{3.5}$$

In equation (3.5), $\mathcal{E}_3$ will be zero even if a primitive (say, p) is present in the candidate (i.e., $\mathcal{C}(p,a,b)=1$) and is absent in the prototype (i.e., $\mathcal{P}(p,a,b)=0$). In other words, modified form of $\mathcal{E}_3$ would not pose any constraint for such a partial match. Similar modification can be incorporated when only consistent matches of the priinitives of the candidate shape is being looked for.

But, $\mathcal{E}_3$ being zero does not ensure correct match because none of the terms of the energy function so far considered guarantees existence of identical spatial constraints between the matched primitives of the candidate and the prototype shapes. This condition is imposed by the term $\mathcal{E}_4$ of the energy function.

$$\begin{aligned}\mathcal{E}_4 \;=\; & E/2 \sum_i \sum_j \sum_p \sum_q v_{ip} v_{jq} \sum_{s=1}^{S} W(s)(\mathcal{RR}(i,j,s)(1-\mathcal{CC}(p,q,s)) \\ & +\mathcal{CC}(i,j,s)(1-\mathcal{RR}(p,q,s))),\end{aligned} \tag{3.6}$$

where,

E	is a positive constant
s	is the index for relational attributes.
$W(s)$	is the priority of the s^{th} spatial relation.
$\mathcal{RR}(i,j,s)$	is the selector function which takes values either 0 or 1 $\mathcal{RR}(i,j,s)=1$ if s^{th} relation exist between i^{th} and j^{th} primitives of the prototype, $\mathcal{RR}(i,j,s)=0$ otherwise.
$\mathcal{CC}(p,q,s)$	is the corresponding selector function for the primitives of the candidate shape.

In equation (3.6), only binary relational constraints are taken into account. Also, as described for $\mathcal{E}_3$, this term becomes zero only for perfect match between the

shapes. Modifications similar to those for the term $\mathcal{E}_3$ can also be incorporated here. This term can be easily modified in the following way to accommodate k-ary relational constraints:

$$\mathcal{E}_4 = E/k \sum_{i_1} \sum_{i_2} \ldots \sum_{i_k} \sum_{p_1} \sum_{p_2} \ldots \sum_{p_k} v_{i_1 p_1} v_{i_2 p_2} \ldots \sum_{s=1}^{S_k} W(s)(\mathcal{RR}(i_1, i_2, \ldots, i_k, s) (1 - \mathcal{CC}(p_1, p_2, \ldots, p_k, s)) + \mathcal{CC}(p_1, p_2, \ldots, p_k, s)(1 - \mathcal{RR}(i_1, i_2, \ldots, i_k, s)) \quad (3.7)$$

If the constants A,B and C are sufficiently large as compared to D and E, a local minima in energy state is expected to give a good match between the candidate and prototype descriptions.

Based on the energy function it is straight forward to determine the input bias and connection weights for the neural network. The coefficients of the quadratic terms corresponds to the weights in the connection matrix and that of the linear terms corresponds to input bias to the neurons. The connection weights (w) are given by,

$$w_{ip,jq} = \underbrace{-A\delta_{pq}(1-\delta_{ij})}_{\textit{inhibitory connections within each row}} \overbrace{-B\delta_{ij}(1-\delta_{pq})}^{\textit{inhibitory connections within each column}} \underbrace{-C}_{\textit{global inhibition}}$$
$$\underbrace{-E\sum_{s=1}^{S} W(s)(\mathcal{RR}(i,j,s)(1-\mathcal{CC}(p,q,s)) + \mathcal{CC}(i,j,s)(1-\mathcal{RR}(p,q,s)))}_{\textit{inhibition by the binary constraints between the primitives}} \quad (3.8)$$

where,

$\delta_{ij} = 1$ if i=j

$=0$ otherwise.

The input bias (I) to each neuron is given by

$$I_{ip} = \underbrace{CN}_{\textit{excitation bias}} \underbrace{-D/2 \sum_{a=1}^{\mathcal{A}} \sum_{b=1}^{\mathcal{B}} W_i(a)(\mathcal{P}(i,a,b)(1-\mathcal{C}(p,a,b)) + (\mathcal{C}(p,a,b)(1-\mathcal{P}(i,a,b))}_{\textit{bias by the attributes of the primitives}} \quad (3.9)$$

In this mapping it is interesting to note that the attributes of the primitives contribute to the input bias of the neurons and the relational constraints contribute to the inhibitory connections between the neurons.

It is clear from the above formulation of the structure matching problem that the weights associated with the links connecting the neurons are determined by the nature of the relational constraints. For symmetric spatial relations weights will be symmetric. But if we consider asymmetric spatial relations, viz. right, left, above etc., $\mathcal{RR}(i,j,s) \neq \mathcal{RR}(j,i,s)$ and $\mathcal{CC}(p,q,s) \neq \mathcal{CC}(q,p,s)$. Consequently connection weights will be no longer symmetric. But the Hopfield network is not guaranteed to converge to minimum energy state for asymmetric connection strengths. It is therefore necessary to identify a modified strategy so that shape descriptions involving asymmetric spatial relations can be reliably matched with Hopfield network using the above formulation. In the next section, a scheme is suggested for mapping the problem to the Hopfield network having symmetric interconnection weights even for shape descriptions involving asymmetric spatial relations.

3.3 Shape Matching with Asymmetric Spatial Relations

In this section we describe a technique for transforming structural descriptions of the shapes so that the shapes described in terms of asymmetric spatial relations can be matched efficiently using Hopfield network with symmetric interconnection weights.

A structural description of a shape can be equivalently represented by an attributed relational graph [222]. The nodes of this graph represent individual primitives and directed arcs are the representation of the spatial relations between the primitives. Let the attribute sets A_{n_i} and $A_{e_{jk}}$ are associated with a node n_i and an arc e_{jk} respectively. The set A_{n_i} contains attributes describing geometric properties of the primitives and the elements of the set $A_{e_{jk}}$ describe the spatial relation between the primitives linked by the arc e_{jk}. Asymmetric spatial relations between the primitives can be most naturally represented using this graph. Symmetric relations between the nodes n_j and n_k can also be represented in this framework by associating same relational attributes with both the edges e_{jk} and e_{kj}. Consider the example of a simple attributed, relational graph shown in Fig.3.1.

For the given example, the directed arc between n1 and n2 indicates that the

Figure 3.1: Example of an asymmetric spatial relation.

primitive n2 is to the left of n1. Now, let us replace the directed arc between n1 and n2 by an undirected edge connecting the two nodes. This edge can be associated with an attribute which would indicate the type of the spatial association between the nodes. Consequently, both the sets $A_{e_{12}}$ and $A_{e_{21}}$ will have identical elements. To encode the exact sense of the spatial relation, additional attributes must be included in the attribute set of the corresponding nodes. Since, the primitive n1 is to the left of n2, A_{n1} will have additional attribute which will be indicative of the spatial relation "Left" while A_{n2} will have additional attribute indicative of "Right" i.e. the spatial relation of n2 with respect to n1. In this manner, all the directed edges between the nodes in an attributed relational graph can be replaced by undirected symmetric edges. Edges will be associated with the types of the spatial relations. Types of the spatial relations will not have any sense of direction or asymmetricity attached with them (henceforth i th type of the spatial relation being indicated by the symbol $\mathcal{T}_i$). For each type of spatial relation, two relational attributes (k^{th} relational attribute being indicated by r_k) will be considered for indicating the exact sense of the spatial relation between the nodes. If a relational attribute r_k indicate an asymmetric spatial relation of the node n_i with node n_j then, we say that the spatial relation of the node n_j with the the node n_i can be indicated by the dual of the relational attribute r_k represented as $\overline{r_k}$. For the given example, if $r_k = Left$ then $\overline{r_k} = Right$. The transformation technique discussed here is a general purpose procedure by which any attributed relational graph can be transformed into an undirected graph.

Let us consider an attributed relational graph $G = (V, E)$. where, V is the set of nodes and E is the set of directed edges. The graph G can be transformed into a undirected graph $G_T = (V_T, E_T)$ by following the procedure described above. Now, in the graph G_T, corresponding to each node n_i the modified attribute set A^T_{ni} will be $A_{ni} \cup Ar_{ni}$. The set Ar_{ni} contain additional relational attributes. Each element of Ar_{ni} is a tuple (r_k, c_k) where, r_k is the relational attribute for the k^{th}

type of the spatial relation (dual of the k^{th} relation can be indicated by using $\overline{r_k}$ or by associating a different index, say j, with it), and c_k is the number of nodes with which the node n_i has the k^{th} spatial relation. The count c_k must be maintained because a node can have same relation with more than one node. Therefore, in the above example, if the relations are transformed then edge-type between $n1$ and $n2$ indicates that there exists a left-right relationship between $n1$ and $n2$ (although it does not specify which one is left of the other), and relational attribute indicates that the node $n1$ has one node ($c_1 = 1$) to its left, while node $n2$ has one node to its right. The graph G_T can be effectively used for representing structural descriptions if and only if the transformation T is unique. Otherwise, a transformed graph G_T can ambiguously represent more than one descriptions. Consequently, any matching strategy which uses the transformed graph for representing the structural descriptions is bound to make inconsistent conclusions. Subsequently, we establish the properties of the transformations and identify the conditions under which it becomes an effective representational scheme.

Theorem 3.1 *Transformation T cannot produce any other attributed graph other than G_T from G.*

Proof : Proof is straightforward because G always has a unique undirected structure and for any particular node, the total number of arcs of a particular type going to or coming from that node is always fixed. □

Now, we need to show that there exists an inverse transformation procedure which will produce a directed attributed relational graph which is the same as that of the original graph G. To start with we present the inverse transformation algorithm.

ALGORITHM Inverse-transform(G_T)

for all nodes $n_i \in V_T$
  for all edge-types $\mathcal{T}_i$ associated with the edges coming out from the node n_i
    if edges of the type $\mathcal{T}_i$ has not yet been associated with directions
      call label($n_i, \mathcal{T}_i, flag$)
      /* $flag$ is set true by the procedure label
      if the labeling is successfully accomplished */

endif
endfor
endfor

Procedure label($n_i, \mathcal{T}_i, flag$)

1. Put in *choice set* the tuple $(r_i, c_{r_i}), (\overline{r_i}, c_{\overline{r_i}}) \in Ar_{n_i}$

/* r_i and $\overline{r_i}$ are the relational attributes of a node corresponding to an edge of type $\mathcal{T}_i$

2. for all edges (n_i, n_j) of type $\mathcal{T}_i$ emanating from n_i

2.1 Choose an attribute, say, r_i (or $\overline{r_i}$)from the *choice set* which has not yet been chosen for that edge

2.2 if no attribute exist in the *choice set* which can be chosen for the edge (n_i, n_j)

2.2.1 $flag = false$; return

2.3 If the set Ar_{n_j} of the node n_j has a tuple, say $(\overline{r_i}, c_{\overline{r_i}})$ (or (r_i, c_{r_i})) with $c_{\overline{r_i}}$ (or c_{r_i})≥ 1,

2.3.1 associate the direction from n_i to n_j (or from n_j to n_i) to the edge (n_i, n_j);

Decrement $c_{\overline{r_i}}$ (or c_{r_i}) of the node n_j

2.3.2 Call label $(n_j, \mathcal{T}_i, flag)$

2.3.3 If $(flag = false)$ go to 2.1

2.3.4 If $(flag = true)$ decrement c_{r_i} (or $c_{\overline{r_i}}$ as the case may be); go to 2

2.4 else go to 2.1

3. end for
4. $flag = true$; return.

Theorem 3.2 *The algorithm inverse-transform produces a graph G' wherein all the spatial constraints specified in the graph G are preserved.*

Proof: In the process of obtaining the graph G_T from the graph $G = (V, E)$ according to the transformation procedure described in this section, no nodes or edges are eliminated. Also, in the algorithm inverse-transform no edge or vertex is being eliminated in any step. Therefore, $|V'| = |V|$ and $|E'| = |E|$. In other words, according to both the graphs G and G' the set of nodes that are spatially related to each other are identical. Also, information about the type of the relations between

the nodes is preserved in the transformed graph G_T and consequently in G'. In the process of transformation, only the edges are stripped off their directions and the algorithm inverse-transform reassociates the directions with the edges such that the sense of the asymmetric spatial relations are preserved. Let us assume that the algorithm produces a graph G' in which direction of at least one edge (n_i, n_j) is different from that in the graph G. Let the edge be of type $\mathcal{T}_i$, and the procedure *label* has associated the opposite direction with the edge (n_i, n_j) because corresponding to that direction the conditions specified in the step 2.3.1 of the procedure was satisfied by the tuples of the attribute set of the node n_j and further recursive calls to the procedure label for associating directions with the edges of type $\mathcal{T}_i$ continued successfully. The sequence of recursive calls got terminated only after associating directions with all the related edges (i.e., the edges belonging to the paths of type $\mathcal{T}_i$ in the graph G_T which passes through the node n_i) without violating the constraints in the step 2.3.1. These constraints can be satisfied only if there exist a sequence of nodes $(n_i, n_j, n_{j+1}, n_{j+2}, ..., n_k)$ with $c_{r_i} \geq 0$ and $c_{\overline{r_i}} \geq 0$ for each such node for compensating the choice of wrong direction for the arc (n_i, n_j). Again, for the given sequence of nodes the compensation must take place without affecting directions of the other edges connected to the nodes. This implies that, for those edges, correct labels are available in the choice set. This can happen, only when error can be compensated by interchanging directions of the edges (n_{l-1}, n_l) and (n_l, n_{l+1}) for all the nodes in the sequence. This is possible only if $n_i \equiv n_k$. In other words, all the above conditions will be true, in case, there exist a directed cycle corresponding to a particular type of spatial relation in the original graph G. The algorithm, inverse-transform, therefore, can associate wrong direction to an edge in a cycle such that direction of the cycle as a whole becomes oppositely defined. As a consequence, the spatial constraint between the primitives in the graph G is preserved in the graph G' in terms of spatial relations having opposite sense. In absence of a cycle, the initial assumption will turn out to be false because in course of recursive calls, as described before, it will encounter a situation where either c_{r_i} or $c_{\overline{r_i}}$ will become zero and hence the initial wrong direction cannot be compensated. So ultimately the procedure label will backtrack and initial mistake will be rectified. Therefore, in the graph G', all the constraints between the primitives as specified in the graph G are preserved. □

As a consequence of the above theorem, asymmetric spatial relations can be taken care of without using asymmetric connection weights between the neurons. Relative significance of the relational constraints will be used for calculating symmetric

interconnection weights. Asymmetric nature of the spatial relations will be instrumental in determining input bias of the neurons. For the neurons involved, selector functions corresponding to the associated primitives will indicate presence or absence of the relational attribute or its dual.

The term $\mathcal{E}_4$ takes care of the difference between ordinary attributes of the primitives, and we need to include another term $\mathcal{E}_6$ in the energy function calculation for taking into account mismatch between the relational attributes of the primitives. In this case, difference in the number of other primitives which satisfy a particular type of spatial relation with the primitives will be of significance. The term $\mathcal{E}_6$ is defined in the following way:

$$\mathcal{E}_6 = F/2 \sum_{i_c} \sum_{j_p} v_{ip} \sum_{s} W(s)(\sigma(c_s^{i_c} - c_s^{j_p}) + \sigma(c_s^{j_p} - c_s^{i_c}), \tag{3.10}$$

Here,

F	is a positive constant.
s	is the index for relational attributes.
$Wi(s)$	is the priority of the s^th relational attribute.
$c_s^{i_c}$	is the number of s^{th} relational attributes associated with the i^{th} primitive of the candidate shape. Here, duals of the primitives have been considered to be appropriately indexed.
$c_s^{j_p}$	is the number of s^{th} relational attributes associated with the j^{th} primitive of the prototype shape.
$\sigma()$	is a function which is defined in the following way: $\sigma(\mathcal{Z}) = \mathcal{Z}$ if $\mathcal{Z} \geq 0$, $\sigma(\mathcal{Z}) = 0$ if $\mathcal{Z} < 0$.

The function $\sigma()$ takes care of the following situations. When the function $\sigma(c_s^{i_c} - c_s^{j_p})$ in $\mathcal{E}_6$ becomes zero, it enables us to ignore relational constraints which are present in the input candidate structural description in addition to those which have been satisfied for the prototype under consideration.

3.4 Shape Matching with Multiple Prototypes

The problem of matching a prototype with a candidate structural description can be extended to the problem of finding the best match for a candidate structure in a library of multiple stored prototypes. Let there be M prototypes, each having a maximum of N primitives. Let us assume that the input candidate shape possesses N primitives only. So, as discussed in Section 3.2 , we need to use $M \times N \times N$ neurons with which we can consider all possible matches between the primitives. These neurons are basically arranged as a stack of two-dimensional arrays, where each plane represents the matching with a single prototype. So, the neuron (i, l, p) is fully active if the i^{th} primitive in the candidate structure matches the l^{th} primitive of the p^{th} prototype.

The basic constraints involved in the process of shape matching can be mathematically formulated as follows:

1. If $v_{ilp} = 1$ then $v_{jkq} = 0 \forall j, k$ and $q \neq p$. This condition inhibits matching of the primitives of a candidate shape with the primitives of different prototypes.
2. In a column or a row of a single plane of neuron, activation of only one neuron can become 1. This implies that one primitive of the candidate can match only one primitive of the prototype and vice-versa.

To take care of the constraints, in the actual network implementation, shapes having lesser number of the primitives are to be appended with dummy primitives. These dummy primitives, can match with only other dummy primitives with minimum cost; but matching of the dummy primitives with any of the valid primitives incur maximum contribution to the energy function. Use of dummy primitives enable us to tackle the problem of missing primitives in this framework. Based on the above mentioned constraints energy function for the net is redefined in the following way:

$$\mathcal{E} = A/2 \underbrace{\sum_i \sum_j}_{i \neq j} \sum_l \sum_p v_{ilp} v_{jlp} + B/2 \sum_i \underbrace{\sum_l \sum_m}_{l \neq m} \sum_p v_{ilp} v_{imp}$$

$$+C/2\underbrace{\sum_{p}\sum_{q}}_{p\neq q}(\sum_{i}\sum_{l}v_{ilp})(\sum_{j}\sum_{k}v_{jkq})$$

$$+D/2(\sum_{i}\sum_{l}\sum_{p}v_{ilp}-N)^2$$

$$+E/2\sum_{i}\sum_{l}\sum_{p}v_{ilp}\sum_{a=1}^{A}\sum_{b=1}^{B}W_i(a)(\mathcal{P}(i,l,a,b)(1-\mathcal{C}(p,a,b)))$$

$$+F/2\sum_{i}\sum_{j}\sum_{l}\sum_{m}\sum_{p}v_{ilp}v_{jmp}\sum_{s=1}^{S}W_{ij}(s)(\mathcal{RR}(i,j,p,s)(1-\mathcal{CC}(p,q,s))$$

$$+\mathcal{CC}(i,j,s)(1-\mathcal{RR}(p,q,s)))$$

$$+G/2\sum_{i}\sum_{l}\sum_{p}v_{ilp}\sum_{s=1}^{S}W_i(s)(\sigma(c_s^{i_c}-c_s^{j_p})+\sigma(c_s^{p}-c_s^{i}))\text{.} \qquad (3.11)$$

Symbols used in equation (3.11) have similar meaning as that in Section 3.2. Based on equation (3.11), interconnection weights and input bias of the neurons of the Hopfield network can be computed as described in equations (3.9) and (3.8).

3.5 Results and Analysis

To establish effectiveness of the matching mechanism developed in the previous sections we have considered application of the technique for solving problems of different domains. In the following subsections, we have summarized our observations.

The network has been simulated on micro-vax (supporting Ultrix-32 operating system). The states of the neurons in the Hopfield network have been synchronously updated in the sequential simulation code. This is done because simulation of the asynchronous updating requires very small discrete time step and consequently, time required for convergence becomes very large. On the other hand, in synchronous updating, since the effect of updating the state of one neuron is communicated to others only after completion of updating of all the neurons for a particular time instant, we can use relatively larger time steps. In the present case, size of the time step was chosen to be 0.001. The initial states of the neurons are chosen in such a way that the total sum of the outputs of the neurons in initial state is the same as that for final state. The total output in the initial state must

be uniformly distributed over all neurons. As the network is updated, depending on the interconnection pattern, output of some of the neurons becomes 1 and that of the others gradually diminish. Since, in the final state, only N neurons, corresponding to the consistent match of N primitives of the input candidate shape should be fully active, we have for the simple single-layered model: $\sum_i \sum_j v_{ip} = N$. Hence, initial states are set as

$$U_i(0) = U_{00} + \delta U_i \tag{3.12}$$

where, $U_{00} = g^{-1}(1/N)$ and $|\delta U_i| \leq 0.1 U_{00}$. The term δU_i introduces a perturbation in the initial states of the neurons such that the network is prevented from getting stuck at the initial configuration. In case of the multi-layered model also, the total number of neurons becoming fully active in the final state must be N. Hence, initial values can be chosen according to the following equation: $U_{00} = g^{-1}(1/M \times N)$. The transfer function g(.) is chosen as

$$g(x) = 1/(1 + exp(-x)) \tag{3.13}$$

With this transfer function $U_{00} = -\log(N-1)$ for single layered networks and $U_{00} = -\log(MN-1)$ for the multiple layered networks.

Quality of the experimental observations are definitely dependent on the interconnection weights among the neurons in the network. Interconnection weights, in turn, are dependent on the values of the parameters A,B,C,D,E, F,G. The parameters A,B,C,D determine relative strength of the domain constraints while the parameters E,F,G determines the relative significance of the spatial constraints among the primitives. If the value of A,B,C,D are much greater than those of E,F,G, mismatch may occur because spatial constraints are violated under such circumstances. On the contrary, if the values of E,F,G are comparatively higher then matches are obtained wherein domain constraints may be violated, viz., in the final result, a single primitive of the test shape may be found to be in correspondence with more than one primitive of the prototype. After careful experimentation, final parameters for present implementation were chosen to be as: $A = 200, B = 200, C = 200, D = 200, E = 500, F = 500, G = 500$. All the weights of the attributes, relations, and relational attributes were taken as unity.

3.5.1 Recognition of Planar Shapes

Structural shape descriptions are useful representations of a variety of shapes which can be easily decomposed into primitive parts having discernible spatial relations between them. Hand tools form a class of objects which can be efficiently represented using structural descriptions. In robotic applications, it should be possible for the vision systems to identify these objects despite partial distortions in their shapes. Also, recognition systems must be capable of discriminating between several similar hand tools. The proposed scheme for matching structural descriptions can be effectively used for this problem. As a case study, we have considered the problem of finding the closest match for a given handtool among the set of similar hand tools known a priori. As a handtool, different kinds of hammers have been considered. Hammers have been considered to be composed of subparts, each of which can be characterized in terms of their shapes, relative size and color. Parts have been found to have any one of the following shapes - triangle, rectangle, trapezium. Parts of similar shape can vary in size. With respect to the size we have classified the parts into three distinct classes - small, medium and large depending on their relative areas. Various types of joins among the parts have been used as spatial relation between the parts. Instead of specific numeric parameters, use of basic nature of joins as relational features has enabled us to obtain invariant shape descriptions despite minor variations in the shape. Following types of joins are considered for the present problem.

$< \mathcal{T}_1, X, Y >$ join of one end of either the primitive X (or Y) with the primitive Y (or X) at the midpoint of Y (or X) in an orthogonal fashion. Exact sense of the relation will be captured by relational label of the edge connecting the nodes which represent the primitives X and Y in the corresponding relational attributed graph of the shape under consideration.

$< \mathcal{T}_2, X, Y >$ join of the end point of the primitive X with that of Y.

$< \mathcal{T}_3, X, Y >$ placement of the primitive X (or Y) at the middle of the other between its both the ends.

Corresponding to these types of the spatial associations between the primitives, labels of the edges of attributed graphs (used for representing the hammers) are R_1, R_2, R_3. Dual of these relations are indicated as $\overline{R}_1, \overline{R}_2, \overline{R}_3$ respectively. The handtools have been assumed to be built of seven types of primitives. They are:

L1 { (shape,rectangular),(size, medium),(color, black) }

L2 { (shape,rectangular),(size, medium),(color, yellow) }

L3 { (shape,rectangular),(size, small),(color, yellow) }

L4 { (shape,rect),(size, small),(color, black) }

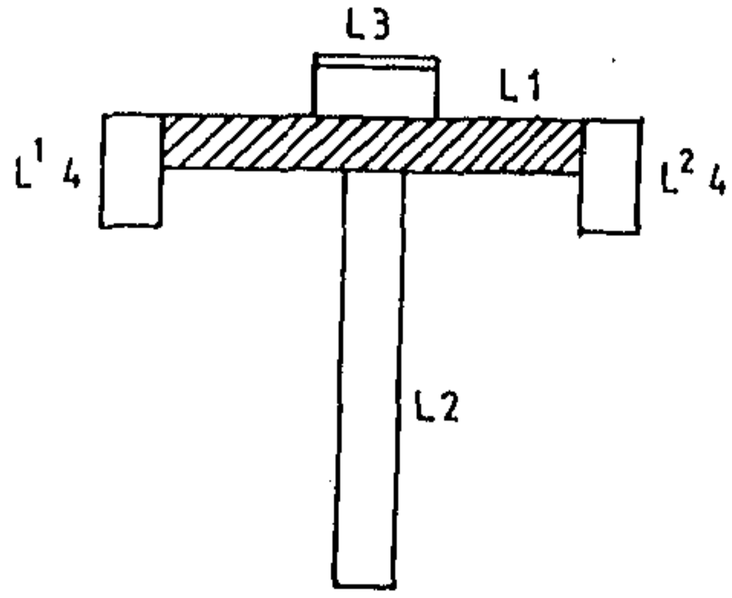
L5 { (shape,triangular),(size, small),(color, yellow)}

L6 { (shape,trapezium),(size,small), (color,white) }

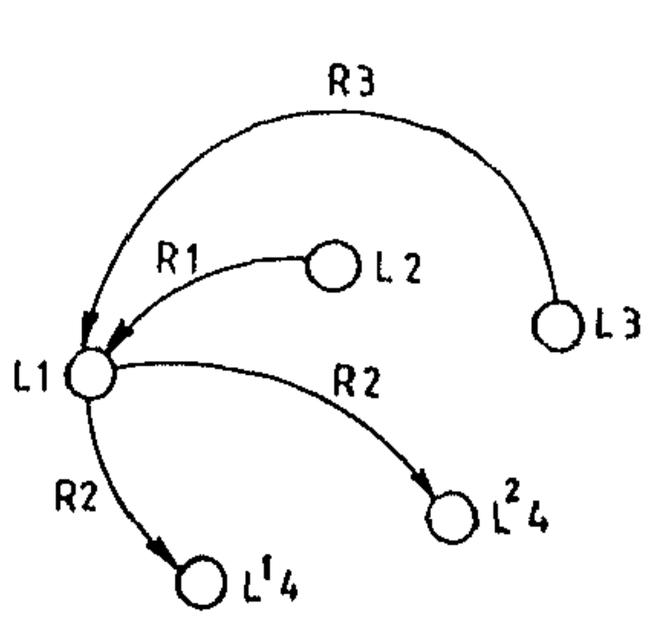
L7 { (shape,triangular), (size,small),(color,black) }

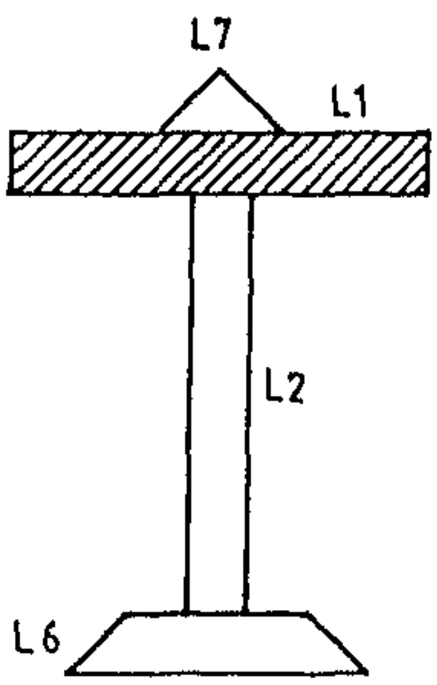
Individual prototype and test shapes and the corresponding attributed relational graphs are shown in Fig.3.2. In the figures and the tables, i^{th} instance of the j^{th} type of the primitive has been indicated as L^ij. But, single instance of a particular type of the primitive (say, k^{th} type) has been denoted simply as Lk. Dummy primitives have been indicated as L^D.

Hopfield network based matching scheme has been applied for finding the correct match of distorted and undistorted instances of the hammers among the stored prototypes. In the first set of experiments, we tried to find a match for the undistorted hammer1. We have found (Table 3.1 and Table 3.2) that, primitives of the hammer1 has matched with the correct counterparts of the stored model when simultaneous matching with all the models has been attempted. Only multiple non-zero neuronal output in the same row has been observed for different instances of the identical type of the primitives. Strength of the output of the neurons corresponding to the wrong matches decreased with the increase in the number of iterations (Table 3.2.). Matching sequentially with the individual prototypes have also yielded maximum number of matched primitives for the correct model but there are instances of false matches with the primitives of other models (Table 3.3). But, in simultaneous matching using multi-layered network there were no false matches with the primitives of the other objects because domain constraints had inhibited these mismatches. This is an important observation which indicates appropriateness of the present technique for matching even similar shapes. This technique was also used for finding match for distorted hammers (e.g., hammers with missing parts). These hammers are shown in Fig.3.3. Even with distorted hammers correct recognition results were obtained (Table 3.4 and Table 3.5).



HAMMER 1





HAMMER 2

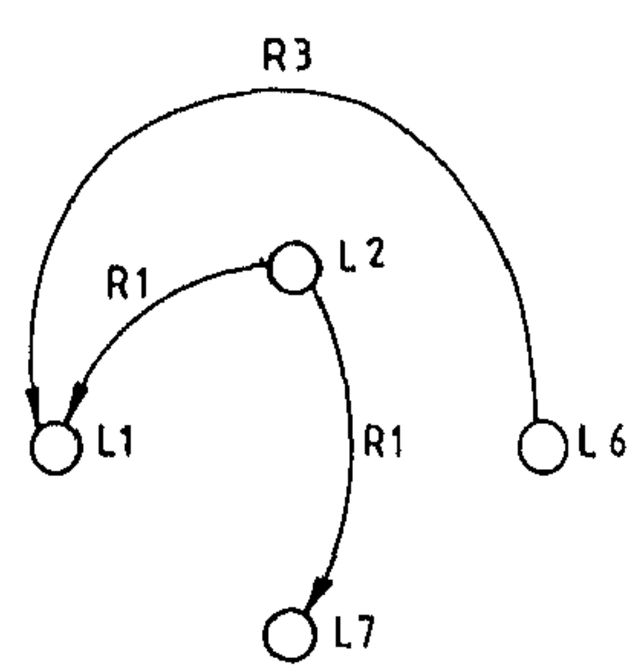
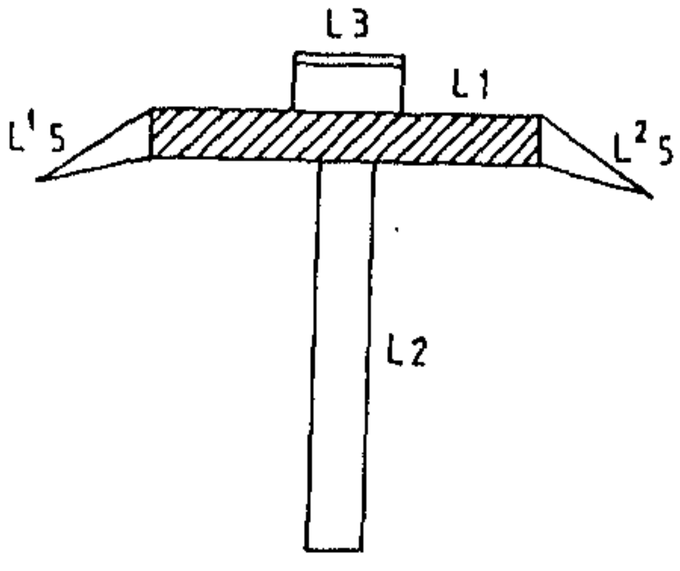
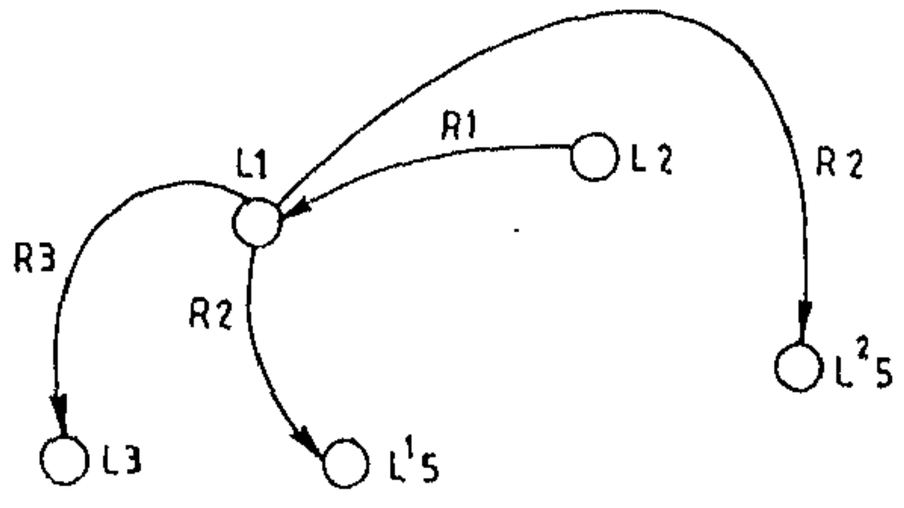
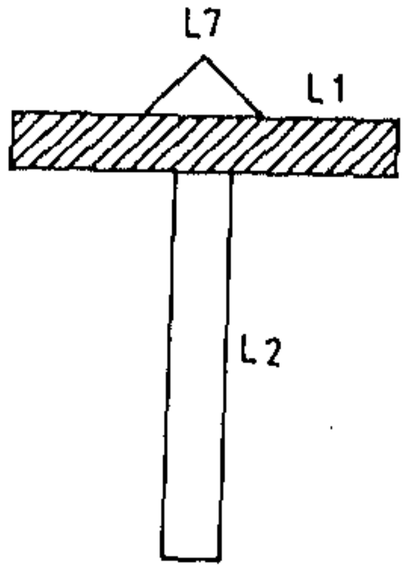


Figure 3.2. Cont'd.



HAMMER 3





HAMMER 4

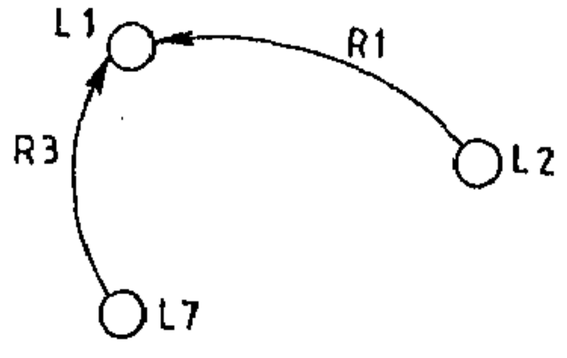
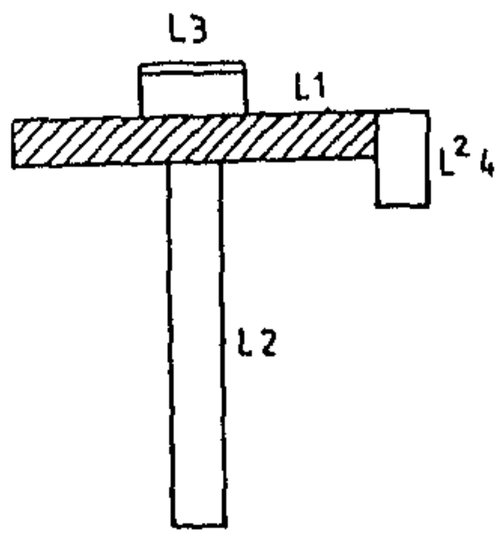
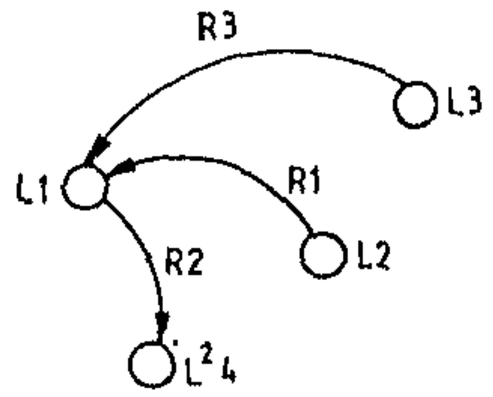
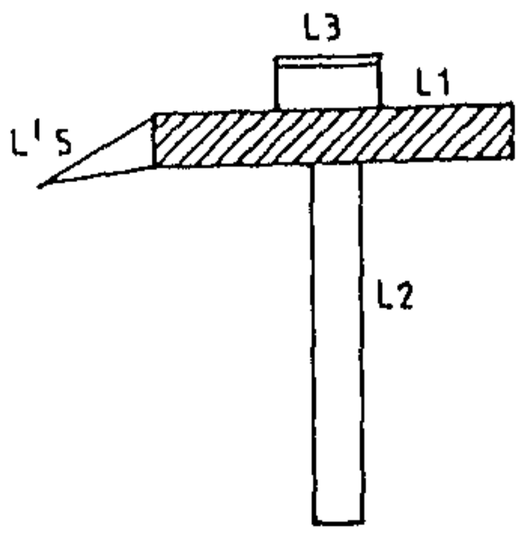


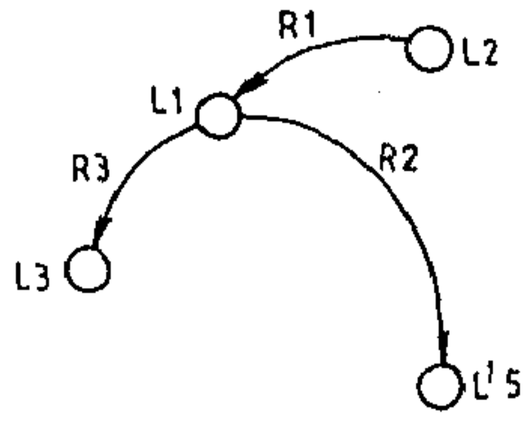
Figure 3.2: Model hand tools.





DISTORTED HAMMER 1





DISTORTED HAMMER 3

Figure 3.3: Distorted hand tools

Table 3.1: Matching hammer1 with multiple prototypes simultaneously (*after 50 iterations*)

candidate	prototype				
hammer1	*hammer1*				
	$L1$	$L2$	$L3$	L^14	L^24
$L1$	0.999	0.0	0.0	0.0	0.0
$L2$	0.0	1.0	0.0	0.0	0.0
$L3$	0.0	0.0	1.0	0.0	0.0
L^14	0.0	0.0	0.0	0.937	0.727
L^24	0.0	0.0	0.0	0.720	0.904

Match score for primitives of other shapes is zero.

Table 3.2: Matching hammer1 with multiple prototypes simultaneously (*after 100 iterations*)

candidate	prototype				
hammer1	*hammer1*				
	$L1$	$L2$	$L3$	L^14	L^24
$L1$	1.0	0.0	0.0	0.0	0.0
$L2$	0.0	1.0	0.0	0.0	0.0
$L3$	0.0	0.0	1.0	0.0	0.0
L^14	0.0	0.0	0.0	0.999	0.511
L^24	0.0	0.0	0.0	0.502	0.999

Match score for primitives of other shapes is zero.

Table 3.3: Matching Hammer1 with individual prototypes using single layered network (*150 iterations*)

candidate	prototype				
hammer1	*hammer1*				
	$L1$	$L2$	$L3$	L^14	L^24
$L1$	1.0	0.0	0.0	0.0	0.0
$L2$	0.0	1.0	0.0	0.0	0.0
$L3$	0.0	0.0	1.0	0.0	0.0
L^14	0.0	0.0	0.0	1.0	0.513
L^24	0.0	0.0	0.0	0.486	1.0

candidate	prototype				
hammer1	*hammer2*				
	$L1$	$L2$	$L6$	$L7$	L^D
$L1$	0.002	0.0	0.0	0.0	0.0
$L2$	0.0	1.0	0.0	0.0	0.0
$L3$	0.0	0.0	0.7	0.0	0.0
L^14	0.0	0.0	0.0	0.0	0.0
L^25	0.0	0.0	0.0	0.0	0.0

candidate	prototype				
hammer1	*hammer3*				
	$L1$	$L2$	$L3$	L^15	L^25
$L1$	1.0	0.0	0.0	0.0	0.0
$L2$	0.0	1.0	0.0	0.0	0.0
$L3$	0.0	0.0	1.0	0.0	0.0
L^14	0.0	0.0	0.0	0.0	0.0
L^25	0.0	0.0	0.0	0.0	0.01

candidate	prototype				
hammer1	*hammer4*				
	$L1$	$L2$	$L6$	L^D	L^D
$L1$	0.7	0.0	0.0	0.0	0.0
$L2$	0.0	1.0	0.0	0.0	0.0
$L3$	0.0	0.0	0.999	0.0	0.0
L^14	0.0	0.0	0.0	0.0	0.0
L^25	0.0	0.0	0.0	0.0	0.0

Table 3.4: *Matching Distorted hammer1 (primitive $L^1 4$ missing) with multiple prototypes simultaneously - Case I (after 50 iterations)*

candidate *distorted hammer1*	prototype *hammer1*				
	$L1$	$L2$	$L3$	$L^1 4$	$L^2 4$
$L1$	0.507	0.0	0.0	0.0	0.0
$L2$	0.0	.999	0.0	0.0	0.0
$L3$	0.0	0.0	0.999	0.0	0.0
$L^2 4$	0.0	0.0	0.0	0.0	0.999
L^D	0.0	0.0	0.0	0.999	0.0

Match score for other shapes is zero

Table 3.5: Matching Distorted hammer3 (primitive $L^2 5$ missing) with multiple prototypes simultaneously - Case I (*after 50 iterations*)

candidate *distorted hammer3*	prototype *hammer3*				
	$L1$	$L2$	$L3$	$L^1 5$	$L^2 5$
$L1$	0.33	0.0	0.0	0.0	0.0
$L2$	0.0	.999	0.0	0.0	0.0
$L3$	0.0	0.0	0.999	0.0	0.0
$L^1 8$	0.0	0.0	0.0	0.981	0.959
L^D	0.0	0.0	0.0	0.0	0.723

Match score for other shapes is zero

3.5.2 Recognition of Bengali Characters

Structural descriptions of the characters encode spatial relations between the individual primitives. Present technique of matching structural shape descriptions can be therefore applied for the character recognition problems. A survey on various character recognition approaches is presented in [224]. A brief discussion on different connectionist approaches for character recognition is presented in Chapter 6. Here, we like to present the effectiveness of the present scheme for matching different structural descriptions of the characters. As a case study we have considered the problem of recognition of Bengali characters. Bengali characters have got features distinct from general Devnagari scripts. In general, as evident from Fig.3.4, spatial constraints between straight or curved segments which meet at junctions characterize different Bengali characters. This features can be effectively exploited by defining appropriate types of the spatial relations and these descriptions can be used for matching with Hopfield network.

For the problem of Bengali character recognition the types of the spatial relations considered are:

$\mathcal{T}_1(X,Y)$ X is connected to Y end-to-end and makes an angle less than or equal to 90 degree; X is to the left of Y.

$\mathcal{T}_2(X,Y)$ The end of Y is connected to any point in the right half of X and Y makes an angle less than 90 degree with the left half of X.

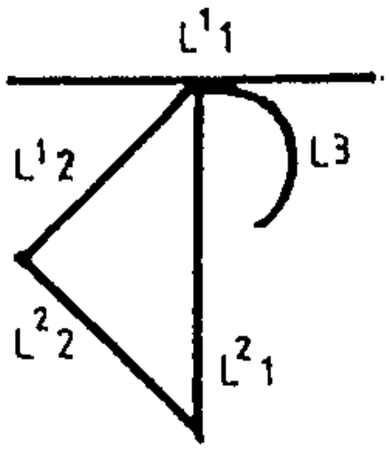
$\mathcal{T}_3(X,Y)$ The end of Y is connected to any point in the right half of X and makes an angle of approximately 90 degree.

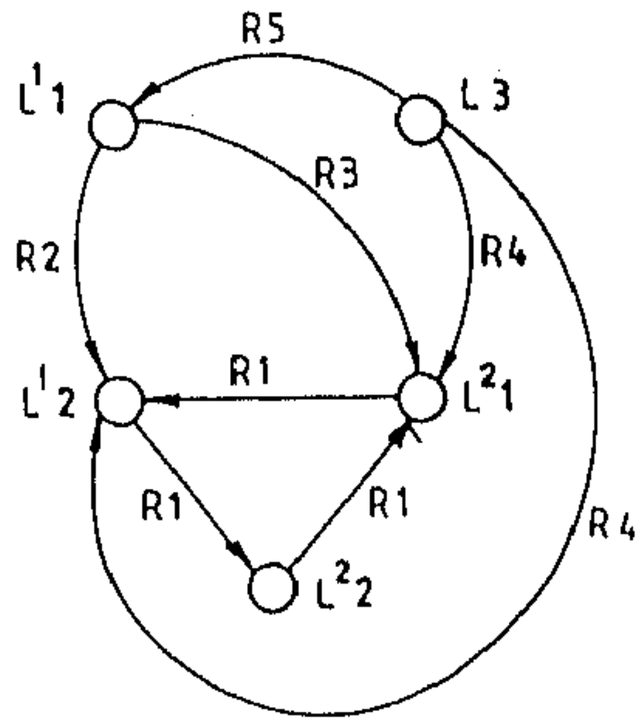
$\mathcal{T}_4(X,Y)$ The end of X is connected to end of Y and angle between them is less than 90 degree.

$\mathcal{T}_5(X,Y)$ The end of X is connected to any point in the right half of Y and the angle between them is less than 90 degree. and convex part of X faces right half of Y.

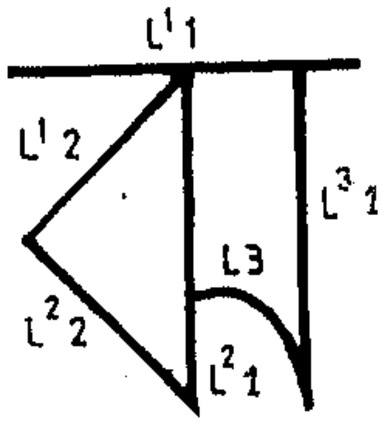
$\mathcal{T}_6(X,Y)$ Same as that of $\mathcal{T}_4$ except that angle between X and Y is greater than 90 degree.

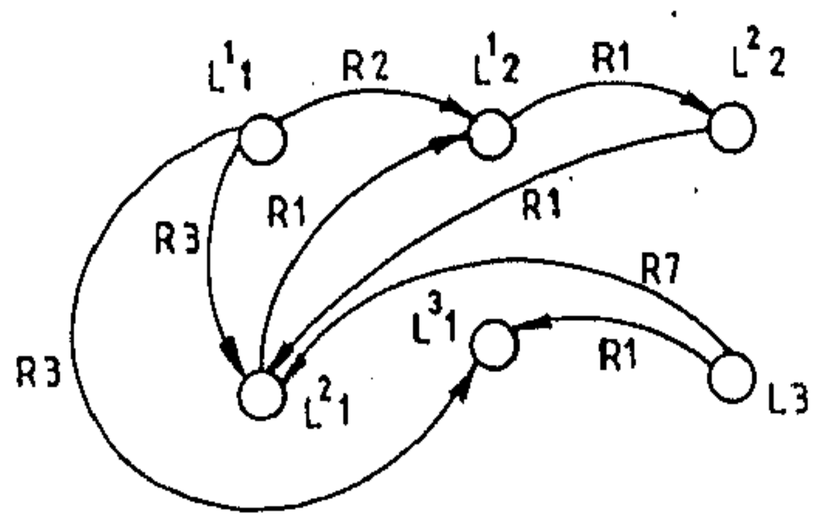
BCHAR 1



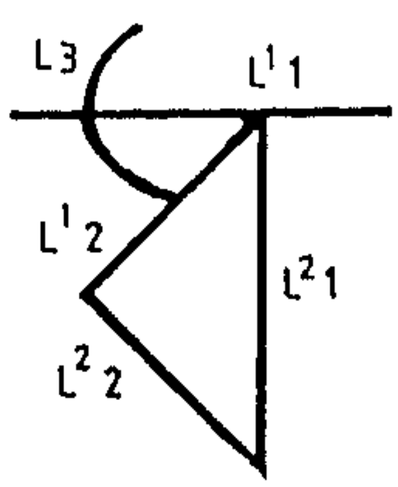


BCHAR 2





BCHAR 3



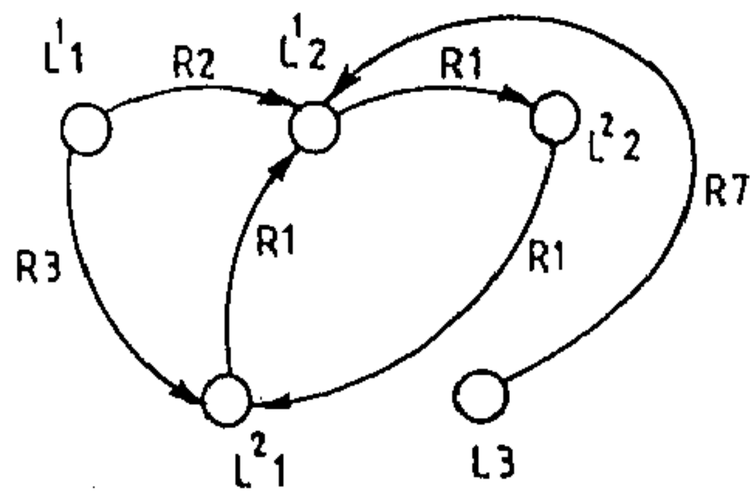
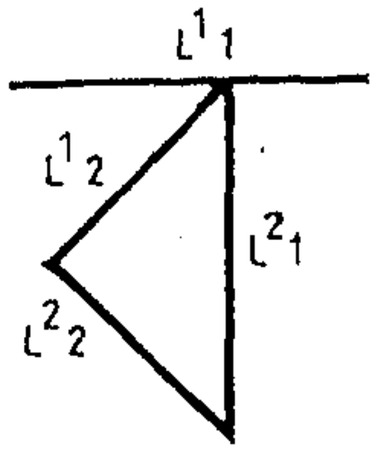
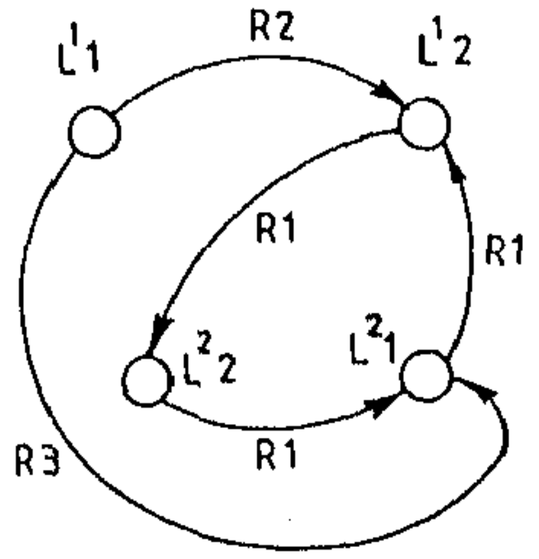
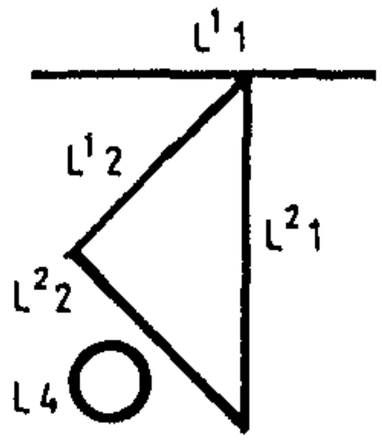


Figure 3.4. Cont'd.



BCHAR 4





BCHAR 5

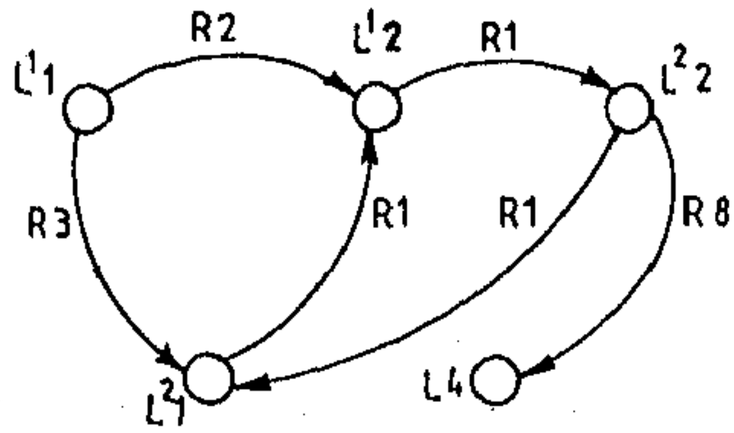


Figure 3.4: Bengali characters.

Corresponding to these types of spatial relations appropriate relational labels and their duals can be specified in a manner similar to that in the previous section. The characters have been found to be built with following primitives:

L1 { (shape, linear), (size, large) }

L2 { (shape, linear), (size, medium) }

L3 { (shape, arc), (size, medium) }

L4 { (shape, circular), (size, small) }

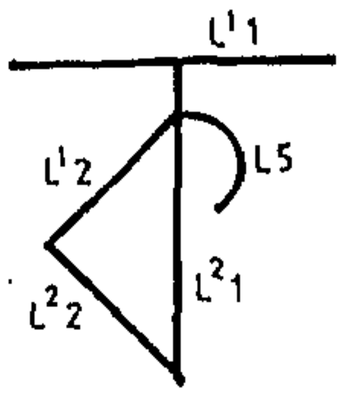
The prototype characters and their models have been shown in Fig.3.4.

In this case, we tried to find the match for a known character (BCHAR1) in a set of characters specified in Fig.3.4. With the multilayered network architecture we got the best results (Table 3.6). Also, matching experiments were done with the distorted samples of BCHAR1 and BCHAR2 (shown in Fig.3.5). In these distorted samples individual primitives were modified. As a result, two additional types of primitives were found in these samples. They are:

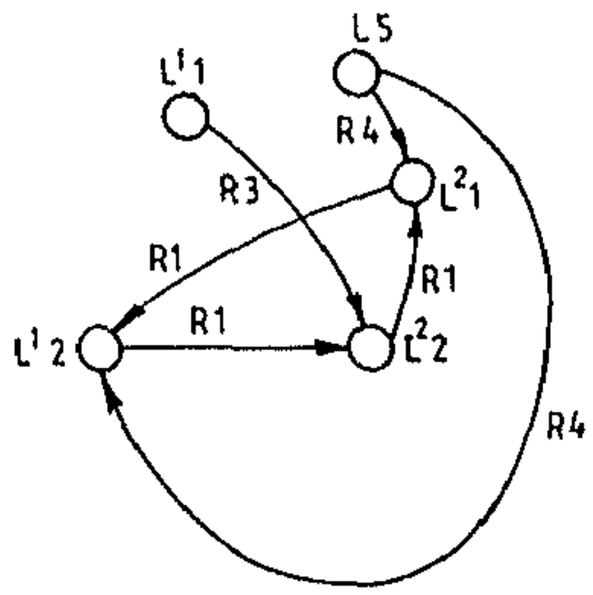
L5 { (shape, arc), (size, small) }

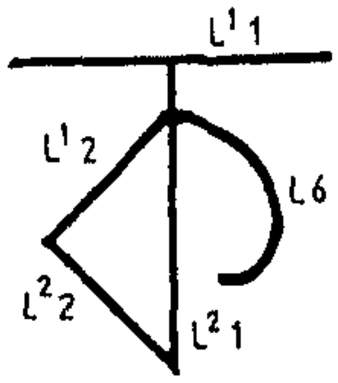
L6 { (shape, arc), (size, large) }

In the test examples, because of modification in the shape of the character's primitives some of the spatial relations between the primitives were changed. Results show that, even though the characters considered were very similar, network has correctly found the matches despite distortions (Tables 3.7-3.10). In this case also multilayered network was used and the representation and matching scheme is found to be successful in the recognition of characters.

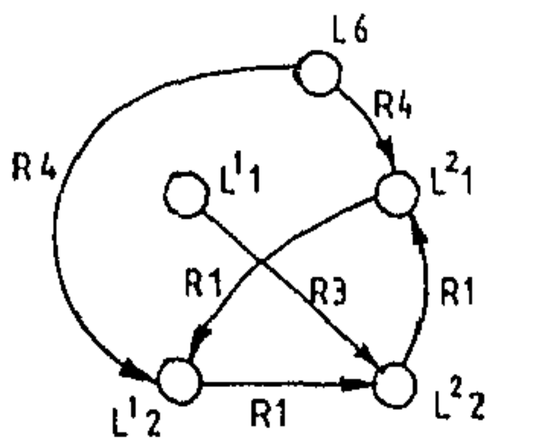


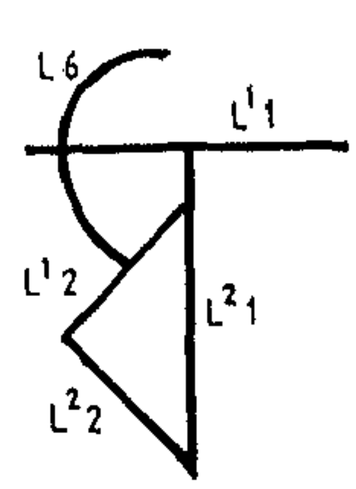
FIRST DISTORTED
BCHAR 1



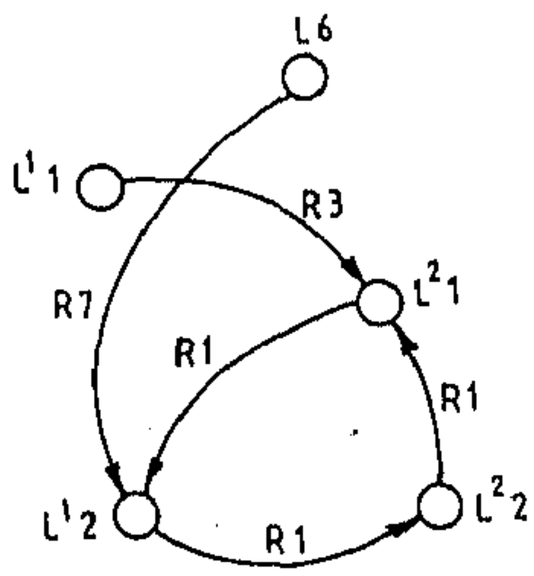


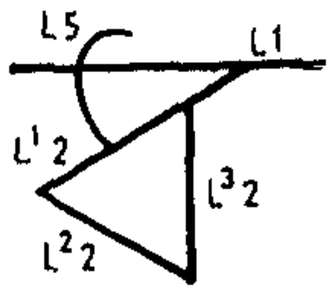
SECOND DISTORTED
BCHAR 2





FIRST DISTORTED
BCHAR 3





SECOND DISTORTED
BCHAR 3

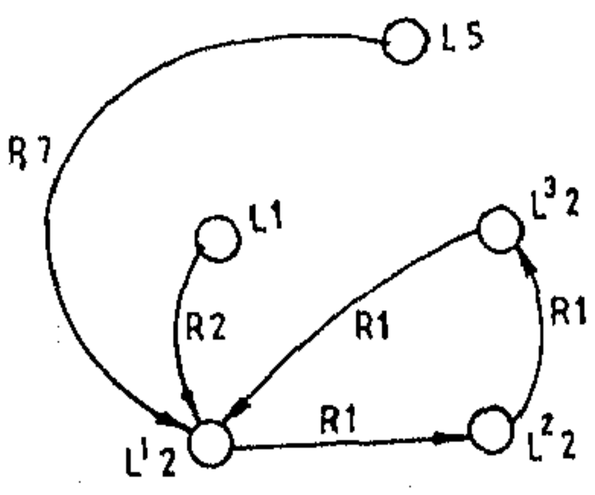


Figure 3.5: Distorted Bengali characters.

Table 3.6: Matching Bengali character BCHAR1 with all the characters simultaneously (*after 100 iterations*)

candidate *BCHAR1*	prototype *BCHAR1*						*BCHAR2*					
	L^11	L^21	L^12	L^22	$L3$	L^D	L^11	L^21	L^31	L^12	L^22	$L3$
L^11	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
L^21	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
L^12	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
L^22	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$L3$	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
L^D	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0

BCHAR1	*BCHAR3*						*BCHAR4*					
	L^11	L^21	L^12	L^22	$L3$	L^D	L^11	L^21	L^12	L^22	L^D	L^D
L^11	0.0	0.0	0.0	0.0	0.0	0.0	0.88	0.0	0.0	0.0	0.0	0.0
L^21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.53	0.0	0.0	0.0	0.0
L^12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
L^22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$L3$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
L^D	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

BCHAR1	*BCHAR5*					
	L^11	L^21	L^12	L^22	$L4$	L^D
	L1	L2	L3	L4	L5	L6
L^11	0.0	0.0	0.0	0.0	0.0	0.0
L^21	0.0	1.0	0.0	0.0	0.0	0.0
L^12	0.0	0.0	1.0	0.0	0.0	0.0
L^22	0.0	0.0	0.0	0.0	0.0	0.0
$L3$	0.0	0.0	0.0	0.0	0.0	0.0
L^D	0.0	0.0	0.0	0.0	0.0	0.0

Table 3.7: Match of first distorted BCHAR1 (*after 150 iterations*)

candidate	prototype					
TEST BCHAR1	*BCHAR1*					
	$L^1 1$	$L^2 1$	$L^1 2$	$L^2 2$	$L3$	L^D
$L^1 1$	1.0	0.0	0.0	0.0	0.0	0.0
$L^2 1$	0.0	1.0	0.0	0.0	0.0	0.0
$L^1 2$	0.0	0.0	1.0	0.0	0.0	0.0
$L^2 2$	0.0	0.0	0.0	1.0	0.0	0.0
$L5$	0.0	0.0	0.0	0.0	0.0	0.0
L^D	0.00	0.0	0.0	0.0	0.0	1.0

Match score is zero for primitives of all other characters

Table 3.8: Match of second distorted BCHAR1 (*after 150 iterations*)

candidate	prototype					
TEST BCHAR1	*BCHAR1*					
	$L^1 1$	$L^2 1$	$L^1 2$	$L^2 2$	$L3$	L^D
$L^1 1$	1.0	0.0	0.0	0.0	0.0	0.0
$L^1 1$	0.0	1.0	0.0	0.0	0.0	0.0
$L^1 2$	0.0	0.0	1.0	0.0	0.0	0.0
$L^2 2$	0.0	0.0	0.0	1.0	0.0	0.0
$L6$	0.0	0.0	0.0	0.0	0.0	0.0
L^D	0.00	0.0	0.0	0.0	0.0	1.0

Match score is zero for primitives of all other characters

Table 3.9: Match of first distorted BCHAR3 (*after 150 iterations*)

candidate	prototype					
TEST BCHAR3	*BCHAR3*					
	$L^1 1$	$L^2 1$	$L^1 2$	$L^2 2$	$L3$	L^D
$L^1 1$	1.0	0.0	0.0	0.0	0.0	0.0
$L^2 1$	0.0	1.0	0.0	0.0	0.0	0.0
$L^1 2$	0.0	0.0	1.0	0.0	0.0	0.0
$L^2 2$	0.0	0.0	0.0	1.0	0.0	0.0
$L6$	0.0	0.0	0.0	0.0	0.873	0.0
L^D	0.00	0.0	0.0	0.0	0.0	1.0

Match score is zero for primitives of all other characters

Table 3.10: Match of second distorted BCHAR3 (*after 150 iterations*)

candidate	prototype					
TEST BCHAR3	*BCHAR3*					
	$L^1 1$	$L^2 1$	$L^1 2$	$L^2 2$	$L3$	L^D
$L^1 1$	1.0	0.0	0.0	0.0	0.0	0.0
$L^3 2$	0.0	0.032	0.0	0.0	0.0	0.0
$L^1 2$	0.0	0.0	1.0	0.0	0.0	0.0
$L^2 2$	0.0	0.0	0.0	1.0	0.0	0.0
$L5$	0.0	0.0	0.0	0.0	0.0	0.0
L^D	0.00	0.0	0.0	0.0	0.0	1.0

Match score is zero for primitives of all other characters

3.6 Conclusions and Discussion

A method for matching structural descriptions using Hopfield network has been presented. The matching scheme is based on an appropriate formulation of the energy function for the Hopfield network. This energy function is designed to accommodate possibilities of partial mismatches between the primitives and spatial relations. Also, differential criticality factor can be associated with attributes of the primitives and spatial relations. A graph-theoretic transformation procedure has been presented for transforming structural descriptions containing asymmetric spatial relations so that the Hopfield network, constructed with the transformed descriptions, will not have asymmetric interconnection weights. Consequently, the present scheme can be applied (unlike the other related methods [223, 162, 164]) for general structural description matching problems involving realistic asymmetric spatial relations like *right,left, on, above* etc.

Although the effectiveness of the method is demonstrated for matching handtools and characters, it can also be adopted for developing recognition systems in other application domains like robot vision, document processing and navigation etc. Here, the automatic extraction of primitives and learning the saliencies (degree of importance) of different primitives are not considered. These issues are discussed in Chapter 6. Note that, the number of neurons necessary to represent a suitable match for a candidate structure (with N primitives) between a set of M prototype structures (each with at most N primitives) is $M \times N^2$. As a result, the required number of neurons become very high for large value of M, i.e, for large number of prototypes. ♠

In this chapter, we have developed a method using a basic neural network (Hopfield) for finding out the match between candidate and prototype structures. In the following chapters, we will be concentrating on the development of some application-specific models for learning and simultaneous recognition of multiple objects/categories.

Chapter 4

X-tron : Supervised Mixed Category Perception

4.1 Introduction

Let us now present a new kind of model for mixed category perception (operated under supervised mode) [207], [208]. The model is named as 'X-tron' where X stands for some unknown category. The model is also extended to exhibit self-organization behavior in Chapter 5. To consider practical problems like missing of the features due to occlusion, formation of new features (e.g., concave corners formed due to the overlap of two objects), and after all for efficient representation of the features, X-tron is suitably employed to design some application-specific connectionist systems in Chapter 6 and Chapter 7.

The principle of mixed category perception has been formulated on the basis of similarity-based induction hypothesis (memory based reasoning) as posed by Stanfill and Waltz [225]. X-tron, a three-layered model, accepts a numerical feature vector as input, and generates an output vector indicating the degree of confidence about the presence of objects/categories by interpreting the input feature vector using iterative verification process. In order to enable the network to automatically capture the saliencies of the features, new learning rules are also designed based on some predetermined probabilistic measures.

Note that, the problem we considered is not finding out a pattern embedded in different patterns, as performed by other related investigations (Cohen and Grossberg [147], [148] or Nigrin [149], [150], [151], [152]). Rather, the network is able to decide if more than one learned category is present in the input even when there is a significant amount of overlapping between the patterns. For example, suppose that the model has learned two categories *cab* and *abd*. In that case, if a new pattern *cabd* is presented to the network, it would be able to decide that the new pattern is a mixture of these two learned categories.

The rest of this chapter is organized as follows. The problem of mixed category perception has been mathematically formulated in Section 4.2. The network architecture and its dynamical behavior is described in Section 4.3. Section 4.4 presents the supervised learning rules used for X-tron along with their convergence. An estimation of the total number of nodes that the network can have is provided in Section 4.5. Section 4.6 presents some experimental results and analysis of the behavior of X-tron with overlapped binary strings and visual patterns. Finally, Section 4.7 comes up with the concluding remarks.

4.2 Description of the Problem and Formulation

The objective of this section is to provide a mathematical formulation for identifying the categories (or object classes) from a given set of input features. The problem can be looked upon as a similarity-based induction hypothesis as posed by Stanfill and Waltz [225]. The goal of this hypothesis is to make decisions by looking for object classes in the given feature set. Informally it can be said that if an object class be present then the features of the object class should be present [1]. Each feature can be associated to more than one object class. Initially, each feature should activate its corresponding object classes (formation of initial hypotheses). On the other hand, each object class should support its constituent features [2]. If a valid feature (present in a scene) do not get proper support from its corresponding object classes, it would strengthen the decision about the presence

[1] In practical situation, occlusion may result in loss of feature. That is taken care of in actual formulation

[2] The word support is used lucidly. The concrete formulation is given later

of its corresponding object classes more to receive proper support. On the other hand, if a feature, which is not really present, gets a substantial support, it would try to weaken the decisions about the presence of its corresponding object classes. By this method of iterative reasoning the proper decision about the presence or absence of the object classes can be performed.

The feature set can be represented as an input confidence vector, where each confidence value is the confidence about the presence of the corresponding feature. In the same way the set of objects can be represented as an output confidence vector. The confidence values can be discrete, i.e., 0 or 1, or continuous in the interval [0,1]. The formulation of the problem for two different situations are discussed in the following subsections.

4.2.1 Formulation for Discrete Confidence Values

Let us assume that at most n features can appear in the input, and there are at most m possible output object classes. Let the entire set of features be $\{f_1, f_2, \ldots, f_n\}$, and the set of output objects be $\{o_1, o_2, \ldots, o_m\}$. The feature set for each object can be represented as an input confidence vector

$$\mathbf{c} = [c_1, c_2, \ldots, c_n],$$

, where $c_i \in \{0, 1\}$ represents the confidence about the presence of feature f_i, i.e.,

$$c_i = \begin{cases} 0 & \text{if the feature } f_i \text{ is absent} \\ 1 & \text{if the feature } f_i \text{ is present} \end{cases}$$

Similarly, the set of objects can be described in terms of an output confidence vector given as

$$\mathbf{c^o} = [c_1^o, c_2^o, \ldots, c_m^o],$$

where $c_j^o \in \{0, 1\}$ represents the confidence about the presence of object o_j, i.e.,

$$c_j^o = \begin{cases} 0 & \text{if object } o_j \text{ is absent} \\ 1 & \text{if object } o_j \text{ is present} \end{cases}$$

Let F_j be the feature set associated with the object class o_j or in other words it can be stated that if any object in the object class o_j be present in the scene then all

the features in the set F_j are expected to be present. Therefore the set $F = \bigcup_{j=1}^{m} F_j$ represents the complete set of features under consideration. Let the association of a feature with an object class be expressed by a matrix **a** such that,

$$a_{li} = \begin{cases} 1 & \text{if } f_i \in F_l \\ 0 & \text{if } f_i \notin F_l \end{cases}$$

Under ideal noiseless condition, all the features associated with an object o_i[3] (i.e., all the elements of the set F_i) will be present in the input whenever the object o_i is expected to be present.

Confidence of an object is determined by the confidence of the features associated with it. For this purpose the different combinations of the input confidences and the object-feature associations are to be considered. Consider the following two subsets of objects given below.

$$\begin{aligned} A_{i1} &= \{o_l : \text{such that } a_{li} = 0\} \\ A_{i2} &= \{o_l : \text{such that } a_{li} = 1\} \end{aligned}$$

Note that there is no association of the objects with the feature f_i in the set A_{i1} (since $a_{li} = 0$), and therefore the output confidence values of the objects in the set A_{i1} do not depend on c_i. Similarly, in the set A_{i2}, the objects are fully associated with the feature f_i (since $a_{li} = 1$), and thus if any object in the set A_{i2} be present then the feature f_i must be present. Therefore if $c_i = 0$ and $o_l \in A_{i2}$ then $c_l^o = 0$ for all l. Again if $c_i = 1$, then it is obvious that at least one element of the set A_{i2} must be present, i. e. $A_{i2} \neq \phi$. So the sets A_{i1} and A_{i2} actually determines which objects will be present in the scene.

Let q features be absent in the input train of binary features such that $c_{i1} = c_{i2} = \ldots = c_{iq} = 0$, then, according to above discussion, the confidence values of the objects in the sets $A_{i12}, A_{i22}, \ldots, A_{iq2}$ must be zero. Thus a set A_ϕ is defined such that $A_\phi = \bigcup_{t \in (i1,i2,\ldots,iq)} A_{t2}$, and it represents the set of objects with zero confidence values. Then note that if A is the set of all objects then $A_r = A - A_\phi$ represents the set of plausible objects to be present in the scene. Now let the s features, present in the input train, be such that $c_{j1} = c_{j2} = \ldots = c_{js} = 1$. Then the set of all possible objects present in the scene will be given as

$$A_p = (A_{j12} \bigcup A_{j22} \bigcup \ldots \bigcup A_{js2}) \bigcap A_r$$

[3]Henceforth object classes would be referred as objects.

Using this technique, the possible output object hypotheses are determined by the input feature set, and they cross-verify the features by generating the support to the features. Consequently, the confidence of any feature should be the maximum of all the confidences of the objects to which the feature is associated. Mathematically,

$$\max_{i=1,\ldots,m} c_l^o a_{li} = c_i^{'} \quad \text{for all} \;\; i, 1 \leq i \leq n$$

where $c_i^{'}$ is the confidence of feature i due to different object hypotheses. Observe that $c_i = c_i^{'}$ for all i when there is no noise.

The formulation, just described, finds out the set of all possible object hypotheses for a given input feature set by simple logical reasoning. But the formulation does not deal with the following prominent points. First, it is very sensitive to noise because if confidence of any input feature becomes zero due to presence of noise, it eliminates all possible object hypotheses corresponding to it. Second, the method gives the complete set of possible objects. Possibly the input feature train could have been interpreted with a proper subset of objects. But the above method does not essentially find out the minimal set. Third, it is assumed that the features are present in the scene with either zero or full confidence which is very ideal and may not be always possible in practical cases. To avoid these problems, we have to consider continuous confidence values in the interval of [0,1], and find out the object set with as minimum number of objects as possible to interpret the given input feature vector.

4.2.2 Formulation for Continuous Confidence Values

In this section we are considering that the features can suffer from noise, and therefore the confidence values will not be perfectly 0 or 1, rather they will be in the interval [0,1]. Thus the confidence values here, to some extent, represent the vagueness of the knowledge about the presence of the features. If the confidence $c_i = 0.5$, it indicates a don't know condition about the presence or absence of the feature. And if $c_i > 0.5$, then the possibility of the feature f_i to be present is more than the possibility that it is absent. The confidences about the presence of output objects also take continuous values in the interval [0,1]. Hence the confidences of the input features will not be perfectly matched to the support from the output objects. The recognition problem in this case may be solved by finding the output confidence values such that the error between the input confidences and the support

from the output objects is minimized. A function quantifying such error (both for discrete and continuous values) can be defined as

$$E_1 = 1/2 \sum_{i=1}^{n} \left(\max_{l=1,\ldots,k} c_l^o a_{li} - c_i \right)^2 . \tag{4.1}$$

Note that E_1 measures the square of the difference of the input confidence and the feedback support over the entire feature set. The output confidence vector which minimizes E_1, represents a possible solution where elements with high confidence values correspond to the object classes which are supported by and also support the input feature vector. However, such a solution may correspond to a superset of object classes because a feature can support and as well as get support from more than one object. To interpret the feature vector with minimum number of objects, another factor (E_2) is added with E_1, where

$$E_2 = 1/2 \sum_{l=1}^{m} (c_l^o)^2. \tag{4.2}$$

The total optimization function that is to be minimized can be taken as

$$E = \kappa_1 E_1 + \kappa_2 E_2, \tag{4.3}$$

where κ_1 and κ_2 are two constants which determine the relative importances of E_1 and E_2. In situations like English character recognition more than one character cannot occur together, and therefore κ_2 should be large. On the other hand in industrial parts recognition, more than one parts are presented frequently, and thereby κ_2 should be kept small[4]. The actual form of optimization function becomes

$$E = \kappa_1/2 \sum_{i=1}^{n} \left(\max_{l=1,\ldots,k} c_l^o a_{li} - c_i \right)^2 + \kappa_2/2 \sum_{l=1}^{m} (c_l^o)^2. \tag{4.4}$$

Although the global minima of E may provide the optimal solution, an acceptable suboptimal solution may be obtained by finding out a local minima. A local minima of E can be found by using gradient descent technique, i.e.,

$$\Delta c_l^o = -\frac{\partial E}{\partial c_l^o}. \tag{4.5}$$

[4]Here the large and small are used only to bring the sense, and not how to select them. It will be shown in the next section that κ_1 and κ_2 are absorbed in the network parameters, and the selection of parameters are described later.

The solution obtained by this gradient descent technique will at least be a sub-optimal one. Thus the solution does not really gives the minimal set of objects interpreting the input features, rather it gives a irredundant set of objects covering all the features. Equation (4.5) can be written as

$$\Delta c_l^o = \kappa_1 \sum_{i=1}^{n} e_{il} - \kappa_2 c_l^o \tag{4.6}$$

where

$$e_{il} = \begin{cases} (c_i - c_l^o a_{li}) a_{li} & \text{if } c_l^o a_{li} \geq c_m^o a_{mi} \quad \forall m \neq l \\ 0 & \text{otherwise} \end{cases}$$

Equation (4.6) can be physically interpreted as follows. Consider every feature is getting support from the object classes to which the feature belongs. The value of a_{li} will be 1 if the feature f_i belongs to the object o_l, and will be 0 if it does not. Therefore, e_{il} measures the difference of the input confidence of the i^{th} feature and the maximum support it is getting from the objects. If the input confidence of a feature is higher than the maximum support then it increases the confidence of the output object producing the maximum support, and will not affect the confidences of other objects (since e_{il} is zero if the maximum support is not from the l^{th} object). Similarly, if the maximum support to any feature is higher than its *input confidence*, then the output confidence of the object producing the maximum support will be decreased. The nature of activation (increase or decrease) will be determined by the factor e_{il}. The Equn. (4.6) shows that the amount of support (positive or negative) from the features to the objects are multiplied by a_{li}. Due to the presence of the constraint of minimum number of objects, the confidences of all output objects are negated by self-feedback, the amount of which is governed by the factor κ_2. The value of κ_2 should be selected small enough so that it does not reduce the output confidences to a great extent, and at the same time it should be large enough to ensure the minimum number of objects in the output.

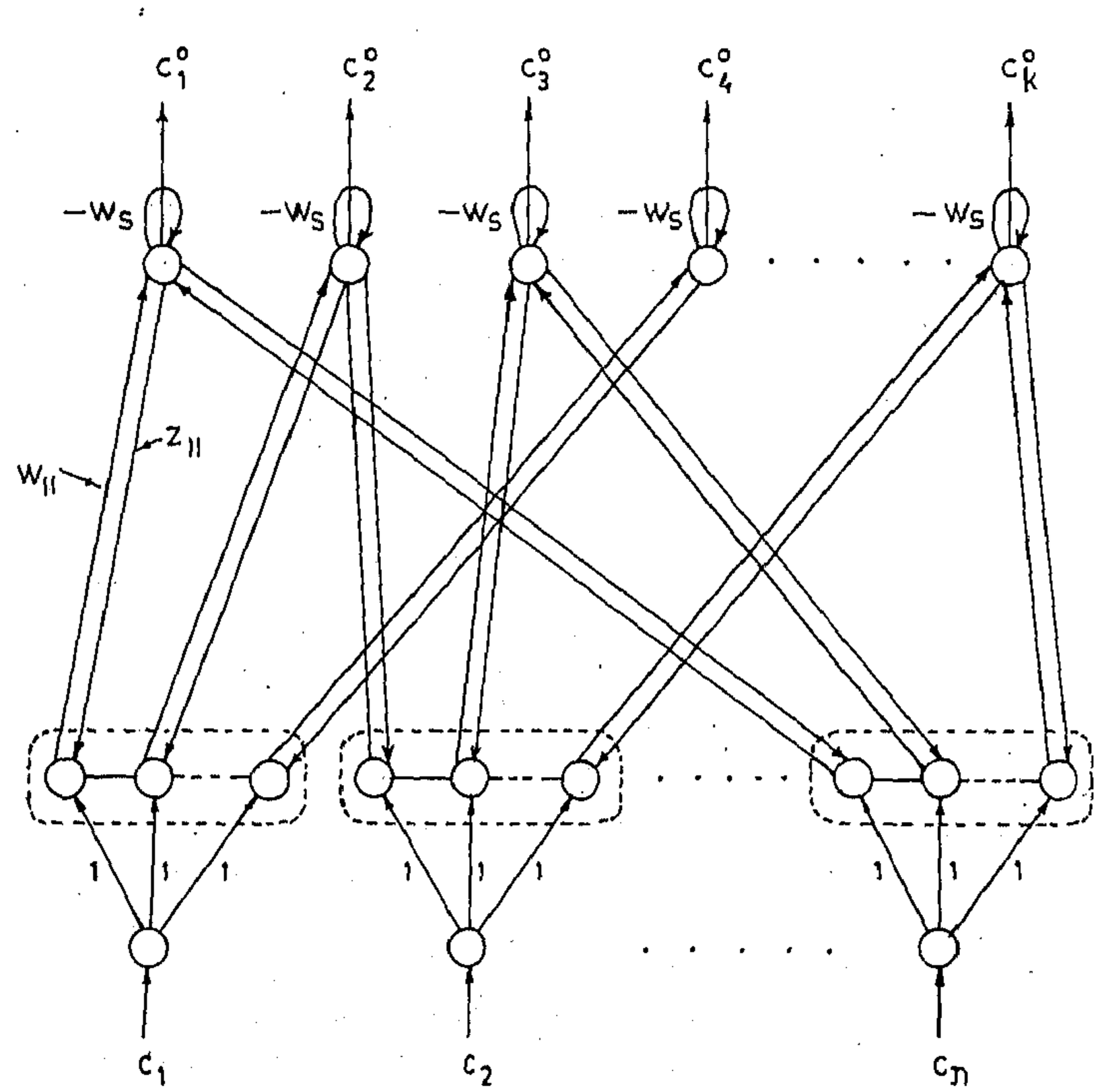
4.3 Neural Network Model

In the methodology described in Section 4.2.2, the confidence values of the objects are iteratively increased or decreased depending on the support from the features. Note that at each step the change in the confidence value of one object does not

depend on that of the other objects. As the process of updating confidence values of the objects are inherently parallel, this iterative process can be successfully implemented on a neural network (or connectionist model), where the node activations totally depend on the local computations. A network model, namely X-tron, has been designed for this particular task. The model structure and updating of states of the nodes are described in the following subsections.

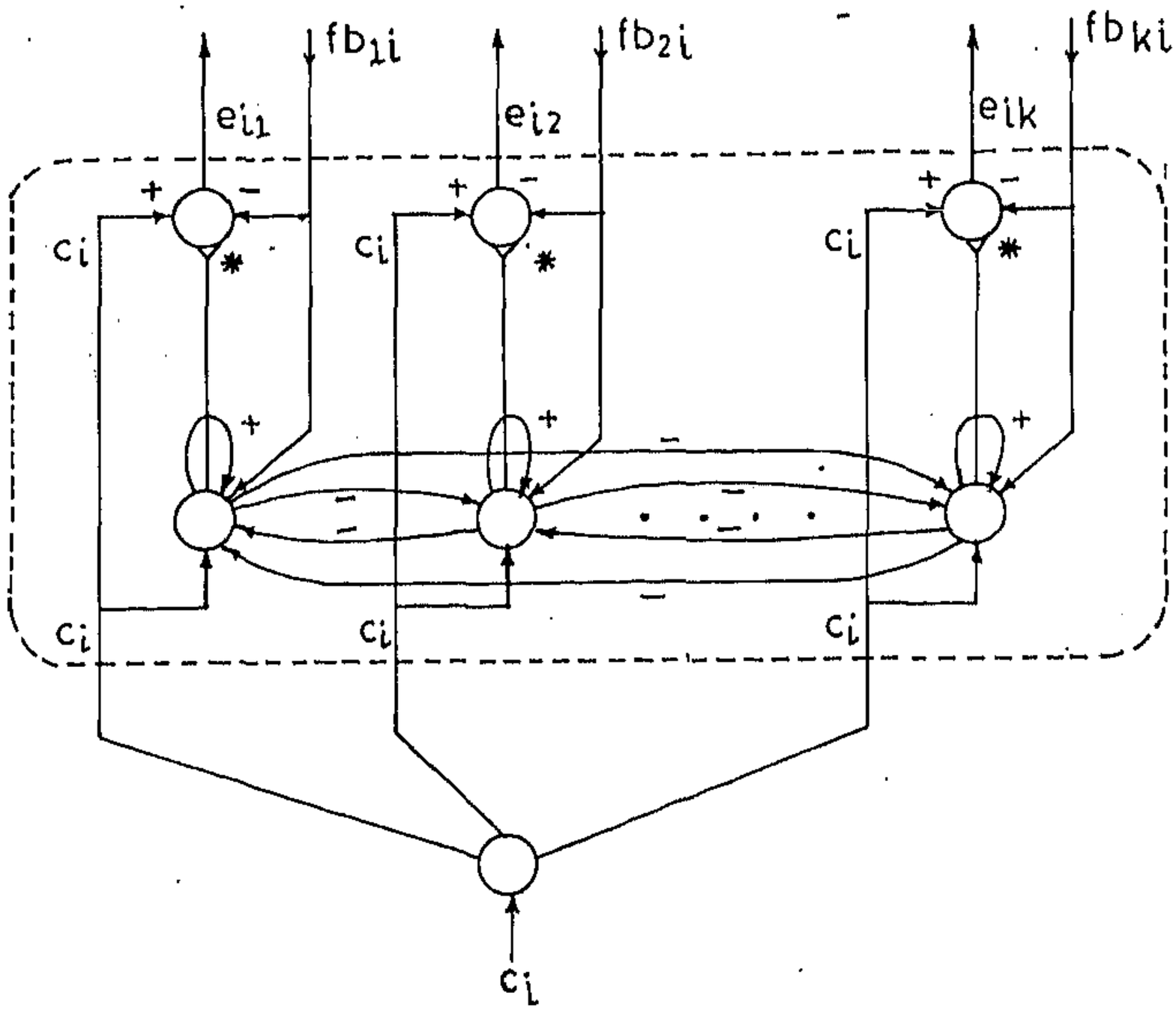
4.3.1 Structure of the Connectionist Model

The structure of X-tron, operated under supervised mode, is shown in Fig.4.1. The network is a three layered model, where the output layer consists of nodes whose activation values represent the confidences about the presence of output objects. The total number of output nodes denote the maximum number of objects the network is able to recognize. In the input layer the node activations represent the confidence values about the presence of the features, and the number of input nodes indicate the maximum dimension of the input feature vector. The feature-object association matrix [a] is embedded in the connection strengths in the network. From Equn. (4.6) it is clear that the supports of a feature to different objects are different, and this depend on the feedback support the feature receives from the objects. To incorporate this differential support (e_{il}) into the network, a set of hidden nodes is associated with each input node. All these nodes constitute the hidden layer lying between the input and output layer. Each hidden node is connected to exactly one input node and exactly one output node, and plays the key role of associating the input node with the output node. The output nodes are actually instantiated from the input layer only through the hidden nodes, and there is no direct connection from the input layer to the output layer. Each hidden node is connected to single output node through two kinds of links. The activations from the hidden nodes proceed to the output nodes through the *bottom-up* links, and the activations of the output nodes are fed back to the hidden nodes through the *top-down* links. There is exactly one unidirectional link from each input node to each hidden node associated with it, and the signal can flow from the input nodes to the hidden nodes but not in the opposite direction. The unidirectional links from the input layer to the hidden layer are of fixed weights, and here the weights are set as unity. Each output node is associated with a negative self-feedback. All the hidden nodes connected to a common input node have lateral inhibitory connections among themselves and each one has a self-excitatory feedback.



(a)

Figure 4.1 cont'd.



(b)

Figure 4.1: (a)Structure of the connectionist model for supervised learning and recognition. The hidden nodes connected to the same input node are enclosed in a dashed box. (b)Structure and connections of the hidden nodes in the network. fb_{li} is the feedback signal to $(i,l)^{th}$ hidden node from the l^{th} output node (i.e., $fb_{li} = z_{li}v_l$). (Note that the index of hidden nodes is varied from 1 to k for sake of simplicity in representation. Actually only those hidden nodes are present for which the corresponding objects have the feature i.) e_{il} represents the differential support which is either zero or $c_i - b_i$, where $b_i = \max_l(fb_{li})$. The symbol denotes the modulating signal appearing through the corresponding link. Each hidden node is basically a conjugation of two nodes (one in the lower level and other in the higher level). In the lower level within the box, the nodes compete and the output of each node modulates the activation of the corresponding node in the higher level within the box. e_{il} is zero if the lower part of the $(i,l)^{th}$ is a loser one, and it will be $c_i - b_i$ if the lower part is a winner. The weights for competitive connections are not shown here.

Whenever a feature train is presented at the input layer of X-tron, the input layer activates the hidden nodes, and activation propagates from the hidden nodes to the output nodes through the bottom-up links. Due to the presence of negative self-feedback in the output layer, the activations reduce slightly, and again propagate back to the hidden nodes through the top-down links. Since there is no path from the hidden nodes to the input nodes, activations cannot flow back to the input nodes. As soon as the hidden nodes receive activations through the top-down links, they start competing between themselves through the lateral inhibitory connections, and the node receiving maximum activation through top-down link becomes winner-take-all (WTA)[5] [23]. The WTA hidden node remains enabled and the other hidden nodes get disabled. In this way corresponding to each input node there would be at most one enabled hidden node[6]. The difference between the activation coming from the input node and the feedback through the top-down link to the enabled node is propagated through the bottom-up link to the corresponding output node. Again the same process repeats i.e., the output activations propagate to the hidden nodes, and one of the hidden nodes corresponding to each input node becomes winner-take-all, and the differential activation is propagated to the output nodes. Each output nodes retains its activation for a period over which the competition in the hidden layer takes place. Each hidden node has functionally two parts, one of them retains the activation coming from the output layer, the other part competes and makes the node enabled or disabled. The negative self-feedback in the output layer ensures that the activations of output nodes getting no support from the hidden layer will reduce to zero.

In this process, if the activation from input layer is less than the feedback from output layer to any enabled hidden node, the activation of the corresponding output node will be negated. The effect is just opposite when the input activation is higher than the feedback from output layer. Since the input activations represent the input feature confidences, the effect is just the same as that of e_{il} described in Equn. (4.6). The weights of the top-down links emulate the factor a_{li}, and the weights of the bottom-up links emulate the constant κ_1, while the weights of the self-feedback in the output layer emulate the constant κ_2. Thus in this process of updating, the output nodes corresponding to the genuine objects interpreting the feature set remain highly activated, while activations of other output nodes would

[5]This is due to MAXNET principle

[6]If the feedback through top-down link is zero to all hidden nodes, then there will be no winner-take-all node corresponding to that input node

gradually disappear. The dynamic behavior of the network is explained in the next subsection.

4.3.2 Dynamic Behavior of Output Nodes

Let us consider that l^{th} object is represented by the l^{th} output node, and the i^{th} feature is represented by the i^{th} input node. The hidden nodes are numbered according to the associated input and output nodes, viz. the hidden node connected to i^{th} input node and l^{th} output node will be numbered as an ordered pair (i, l). The notations used here are given below.

v_l : output (or state) of l^{th} output node
u_l : instantaneous input to l^{th} output node
w_{il} : weight of the bottom-up link from $(i, l)^{th}$ hidden node to l^{th} output node
z_{li} : weight of the top-down link from l^{th} output node to $(i, l)^{th}$ hidden node
w_s : weight of negative self-feedback of each output node
c_i : input to i^{th} input node

The states of the output nodes are updated according to the differential equation given as

$$\frac{du_l}{dt} = \sum_{i=1}^{n} w_{il} e_{il} - w_s v_l \tag{4.7}$$

where e_{il} measures the difference of the input activation from i^{th} input node and the feedback from l^{th} output node, provided the $(i, l)^{th}$ hidden node is enabled (winner-take-all node) [7]. Mathematically,

$$e_{il} = \begin{cases} c_i - z_{li} v_l & \text{if } z_{li} v_l \geq z_{mi} v_m \ \ \forall m \neq l \\ 0 & \text{otherwise} \end{cases}$$

The output v_l is related to the instantaneous input u_l (to the lth output mode) by a nondecreasing transfer function $g(.)$, i. e. $v_l = g(u_l)$. In the present work $g(.)$ is

[7] The same notation has been used for differential support as in equation (4.6). But e_{il} used here represents a differential signal.

chosen as an S-function given as

$$\begin{aligned} g(u) &= 0 && \text{for } u \leq 0 \\ &= \tfrac{1}{2}(2u)^{1/\epsilon} && \text{for } 0 < u \leq 0.5 \\ &= 1 - \tfrac{1}{2}(2-2u)^{1/\epsilon} && \text{for } 0.5 < u \leq 1 \\ &= 1 && \text{for } u > 1 \end{aligned}$$

In the transfer function $g(.)$ the value of ϵ controls the gain of the transfer function. The value of ϵ varies in the range $[0, 1]$. If ϵ is 0, then output takes values only 0 and 1. On the other hand, if ϵ is 1, then the gain is simply a linear one.

The dynamic system described by Equn. (4.6) can be shown to converge to stable states by having an energy function $\mathcal{E}(t)$ defined as

$$\mathcal{E}(t) = 1/2 \sum_{i=1}^{n} \lambda_i' \left(\max_{l=1,\ldots,k} z_{li} v_l - c_i \right)^2 + 1/2 w_s \sum_{l=1}^{m} v_l^2 \tag{4.8}$$

where n is the number of input nodes, and m is the number of output nodes. λ_i' is a multiplication factor such that

$$\lambda_i' = w_{is}/z_{si} \qquad \text{if } z_{si} v_s = \max_{l=1,\ldots,m} z_{li} v_l$$

The change of energy with the parameter t (time) can be written as

$$\frac{d\mathcal{E}}{dt} = \sum_{l=1}^{m} \frac{\partial \mathcal{E}}{\partial v_l} \frac{dv_l}{dt} \tag{4.9}$$

But,

$$\frac{\partial \mathcal{E}}{\partial v_l} = -\sum_{i=1}^{n} z_{il} \lambda_{il} \delta_{il} - w_s v_l$$

i. e.

$$\frac{\partial \mathcal{E}}{\partial v_l} = -\frac{du_l}{dt}$$

So the Equn. (4.9) can be written as

$$\frac{d\mathcal{E}}{dt} = -\sum_{l=1}^{m} g^{-1'}(v_l) \left(\frac{dv_l}{dt} \right)^2 \tag{4.10}$$

From Equn. (4.10) it is evident that $\frac{d\mathcal{E}}{dt} \leq 0$ for all $t \geq 0$. As $t \to \infty$, $\frac{d\mathcal{E}}{dt} \to 0$, and thereby the dynamic system converges to the local energy minima.

Comparing the energy function $\mathcal{E}(t)$ in Equn. (4.8) and the optimization function E in Equn. (4.4), it is clear that the weights of the top-down links play the same role as the quantity a_{li} in Equn. (4.4). The weights of the top-down links should measure the relative importance of the features with respect to the objects. The factor λ_i' to some extent imitates the constant κ_1. Actually in the optimization function E the effects of the errors due to mismatch of the input confidence and the feedback support were taken to be the same for all features. But in practice the situation should be different; if a feature is important with respect to some object, the effect of mismatch corresponding to that feature-object association should be more. Actually, these preferential effects of mismatch is taken care of in the energy function (by the use of different weights in the bottom-up links), and thereby in the the dynamic behavior of the output nodes. The way to select and automatically learn the weights of the links will be discussed in the next section.

4.4 Learning the Object Categories

X-torn would be able to correctly recognize the objects from the input feature train, and thereby able to interpret the input features with irredundant set of objects only when the weights of the top-down and bottom-up links are set properly. The setting of the weights of these links depend on how well the features are associated with the objects. But since there is no a priori knowledge about these associations, knowledge has to be procured adaptively. The acquisition of the knowledge (or learning) about these associations can be performed with the help of a set of training samples (*supervised mode*), or without any help of training samples (*unsupervised mode*). A supervised mode of learning has been considered in this chapter. In this learning mode, a single object (training sample) is presented to the network at a time, and the input confidence vector of the corresponding feature set is presented at the input. At the output layer, the node corresponding to the training sample is enabled, and all other nodes are disabled. The weights of the links are then modified according to the learning rules, as discussed in the following subsections. The process of learning is continued until the change in the weights become insignificant.

The structure of the network is also incrementally adjusted during the process of learning. Initially, there exists a pool of input nodes, hidden nodes, and output

nodes in the network. Whenever a training sample is presented, the corresponding input nodes are enable and one output node is activated. If there is no association between the enabled output node and an enabled input node (i. e. they are not connected to a common hidden node), then a hidden node is created and it is connected to that input and output node (bottom-up and top-down links are created).

If corresponding to an input-output association, there already exists a hidden node, then the the weights of the links are modified. In the modification process the rate of change of weights should not be the same for all links. This is due to the fact that as the learning process goes on the rate of learning decreases which ensures the convergence (discussed in Section 4.4.3). Therefore if a link is created in the earlier part of learning process, its rate will be decreased compared to a newly created link. Each node (input and output) has an attribute which determines for how much time the node has been enabled. Henceforth this attribute will be termed as *agility factor*. Actually, agility factor decreases with the amount of time for which a node has been enabled. Although each hidden node is enabled for a less amount of time compared to its associated input node, it has no separate agility factor, and this is the same as that of the associated input node. The rate of learning of any link at any instant is governed by the agility factors of the nodes at its two ends. For learning the weights, a suitable measure of the weights of the links are considered and weights are iteratively changed to asymptotically reach these measures.

4.4.1 Measure of Weights

It has already been discussed that the effect of mismatch between the input confidence value and the feedback from output through top-down link should not be the same for all feature-object associations. Actually this effect is more if a feature is more important with respect to an object and vice-versa. This preferential effects are incorporated in the weights of the bottom-up links from the hidden nodes to the output nodes. The importance of any feature with respect to an object model primarily depends on two factors.

First, to what extent the feature is consistent with respect to a particular object, i. e. given the object is present, what is the likelihood that the feature will appear in

the input. Actually, this factor ensures that if a feature is erratic and very prone to noise then it is better not to associate much importance with that particular feature-object association. This is supported by the fact that the change in output confidence of any output object due to mismatch in any input feature is proportional to a_{li} (Equn. (4.6)), where a_{li} is 1 or 0 depending on whether i^{th} feature belongs to the l^{th} object or not. If the value of a_{li} is extended to the continuous domain in $[0, 1]$, then a_{li} basically gives a measure of the likelihood of the appearance of the i^{th} feature with respect to the l^{th} object.

Second, the importance of a feature with respect to a particular object means that how much the object is likely given that the feature is present. Therefore, if a feature is consistent and unique with respect to a particular object, then the feature should have high importance with respect to that object. On the other hand, despite consistency, if the feature is shared evenly by a large number of objects, the importance of the feature will be low with respect to anyone of the objects.

Considering both the factors, just described, the weights of the bottom-up links are taken to be proportional to the likelihood of the appearances of the features with respect to the objects, and as well as the likelihood of the appearances of the objects with respect to the features. A measure of importance of any feature f_i with respect to an object o_l can be given as

$$m(o_l, f_i) = p(o_l|f_i)p(f_i|o_l)$$

where $p(o_l|f_i)$ is the conditional probability of occurrence of o_l given that f_i is present (similar explanation for $p(f_i|o_l)$)[8]. Instead of using the probability values directly, a modified measure also can be used which is given as

$$m(o_l, f_i) = \sigma_1(p(o_l|f_i))\sigma_2(p(f_i|o_l))$$

where $\sigma_1(.)$ and $\sigma_2(.)$ are two semilinear nondecreasing transfer functions saturated at 0 and 1. The use of $\sigma_1(.)$ and $\sigma_2(.)$ enhance the importance of the measure. Since the weights of the bottom-up links are proportional to the importance of the feature-object association, the learning process can be designed so that after convergence of the learning, weights of the bottom-up links capture the measure $m(o_l, f_i)$, i. e. $w_{il} \propto m(o_l, f_i)$

[8]Here for simplicity of representation o_l has been used to denote the occurrence of o_l.

The measure can also be written as the product of two different measures like

$$m(o_l, f_i) = m_1(o_l, f_i) m_2(o_l, f_i)$$

$$\begin{aligned} \text{where} \quad & m_1(o_l, f_i) = \sigma_1(p(o_l|f_i)) \\ \text{and} \quad & m_2(o_l, f_i) = \sigma_2(p(f_i|o_l)) \end{aligned}$$

The measure $m_2(.)$ represents the consistency of f_i with respect to object o_l. As described in the dynamic behavior of the network in Equn. (4.7), the effect of output activation is modulated by the weight of the top-down link. The effect is same as multiplying c_l^o by a_{li} in Equn. (4.6). Actually if a feature is erratic with respect to an object, it should not get proper support from the object, and the weight of the top-down link measures the consistency of the feature with respect to the object. Therefore the learning process designed so that after convergence of learning the weight of the top-down link should capture the measure m_2, i.e., $z_{li} \propto m_2(o_l, f_i)$. Again $m_2(o_l, f_i)$ is 1 if feature f_i is fully consistent with object o_l, i.e., $a_{li} = 1$ in case of perfect features, as described in Section 4.2.1. The weight of the top-down link, therefore, can be taken exactly equal to $m_2(.)$, i. e. $z_{li} = m_2(o_l, f_i)$. The measure $m_1(.)$ actually gives a value proportional to w_{il}/z_{li}. In Equn. (4.7) describing the dynamic behavior of the nodes, λ_{il} is therefore proportional to $m_1(o_l, f_i)$, which supports the discussion presented in Section 4.2.2. We will denote the measure $m_2(o_l, f_i)$ as λ_{il}.

In selection of the actual weights (i.e., after learning what should be the weights), the weights of the bottom-up links are selected in such a way that the total input going to an output node, with perfect input feature train presented, be less than or equal to unity. This is done because the gain function of each node is chosen as an S-function, which saturates if the input is greater than or equal to unity. In order that an output node does not saturate for noisy input confidence vector and reaches the edge of saturation for perfect input confidence vector, the sum of all the bottom-up weights going to that output node must be less than or equal to unity. Mathematically,

$$\sum_{i=1}^{n} w_{il} \leq 1. \tag{4.11}$$

Moreover, if the feature set of an object (say, A) is a proper subset of the feature set of another object (say, B), and the probabilities of appearance of both the objects are same, then whenever the larger feature set (i.e., F_B) is presented during recognition mode, both the objects (i.e., A and B) would be equally activated. Therefore, in the hidden layer there would be a confusion during the competition

process. To prevent such situation, the measure of the weights of the bottom-up links are modified according to Weber's law, as described in adaptive resonance theory [130]. The modified measure is given as

$$w_{il} = \frac{m(o_l, f_i)}{\gamma + \sum_{i=1}^{n} m(o_l, f_i)} \quad . \tag{4.12}$$

where γ is a constant. It will be shown that γ actually determines the rate of learning also. The actual learning algorithm considers how to embed the measures in the weights of the links.

4.4.2 Learning of the Weights

The weights of the links in X-tron are changed in such a way that they reach the measures asymptotically after sufficient number of learning trials. The conditional probability $p(o_l|f_i)$ can be considered to be equal to N_l'/N_i (almost everywhere) for large values of N_l' and N_i. N_i represents the total number of occurrences of feature f_i, and N_L' is the number of occurrences of object o_l given that feature f_i has occurred. Initially the ratio does not give actual probability values, but if the measure is made equal to the ratio, then as the number of presentations becomes very high, the measure converges to the actual probability values. Therefore at any instant of time t,

$$\lambda_{il}(t) = \frac{N_l'}{N_i}, \tag{4.13}$$

where N_l' and N_i are interpreted as the respective number of occurrences till the instant t. Similarly, the weight of bottom-up links take the values given as

$$z_{li}(t) = \frac{N_i'}{N_l}, \tag{4.14}$$

where N_l is the total number of occurrences of object o_l, and N_i' is the number of occurrences of feature f_i given the object o_l is present till the instant t. If the feature f_i is present at time instant $t+1$ then the value of $\lambda_{il}(t)$ changes as

$$\lambda_{il}(t+1) = \begin{cases} \frac{N_l'+1}{N_i+1} & \text{if } o_l \text{ is present at instant } t+1 \\ \frac{N_l'}{N_i+1} & \text{if } o_l \text{ is absent at instant } t \end{cases}$$

If the interval of occurrences are considered instead of the number of occurrences, then $\lambda_{il}(t)$ can be written as

$$\lambda(t) = \frac{T_l'(t)}{T_i(t)}, \tag{4.15}$$

where T_i is the total time for which the feature f_i is presented, and T_l' is total interval for which o_l was present given that f_i was present.

In the supervised learning process, every time a teaching vector is presented at the output layer, which enables a single output node and disables the other nodes. Let the desired output vector be denoted by

$$\mathbf{y}(t) = [y_1(t), y_2(t), \ldots, y_k(t)],$$

i.e., for each output node there is a desired output which changes over time t. If the feature f_i is present for the interval Δt then the value of $\lambda_{il}(t)$ changes to

$$\lambda_{il}(t + \Delta t) = \frac{T_l'(t) + y_l(t).\Delta t}{T_i(t) + \Delta t}. \tag{4.16}$$

Equation (4.16) is valid if Δt is small enough, and desired output $y_l(t)$ considered to remain unchanged over the interval Δt. The change of $\lambda(t)$ can be written as

$$\Delta\lambda_{il}(t) = \lambda_{il}(t + \Delta t) - \lambda(t)$$

i.e.,

$$\Delta\lambda_{il}(t) = \begin{cases} 0 & \text{if} \quad c_i = 0 \\ \frac{T_l' + y_l(t).\Delta t}{T_i + \Delta t} - \frac{T_l'}{T_i} & \text{if} \quad c_i \neq 0 \end{cases}$$

i.e.,

$$\Delta\lambda_{il}(t) = \frac{\Delta t \left(T_i y_l(t) - T_l'\right) c_i}{T_i \left(T_i + \Delta t\right)}. \tag{4.17}$$

In the above expression it is considered that c_i can take values of 0 and 1 only. But the same expression can be taken to be valid for fractional values of c_i in the range of $[0, 1]$. The convergence of the above expression for continuous values of c_i will be given in the next section. The rate of change of $\lambda_{il}(t)$ can be written as

$$\frac{\Delta\lambda_{il}(t)}{\Delta t} = \frac{y_l - \lambda_{il}(t)}{T_i + \Delta t} c_i$$

Letting $\Delta t \to 0$ the differential change of $\lambda_{il}(t)$ with time t becomes

$$\frac{d\lambda_{il}}{dt} = \alpha_i c_i \left(y_l - \lambda_{il}\right) \tag{4.18}$$

where

$$\alpha_i = 1/T_i$$

α_i decreases with the time of activation of the i^{th} input node. α_i is actually the agility factor associated with i^{th} input node. In the network initially agility factor for all nodes are set to be unity. As the nodes are enabled in the learning mode, the agility factor decrease. The change of agility factor for i^{th} input node in time Δt can be given as

$$\Delta\alpha_i(t) = \begin{cases} 0 & \text{if } c_i = 0 \\ 1/T_i - 1/(T_i + \Delta t) & \text{if } c_i \neq 0 \end{cases}$$

Letting $\Delta t \rightarrow 0$ the rate of change of $\alpha_i(t)$ becomes

$$\frac{d\alpha_i}{dt} = -\alpha_i^2 c_i \tag{4.19}$$

Considering the measure $m_1(o_l, f_i)$, the rate of change of weight of the top-down links can be given as

$$\frac{dz_{li}}{dt} = \alpha_l^o y_l (c_i - z_{li}) \tag{4.20}$$

where α_l^o is the agility factor of the l^{th} output node. The rate of change of α_l^o can be given as

$$\frac{d\alpha_l^o}{dt} = -\alpha_l^{o2} y_l \tag{4.21}$$

The total weights of the bottom-up links should be such that for a set of inputs, the output equals the desired output vector. Therefore the expression for w_{il} after convergence can be written as

$$w_{il} = \frac{\lambda_{il} z_{li} g^{-1}(y_l)}{\gamma + \sum_{j=1}^{n} \lambda_{jl} z_{lj} c_j} \tag{4.22}$$

The constant γ is used to implement Weber's law. The gain $g(.)$ is considered to be the same for all nodes. With this expression for w_{il}, the total input (u_l) to the l^{th} output node becomes

$$u_l = \frac{g^{-1}(y_l) \sum_{i=1}^{n} \lambda_{il} z_{li}}{\gamma + \sum_{j=1}^{n} \lambda_{jl} z_{lj} c_j}$$

By mathematical restructuring, the expression can be written as

$$u_l = g^{-1}(y_l) - \frac{\gamma w_{il}}{\lambda_{il} z_{li}}$$

i.e.,

$$y_l = g\left(u_l + \frac{\gamma w_{il}}{\lambda_{il} z_{li}}\right)$$

The constant γ is considered to be very small. Using Taylor's expansion, and restoring up to second term the expression can be written as

$$y_l - o_l = \frac{\gamma w_{il} g^{'}(u_l)}{\lambda_{il} z_{li}} ,$$

i.e.,

$$w_{il} = \frac{\varepsilon_l \lambda_{il} z_{li}}{\gamma g^{'}(u_l)}$$

where

$$\varepsilon_l = y_l - o_l$$

The factor ε_l measures the error between the desired output and the actual output at the l^{th} output node. The required rate of change of w_{il} should be

$$\frac{dw_{il}}{dt} = \frac{\varepsilon_l}{\gamma g^{'}(u_l)} \left(z_{li} \frac{d\lambda_{il}}{dt} + \lambda_{il} \frac{dz_{li}}{dt} \right) . \tag{4.23}$$

Let $\delta = \frac{\varepsilon}{\gamma g^{'}(u_l)}$. Using Equn. (4.18) and (4.20) the above expression can be written as

$$\frac{dw_{il}}{dt} = \overbrace{\left(\alpha_i \delta_l z_{li} + \alpha_l^o \left(\frac{w_{il}}{z_{li}} \right) \right) c_i y_l}^{I} - \overbrace{\left(\alpha_i c_i + \alpha_l^o y_l \right) w_{il}}^{II} \tag{4.24}$$

In Equn. (4.24) the factor $1/\gamma$ determines the rate of learning. If γ increases the rate decreases, and vice-versa. The part I of Equn. (4.24) indicates that the weight of a link changes with the product of the activations of the nodes at the two ends, modulated by the error at the output node. This, in fact, supports the Hebbian rule of learning [226]. Part II shows a decay in the weight of the link, which actually supports the associative decay of the weights of the links [130].

4.4.3 Convergence of the Weights

In this section convergence of the weights of the top-down and bottom-up links have been considered. If the presentation of the patterns are considered to be random then $\mathbf{y(t)}$ is a stochastic process, and the appearances of the input activations are also random, i.e., $\mathbf{c(t)}$ is random. Initially, the agility factors for all the nodes are set to unity. The agility factor of a node (input or output) decreases whenever it is activated. Whenever an input-output association is formed through a hidden node, the weight of the corresponding top-down link is set to unity. In the learning process, if the input-output association already exists in the network, the weight

of the corresponding top-down link will change according to Equn. (4.20). The weight of the bottom-up links are initially set to zero, and they increase according to Equn. (4.24). The initial values of the link weights and the agility factors can be summarized as follows :

$$z_{li}(0) = 1, \quad w_{il}(0) = 0, \quad \alpha_i(0) = 1, \quad \alpha_i^o(0) = 1$$

where $t = 0$ indicates the association of i^{th} input node and l^{th} output node is just formed. The convergence of the bottom-up weights need not be proved separately, rather if the convergence of λ_{il} and z_{li} can be shown then it can be said that w_{il} converges for a given γ. The convergence of the weights, with the given initial conditions can be proved both for perfect binary inputs and noisy inputs.

For perfect binary inputs α_i decreases and λ_{il} changes only when $c_i = 1$. Let $c_i = 1$ at the time intervals $(\tau_0, \tau_1), (\tau_2, \tau_3), (\tau_4, \tau_5), \ldots$ and so on. Let us denote the total time for which $c_i = 1$ as τ_i, i. e.

$$\tau_i = \sum_{r=0} (\tau_{2r+1} - \tau_{2r})$$

For nonzero probability of the i^{th} feature, $\tau_i \to \infty$ as $t \to \infty$. Since there is no change in the values of either α or λ when $c_i \neq 1$ (i.e., $c_i = 0$), λ_{il} will converge as $\tau_i \to \infty$ if λ_{il} converge for $t \to \infty$, and vice versa. Therefore the convergence of λ_{il} can be proved considering $c_i = 1$ for all $t \geq 0$. The Equn. (4.19) can be rewritten as

$$\frac{d\alpha_i}{dt} = -\alpha_i^2$$

Therefore $\alpha_i = 1/(t+1)$ with the initial value 1. From Equns. (4.18) and (4.19), λ_{il} can be rewritten as

$$\frac{d\lambda_{il}}{d\alpha_i} = \frac{\lambda_{il} - y_l}{\alpha_i}. \tag{4.25}$$

From the above equation, it is evident that λ_{il} will solely depend on the change of y_l. Let $y_l = 1$ in the intervals $(t_0, t_1), (t_2, t_3), (t_4, t_5), \ldots$ and so on. Separately integrating for $y_l = 1$ and $y_l = 0$, we have

$$\left.\begin{array}{rrcl} & \log\left(\frac{1-\lambda_{il}(t_{2r+1})}{1-\lambda_{il}(t_{2r})}\right) & = & \log\left(\frac{\alpha_i(t_{2r+1})}{\alpha_i(t_{2r})}\right) \\ \text{and} & \log\left(\frac{\lambda_{il}(t_{2r+2})}{\lambda_{il}(t_{2r+1})}\right) & = & \log\left(\frac{\alpha_i(t_{2r+2})}{\alpha_i(t_{2r+1})}\right) \end{array}\right\} \quad \forall r \geq 0. \tag{4.26}$$

The sequence of values that λ_{il} can take are

$$\left.\begin{array}{rrcl} & \lambda_{il}(t_{2r+1}) & = & 1 - \frac{\alpha_i(t_{2r+1})}{\alpha_i(t_{2r})}\left(1 - \lambda_{il}(t_{2r})\right) \\ \text{and} & \lambda_{il}(t_{2r+2}) & = & \frac{\alpha_i(t_{2r+2})}{\alpha_i(t_{2r+1})}\lambda_{il}(t_{2r+1}) \end{array}\right\} \quad \forall r \geq 0. \tag{4.27}$$

By simple algebraic calculation, the value of $\lambda_{il}(t_{2k+1})$ can be written as

$$\lambda_{il}(t_{2k+1}) = \sum_{r=0}^{2k+1} (-1)^{2k-r+1} \frac{\alpha_i(t_{2k+1})}{\alpha_i(t_r)}. \tag{4.28}$$

Since $\alpha_i(t) = 1/(t+1)$, equation (4.28) can be written as

$$\lambda_{il}(t_{2k+1}) = \frac{\sum_{r=0}^{k} (t_{2k+1} - t_{2r})}{1 + t_{2k+1}}. \tag{4.29}$$

In equation (4.29), the numerator of represents the time for which the l^{th} output node is active during supervised learning, i. e. $y_l = 1$ up to the time instant t_{2k+1}. Therefore

$$\lim_{k \to \infty} \lambda_{il}\,(t_{2k+1}) = \Pr\,(y_l = 1 | c_i = 1)$$

almost everywhere by strong law of large numbers [219]. This ensures that $\lambda(t)$ converges to the conditional measure proposed in the previous section. By similar reasoning it can be proved that

$$\lim_{k \to \infty} z_{li}\,(t_{2k+1}) = \Pr\,(c_i = 1 | y_l = 1)$$

Therefore, the product $\lambda_{il} z_{li}$ converges, and w_{il} converges.

If the input feature train is contaminated by noise then the reasoning about the presence or absence of feature no more holds good. Then the convergence of the weights can be proved by considering the inputs and outputs to be continuous signals, i.e., a signal is considered to be present at each input node and at each output node. If a perfect feature is present at an input node, the activation at that node reaches its full strength (i. e. unity), and if the feature is noisy, the activation is less than unity. Similarly, if the feature is absent without any additive noise, the activation at the corresponding input node is zero, and for nonzero additive noise a nonzero activation is present at that node. Considering the change of agility factor the Equn. (4.19) can be written as

$$\frac{d\alpha_i}{\alpha_i^2} = -c_i dt$$

Integrating over the time interval [0,t], and considering initial value of α_i, the above expression takes the form

$$\frac{1}{\alpha_i(t)} = 1 + \int_0^t c_i(\tau) d\tau \tag{4.30}$$

where τ is a dummy variable. Putting the value of $\alpha_i(t)$ in Equn. (4.18) the expression for change in $\lambda_{il}(t)$ can be written as

$$\frac{d\lambda_{il}}{dt} = \frac{c_i(t)(y_l - \lambda(t))}{1 + \int_0^t c_i(\tau)d\tau}$$

i.e.,

$$\frac{d\lambda_{il}}{dt} + \frac{\lambda_{il}(t)c_i(t)}{1 + \int_0^t c_i(\tau)d\tau} = \frac{c_i(t)y_l(t)}{1 + \int_0^t c_i(\tau)d\tau}. \tag{4.31}$$

Multiplying both sides by $\left(1 + \int_0^t c_i(\tau)d\tau\right)$, and integrating in the interval [0,t], the value of $\lambda_{il}(t)$can be written as (with initial value equal to 0)

$$\lambda_{il}(t) = \frac{\int_0^t c_i(\tau)y_l(\tau)d\tau}{1 + \int_0^t c_i(\tau)d\tau} \tag{4.32}$$

In Equn. (4.32) if c_i and y_l take values only $\{0, 1\}$, then the value of $\lambda_{il}(t)$ converges to the conditional probability measure as $t \to \infty$. With similar considerations it can be shown that $z_{li}(t)$ also converges as $t \to \infty$, and thereby w_{il} converges.

This section proves the convergence of the weights of the links. Initially in the formation of the network, each hidden node is connected to a single input-output node pair. The next section provides an estimate about the total number of nodes that can be present in the network.

4.5 Estimation of the Number of Nodes

In Section 4.2 it has been mentioned that the network is incrementally formed during the learning process. The total number of nodes in the network is equal to $n + h + m$, where h is the number of hidden nodes, m is the number of output nodes (i.e., maximum number of categories that can be identified), and n is the number of input nodes (maximum number of input features). Apparently, it seems that the number of hidden nodes will be approximately equal to the product of the number of input nodes and the number of output nodes, i.e. of $O(m \times n)$. But in practice, h depends on the number of categories that actually share each input feature, because the number of hidden nodes associated with an input node is equal to the maximum number of objects sharing the feature corresponding to that input node. In one extreme case h is of $O(m \times n)$ and in the other extreme, it is equal to the number of input nodes, when each feature belongs to exactly one

object. Here, it is assumed that there is no redundant feature, i.e., each feature belongs to at least one output category.

The expected number of hidden nodes can be calculated by considering a particular model of input-output association. It is very unlikely that a feature will belong to all output categories. Again, the probability that a feature will not belong to any object is zero. Therefore, for any particular feature i, if it is shared by approximately μ_i number of categories, then the probability that the i^{th} feature is shared by μ_i categories will be maximum, and it will decrease as the number of objects increase or decrease. Following the nature of the input-output association, it can be validly assumed that the association follows a truncated Poisson distribution. The probability that the i^{th} feature is shared by j number of categories is given by

$$\Pr(i^{th}\text{feature shared by j objects}) = \frac{\frac{\exp(-\mu_i)\mu_i^{j-1}}{(j-1)!}}{\sum_{r=1}^{m-1}\frac{\exp(-\mu_i)\mu_i^{r-1}}{(r-1)!}} \tag{4.33}$$

The expression is normalized because a feature can be shared by at most m categories, i.e., the summation is in the range $j \in [1, m]$. The total number of hidden nodes can be written as $h = \sum_{i=1}^{n} h_i$ where h_i is the number of hidden nodes associated with the i^{th} input node. The expected value of h_i is given as

$$\begin{aligned} h_i &= \sum_{j=1}^{m} j.\Pr(i^{th}\text{feature is shared by j objects}) \\ &= 1 + \frac{\sum_{j=0}^{m-1}\frac{j.\exp(-\mu_i)\mu_i^j}{j!}}{\sum_{r=0}^{m-1}\frac{\exp(-\mu_i)\mu_i^r}{r!}} \\ &= 1 + \mu_i\left[\frac{\sum_{j=0}^{m-2}\frac{\mu_i^j}{j!}}{\sum_{r=0}^{m-1}\frac{\mu_i^r}{r!}}\right] \\ &= 1 + \mu_i H_i \end{aligned} \tag{4.34}$$

where

$$H_i = \left[\frac{\sum_{j=0}^{m-2}\frac{\mu_i^j}{j!}}{\sum_{r=0}^{m-1}\frac{\mu_i^r}{r!}}\right] \tag{4.35}$$

From equation 4.35, it is evident that $H_i < 1$. Again by simple algebraic calculation, H_i can be written as

$$H_i = 1 - \frac{\frac{\mu_i^{m-1}}{(m-1)!}}{\sum_{r=0}^{m-1}\frac{\mu_i^r}{r!}}$$

But

$$\sum_{r=0}^{m-1}\frac{\mu_i^r}{r!} = \exp(\mu_i) - \sum_{r=m}^{\infty}\frac{\mu_i^r}{r!}$$

$$< \exp(\mu_i) - \frac{\mu_i^{m-1}}{(m-1)!}$$

Therefore,

$$H_i > 1 - \frac{\frac{\mu_i^{m-1}}{(m-1)!}}{\exp(\mu_i) - \frac{\mu_i^{m-1}}{(m-1)!}} \tag{4.36}$$

From equation 4.36, it can be inferred that the total number of hidden nodes in the network is bounded, and the upper and lower bounds in the number of hidden nodes are governed by the bounds of H_i for all i. The total number of hidden nodes h can be written as

$$n + \sum_{i=1}^{n} \mu_i > h > n + \sum_{i=1}^{n} \mu_i - \sum_{i=1}^{n} \frac{\frac{\mu_i^m}{(m-1)!}}{\exp(\mu_i) - \frac{\mu_i^{m-1}}{(m-1)!}} \tag{4.37}$$

In equation (4.37), if we assume $\mu_1 = \mu_2 = \cdots = \mu_n = \mu(say)$ then the total number of hidden nodes cannot be greater than $n + n\mu$. For a fixed value of μ, the total number of hidden nodes will be $O(n)$, and thereby the total number of nodes will be of $O(m + n)$.

4.6 Results and Analysis

X-tron has been simulated on SUN 3/60 workstations and tested for different synthetic patterns. The experiment has been performed in two stages. In the first stage, the network is allowed to learn the input patterns under supervised mode. After suitable number of learning trials the network is tested for recognition task in the second stage. This section describes the results of these experiments and analyzes the results.

During learning under supervised mode, the pattern vectors to be learned are considered to have preassigned probabilities of appearances in the network. Input vectors are applied to the input in a frequency determined by their preassigned probability values. With each input vector the information about the corresponding object is stored, and from this stored information the output vector is generated which acts as the teaching vector. In actual network formation, a hidden node is connected in the network whenever a new input-output pair appears. In the simulated network, for sake of simplicity, the total number of hidden nodes is considered

to be equal to the product of the number of input and output nodes. Although there may be lot of redundant nodes in the hidden layer, but the performance of the network in terms of learning the input vectors and recognizing the objects will not be affected at all. The weights of the bottom-up links corresponding to the uninstantiated hidden nodes happen to be zero and do not anyway affect the recognition task. Only the time performance of the network may degrade to some extent during recognition on a sequential machine.

4.6.1 Binary Strings

In the first set of experiment, X-tron is simulated with fourteen input nodes and seven output nodes. Seven different 14-bit vectors are presented to the network during the learning trials. The input vectors are shown in Table 4.1. The preas-

Table 4.1: Feature vectors of different objects

objects	*features*													
object 1	0	0	0	1	0	0	0	1	0	0	0	1	1	1
object 2	1	1	0	0	1	1	1	0	1	0	0	0	1	0
object 3	0	1	1	0	0	1	0	0	1	1	1	0	0	1
object 4	1	0	0	0	0	1	0	0	1	1	1	0	0	0
object 5	0	1	1	0	1	0	1	0	0	1	0	0	1	1
object 6	1	1	0	1	1	0	0	0	0	0	0	0	0	0
object 7	1	1	0	1	1	0	0	1	1	0	0	0	0	0

signed probability values of the seven objects are shown in Table 4.2. The objects

Table 4.2: Probabilities of appearances of the objects

objects	1	2	3	4	5	6	7
probabilities	0.1	0.14	0.16	0.2	0.1	0.15	0.15

are randomly presented to the network according to their probability values. This is done by generating a random number in the range [0,1] by the system-built pseudo-random number generator. If the number is such that $P_{k-1} \leq N < P_k$, where $P_k = \sum_{i=1}^{k} p_i$, (p_i is the preassigned probability value of the i^{th} pattern) then the generated pattern is considered to be the k^{th} pattern. The network is

trained with 15000 learning trials and the corresponding results are shown in Table 4.3 and Table 4.4. The value of γ, determining the rate of learning, is chosen

Table 4.3: Weights of the bottom-up links (w_{ij})

Inputs(i)	*Outputs(j)*						
	1	*2*	*3*	*4*	*5*	*6*	*7*
1	0.0	0.097	0.0	0.150	0.0	0.196	0.117
2	0.0	0.090	0.085	0.0	0.073	0.180	0.108
3	0.0	0.0	0.221	0.0	0.189	0.0	0.0
4	0.113	0.0	0.0	0.0	0.0	0.309	0.184
5	0.0	0.115	0.0	0.0	0.093	0.233	0.139
6	0.0	0.124	0.117	0.191	0.0	0.0	0.0
7	0.0	0.253	0.0	0.0	0.205	0.0	0.0
8	0.176	0.0	0.0	0.0	0.0	0.0	0.287
9	0.0	0.096	0.091	0.147	0.0	0.0	0.115
10	0.0	0.0	0.127	0.206	0.109	0.0	0.0
11	0.0	0.0	0.161	0.261	0.0	0.0	0.0
12	0.409	0.0	0.0	0.0	0.0	0.0	0.0
13	0.133	0.180	0.0	0.0	0.146	0.0	0.0
14	0.126	0.0	0.161	0.0	0.138	0.0	0.0

as 0.1, and the time step is chosen as 0.005. From Table 4.3 it is evident that the weights of the bottom-up links corresponding to the features shared by large number of objects are comparatively small. And the weights of the bottom-up links corresponding to the features shared by less number of objects are comparatively high. During the learning trials the input vectors may be contaminated by noise. In the present case supervised learning is performed with perfect input vectors. Therefore, if the weight of a top-down link becomes unity, it remains unchanged.

In the second stage of experiment, the noisy patterns are presented at the input and X-tron is executed in the recognition mode. Different synthetic patterns (may be caused by one or more objects) with different noise levels are applied at the input. The results of the recognition task are shown in Table 4.5. The network is simulated for 500 iterations for each instance of input pattern with the self-feedback equal to 0.1 and timestep = 0.05. Noise (with uniform distribution) is added to the input by the system-built pseudo-random number generator. The power of the network is tested for overlapped patterns of more than one object. Table 4.5 shows

Table 4.4: Weights of the top-down links(z_{ji})

Inputs(j)	*Outputs(i)*						
	1	*2*	*3*	*4*	*5*	*6*	*7*
1	0.0	1.0	0.0	1.0	0.0	1.0	1.0
2	0.0	1.0	1.0	0.0	1.0	1.0	1.0
3	0.0	0.0	1.0	0.0	1.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0	1.0	1.0
5	0.0	1.0	0.0	0.0	1.0	1.0	1.0
6	0.0	1.0	1.0	1.0	0.0	0.0	0.0
7	0.0	1.0	0.0	0.0	1.0	0.0	0.0
8	1.0	0.0	0.0	0.0	0.0	0.0	1.0
9	0.0	1.0	1.0	1.0	0.0	0.0	1.0
10	0.0	0.0	1.0	1.0	1.0	0.0	0.0
11	0.0	0.0	1.0	1.0	0.0	0.0	0.0
12	1.0	0.0	0.0	0.0	0.0	0.0	0.0
13	1.0	1.0	0.0	0.0	1.0	0.0	0.0
14	1.0	0.0	1.0	0.0	1.0	0.0	0.0

Table 4.5: outputs with superposed feature vectors

pattern	*recognition score*								
	noise level = 0.1			noise level = 0.2			noise level = 0.3		
	true	false	extra	true	false	extra	true	false	extra
objects 3 and 4	100	0	0	94	0	0	84	0	0
objects 5 and 6	100	0	0	100	0	0	100	0	0
objects 1 and 2	47	0	53	43	0	57	38	0	62
object 7	100	0	0	92	0	8	85	0	15

that whenever the overlapped pattern corresponding objects 5 and 6 is presented to the network, the network recognizes the objects correctly without any error even with 30% noise level. When the overlapped pattern corresponding to objects 3 and 4 is presented to the the network, it recognizes correctly both the objects even with 10% noise in the input. But it fails to recognize the object 4 in 6% of cases when input is contaminated with 20% of noise. This is due to the fact that when the patterns of object 3 and 4 are superposed, the object 4 retains only one distinctive feature which does not belong to object 3 (feature 1). On the other hand, object 3 has three distinctive features not belonging to object 4 (features 2, 3 and 14). Therefore the initial activation of object 3 is higher than that of object 4, and in the iterative process of recognition, object 3 gets support from the common features (features 6, 9, 10 and 11) and its own distinctive features as well. On the other hand object 4 gets support from only one distinctive feature. If the input patterns are heavily contaminated by noise the initial activation of the objects will be very low. And in the process of iterative activation object 4 sometimes does not get much of the support from its single distinctive feature and that is flagged in the output. The results also show that the network fails to recognize object 4 in 16% of cases when the input is contaminated by 30% of noise. When the overlapped pattern of objects 1 and 2 is presented, the network recognizes the object 7 along with the objects 1 and 2 almost 47% of time with 10% noise. The chance of recognizing the extra object increases with the percentage of noise. This is due to the fact that the difference of the pattern corresponding to object 7 with the overlapped pattern of objects 1 and 2 is only in the last three bits. The noise injected in the patterns acts both in additive and subtractive fashion. In the initial settling process of the network, the total percentage of error in the pattern of object 7 and that in the pattern of object 1 and object 2 become comparable. Therefore the output of the node corresponding to object 7 becomes high initially and it is supported by the features in the overlapped pattern, and the recognition of extra object occurs. The fourth row of Table 4.5 shows that when the pattern of object 7 is presented to the network with 20% of noise, the object 6 is recognized along with object 7 in 8% cases. This is due to the fact that object 7 has two extra features than object 6. Therefore whenever object 7 is presented, all the features of object 6 is present in the scene. But the formulation in Equn. (4.22) prevents the object 6 to become fully active when the pattern of object 7 is presented. Again the effectiveness of this discrimination will depend on the constant γ. With the chosen value of γ, the effectiveness of this restriction fails in 8% cases with 20% noise, and recognition of extra object occurs. As the noise increases, the chance of recognizing object

6 along with object 7 increases. However recognition is not affected up to 10% noise level in the input. The results show that whenever an overlapped pattern is presented, the genuine objects are always recognized, and sometimes extra objects are recognized depending on the noise level and the nature of the patterns. But the results presented here deal with the patterns having a very high overlap. In almost all practical situations patterns are expected to have much less overlap, and such problems will seldom arise.

4.6.2 Visual Patterns

In the second set of experiment, X-tron is presented with four different synthetic patterns representing four different objects (as shown in Fig.4.2). The objects

Table 4.6: Response at the output layer with the input pattern presented shown in Fig.4.3

objects	*response at respective output nodes*		
	n.l. = 0.1	n.l. = 0.15	n.l. = 0.2
object 1	0.89	0.88	0.87
object 2	0.91	0.90	0.88
object 3	0.02	0.04	0.05
object 4	0.10	0.12	0.13

Table 4.7: Responses at the output nodes with the input pattern shown in Fig.4.4

objects	*response at respective output nodes*		
	n.l. = 0.1	n.l. = 0.15	n.l. = 0.2
object 1	0.89	0.88	0.87
object 2	0.09	0.10	0.10
object 3	0.95	0.94	0.93
object 4	0.02	0.04	0.05

are learned by the network in supervised mode treating each pixel as a separate feature. The objects could have been learned by extracting out the different local features in the objects, but the objective of this chapter is to show the capability of the network to learn and recognize multiple objects simultaneously under noisy

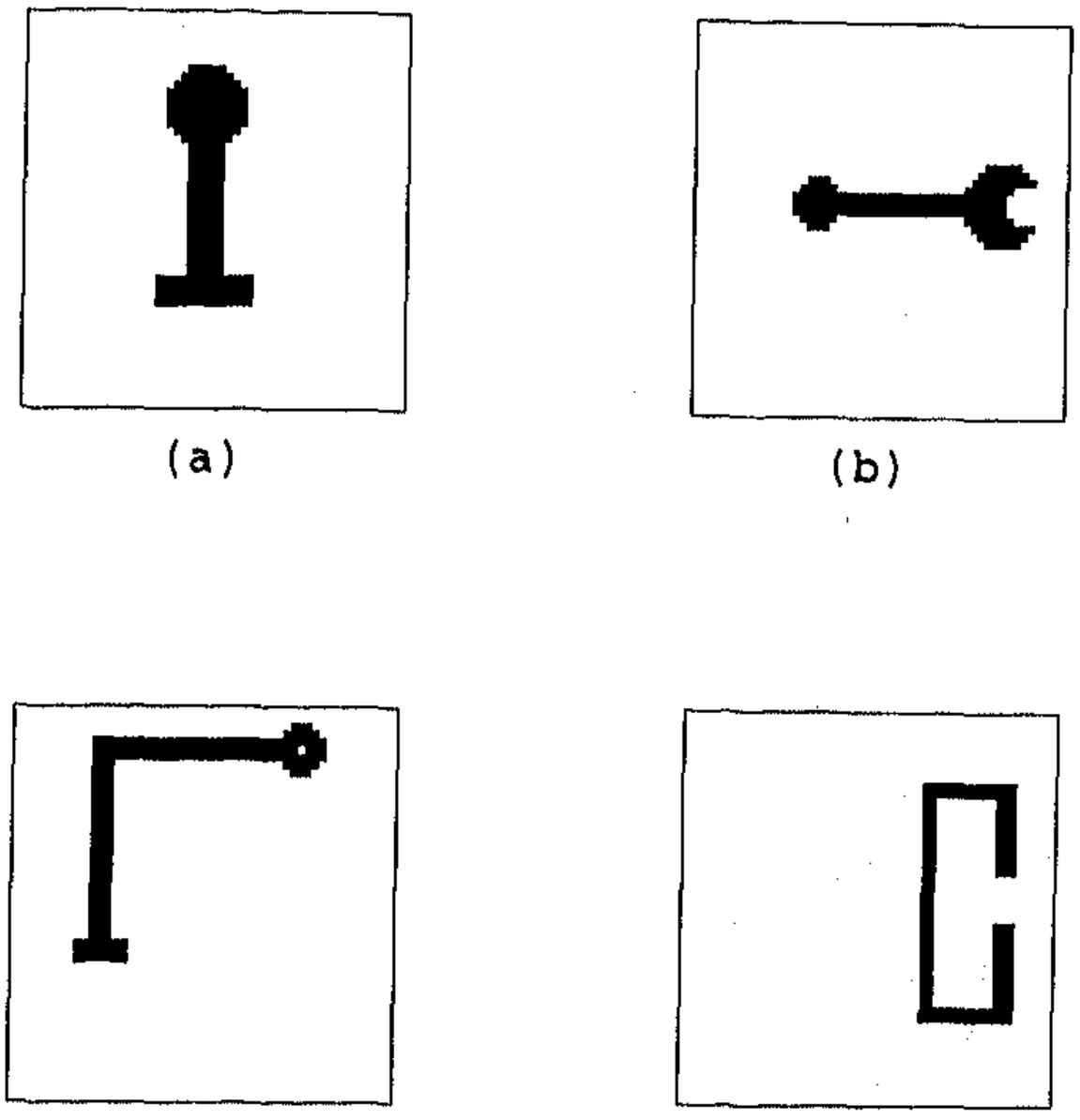


Figure 4.2: Four different objects presented to the network.

Table 4.8: Responses at the output nodes with the input pattern shown in Fig.4.5

objects	*response at respective output nodes*		
	n.l. = 0.1	n.l. = 0.15	n.l. = 0.2
object 1	0.07	0.08	0.10
object 2	0.91	0.90	0.88
object 3	0.02	0.04	0.05
object 4	0.96	0.94	0.93

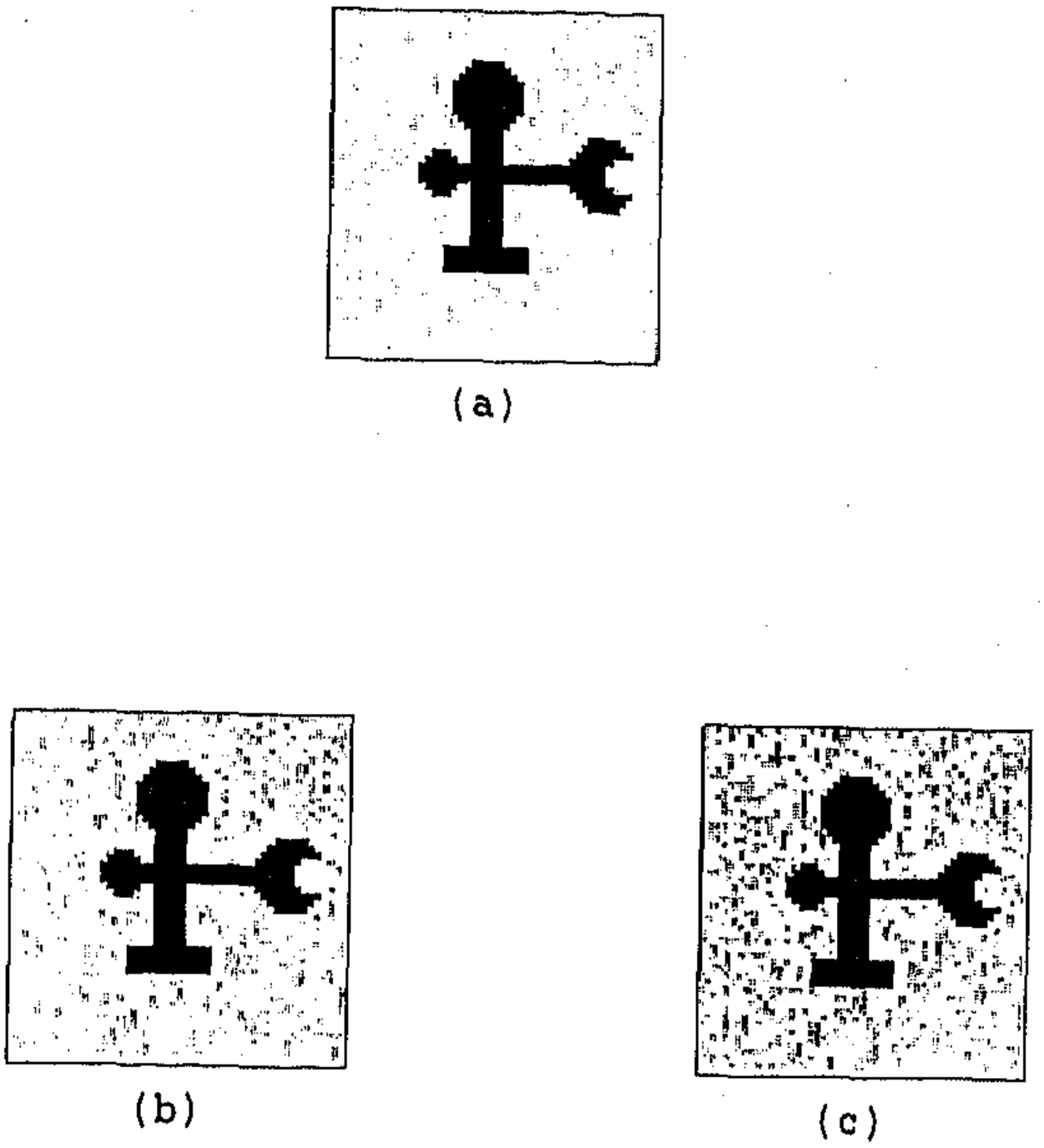
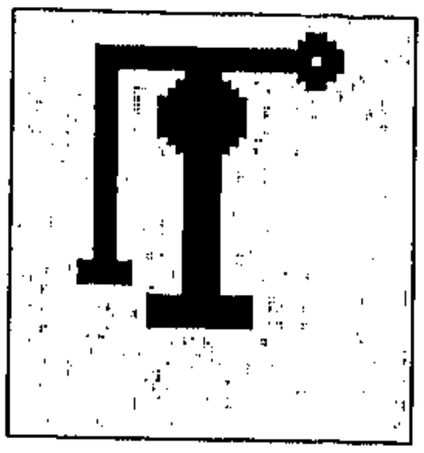


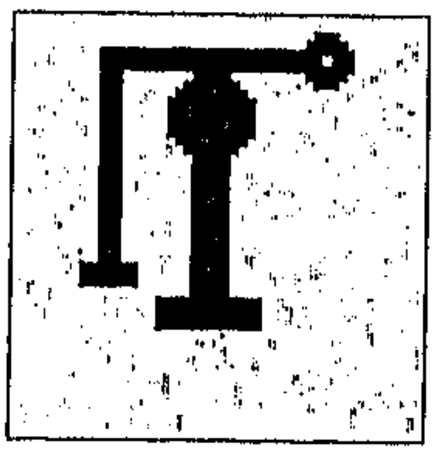
Figure 4.3: Superposed pattern of objects 1 and 2 (a) with 10% noise, (b) with 15% noise, (c) with 20% noise.

Table 4.9: Responses at the output nodes with the input pattern shown in Fig.4.6

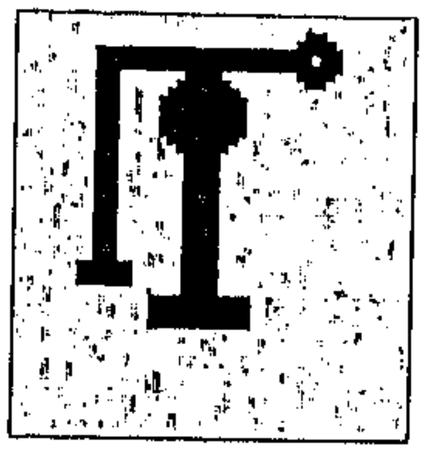
objects	*response at respective output nodes*		
	n.l. = 0.1	n.l. = 0.15	n.l. = 0.2
object 1	0.02	0.04	0.05
object 2	0.12	0.13	0.14
object 3	0.95	0.94	0.93
object 4	0.96	0.94	0.93



(a)

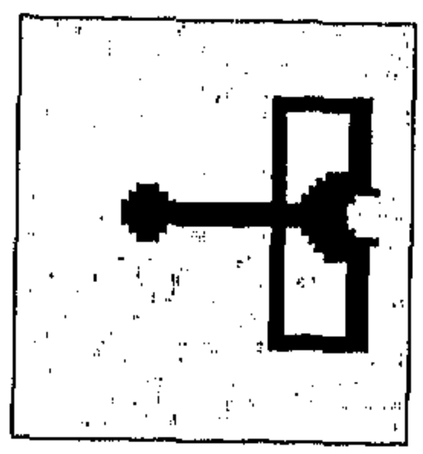


(b)

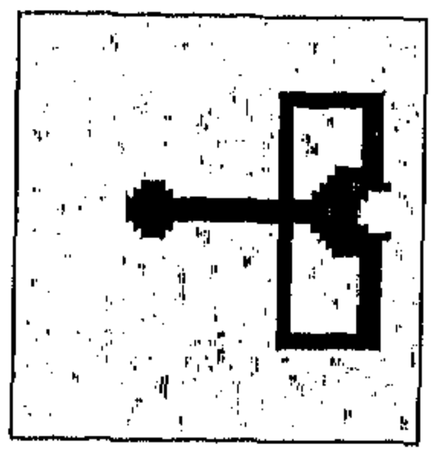


(c)

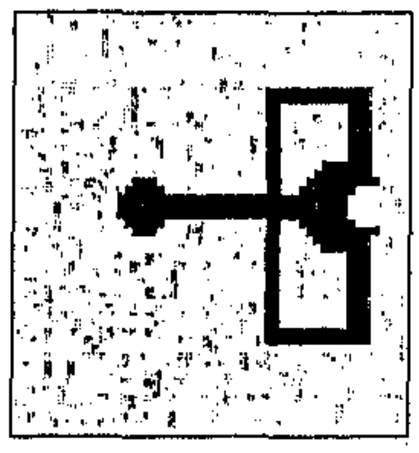
Figure 4.4: Superposed pattern of objects 1 and 3 (a) with 10% noise, (b) with 15% noise, (c) with 20% noise.



(a)



(b)



(c)

Figure 4.5: Superposed pattern of objects 2 and 4 (a) with 10% noise, (b) with 15% noise, (c) with 20% noise.

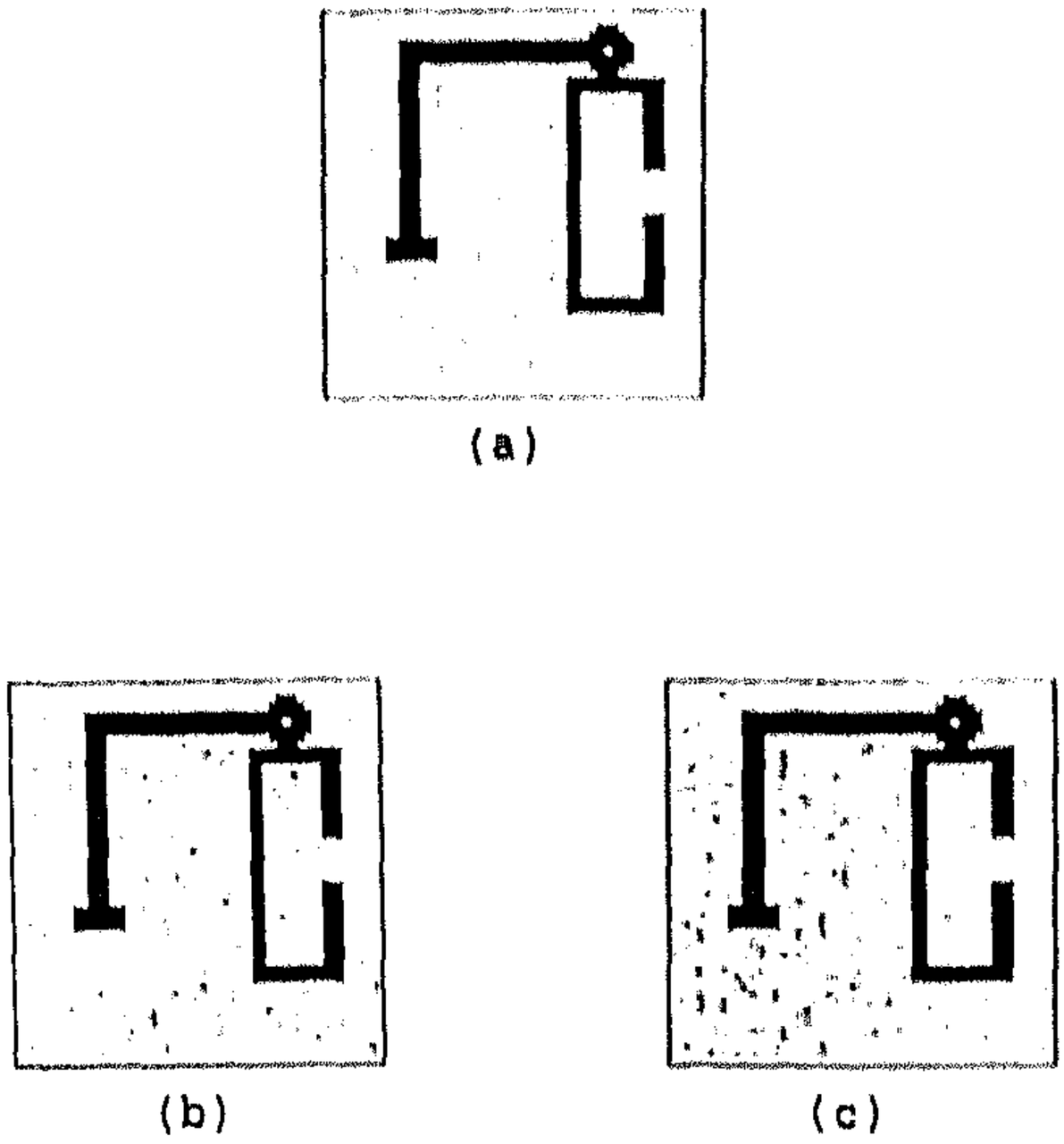


Figure 4.6: Superposed pattern of objects 3 and 4 (a) with 10% noise, (b) with 15% noise, (c) with 20% noise.

environment (note that the results with extracted local features from real-life scene will be presented in Chapters 6 and 7). The normalized graylevel of each pixel is taken as the confidence of the corresponding feature. The weights of the bottom-up and top-down links are set up according to the pixel values in the images. After learning trials are over the images are superimposed and noise is injected to produce the test patterns for recognition. The different test patterns are shown in Fig.4.3,4.4,4.5,4.6. Each image consists of 2500 pixels. During learning, weights are updated for 100 trials with the time step equal to 0.005 and $\gamma = 5$. Note that, the value of γ is much higher than that chosen in the previous experiment. This is due to the fact that the number of features is much higher in this case. Again, since the normalized weights of the bottom-up links are much less than that in the previous case (because number of features are much higher) the learning process converge quickly and only 100 learning trials are necessary. The network is executed in recognition mode with the self-feedback equal to 0.1, and time step = 0.005. The results after 25 iterations with different test patterns and different noise levels are presented in Tables 4.6, 4.7, 4.8, 4.9. From the tables, it is evident that with the increase in noise level, the output of the genuine objects decrease, and that of the "false" objects increase.

4.7 Conclusions and Discussion

A new model for learning and recognition of mixed categories is described in this chapter. The design of the network architecture and the state dynamics of the nodes are guided by an explicit mathematical formulation of the mixed category perception problem which is mostly guided by the memory based reasoning [225]. The learning rules are defined based on some predefined probabilistic measures. Although the derivation of the learning rules does not assume any explicit biological functionality, they exhibit some similarity with the Hebbian rule [226] and the associative decay phenomenon [130].

X-tron is comparable to other existing network models and superior to some of them. If the agility factors of the nodes are kept fixed at unity then the learning rules basically reduce to the learning rules used in the adaptive resonance theory [130], operated under supervised mode. In the present model, since the probabilities of appearances of the objects/categories are also considered, this is supposed

to provide a stronger paradigm for learning the object classes. ♠

Note that, X-tron, developed here, operates only under supervised mode. In the next chapter, we demonstrate its self-organizing ability (unsupervised) alongwith its comparisons with other related models.

Chapter 5

X-tron : Unsupervised Mixed Category Perception

5.1 Introduction

In X-tron, presented in Chapter 4, for mixed category perception, the learning rules operate only under supervised mode. It may be sometimes necessary to automatically capture the key features of an object without the help of an external teacher. It may be mentioned in this connection that all biological systems exhibit self-organizing property, where the entities are grouped into categories automatically without the help of any external teacher. In this chapter, we have formulated a theory of categorization in the presence of mixed/overlapped patterns, and based on this, the self-organizing architecture of X-tron is designed [209]–[211].

Some of the investigations on self-organization have been mentioned in Section 1.3.1. The concept of self-organization in connectionist models is analogous to the idea of clustering in pattern recognition [72]. In general, the idea is to select a seed for some category so that its distance (which can be Euclidian or Mahalanobis or city-block or any other suitable measure) from any incoming pattern can be measured. Some objective function based on this distance is then used to decide whether the pattern belongs to that category or not. This very concept was used in most of the self-organizing networks mentioned before. In our problem more than one category (in mixed form) can be present at a time in a given input.

In that case, it would be rather misleading to compare any single category with the incoming pattern when a particular combination of more than one category is comparable (similar) to the input. Therefore, instead of using the conventional concept of clustering in pattern recognition, a measure of similarity between the feature set presented and the predicted feature set by the currently hypothesized objects is used to control the formation of new object categories in the output layer.

The underlying concept of the present self-organizing system is as follows. Whenever a feature train appears at the input the network is allowed to settle with its existing structure and set of connection weights. After settling, the output categories feed back the activation to the input features. For each feature some kind of ambiguity is then measured depending on the external input and the amount of feedback. On that basis, a certainty factor is derived in order to monitor if the incoming pattern is a new category or a combination of the existing categories.

In the process of self-organization, whenever a new category is detected (by considering the certainty factor), an output node is allocated for the category. To incorporate the knowledge about input-output association of the new category, some hidden nodes may also added to the network. As a result, the network incrementally changes the size of the layers (hidden and output) to accommodate the new unknown categories.

The rest of this chapter is organized as follows. Some properties of X-tron, operated under supervised mode, are described in Section 5.2. In Section 5.3, the principles of categorization, in the presence of mixed categories, are formulated on the basis of the properties described in Section 5.2. The categorization architecture alongwith the noise characteristics and issues of stability is also described in Section 5.3. Some experimental results demonstrating the effectiveness of X-tron for categorization in the presence of overlapped binary strings and visual patterns are provided in Section 5.4. Finally, some concluding remarks alongwith the comparison of X-tron with other related investigations are provided in Section 5.5.

5.2 Properties of X-tron

In this section some new properties of X-tron, operated under supervised mode, have been established, based on which the proposed theory of categorization (or self-organization) has been developed. Before proceeding to the logical formulation of these properties a couple of definitions are provided.

Definition 5.1 : *The network is said to have learnt an object class p if and only if the change of weights of the bottom-up and top-down links for the corresponding input feature set is less than some threshold ϵ, whatever small it may be.*

Definition 5.2 : *The network is said to have recognized an object class p if and only if the output of the corresponding node in the output layer is greater than some threshold T.*

The way of selection of T will be discussed in Sections 5.3 and 5.4.

Before proceeding to the design of actual categorization (or self-organization) methodology, we must ensure that the network is able to recognize an object if it is claimed that the network has learned that object. Because only in that case the network would be able to proceed into further categorization correctly. Since the present work is aimed at categorization of more than one objects simultaneously (or mixed patterns), we must, at first, ensure that the network is able to correctly recognize the individual categories constituting the mixed patterns. Let us now present some comments on the capabilities of X-trons.

Some Remarks :

The asymptotic values of the bottom-up links takes care of Weber's law. Therefore, if two categories are such that the feature set of one category is a proper subset of the other, then also the network would be able to identify these two categories separately.

As mentioned before, the network is able to recognize mixed patterns. It also tries to reduce the redundancy in the decision. For example, assume that the network has learned three patterns say, *ab*, *cd* and *abcd*. In that case, if *abcd* is presented as an input pattern, only the node corresponding to *abcd* will be activated (though the other patterns *ab* and *cd* are proper subset of *abcd*, nodes corresponding to

them will not be activated). The reason is as follows.

Whenever *abcd* is presented to the network, the initial activations of all three nodes will be high (rest of the output nodes will have low activations considering that no other category has overlap with *abcd*). But as the Weber's law is considered in the asymptotic measures of the weights of the links, the initial activation value of the node for *abcd* will be higher than the rest two (i.e., $O_{abcd} > O_{ab}$ and $O_{abcd} > O_{cd}$). The output nodes feed these activation values to the hidden layer. The hidden nodes which are connected to the same input node compete between themselves, i.e., competition occurs for associating a feature with certain category and not between the categories themselves. Therefore, the hidden nodes connected to the node for *abcd* will receive higher feedback than the hidden node receiving feedback from *ab* or *cd*. The competition in the hidden layer occurs only in the feedback activations. As a result, the hidden nodes connected to the output node for *abcd* will win. In the settling process since an input node sends activation only through the winner-take-all hidden node, output node corresponding to *abcd* will receive support from input. The other two nodes (*ab* and *cd*) will not receive any support because they have only loser hidden nodes (refer equation (4.7)). Since the output nodes are also associated with a negative feedback, the activation values of *ab* and *cd* will gradually diminish while that of *abcd* will reach a stable condition.

Note that the network is not designed to learn the embedded patterns which is performed in SONNET [150]. Rather, it takes the advantage of dissociating the competition process from the activation values in the output layer. The competition takes place for associating a feature with some object which is performed in the hidden layer. This enables the network to recognize patterns even under high amount of overlapping. For example, let us consider that the network has learned two categories *abc* and *bcd*. If the network is now presented with a pattern *abcd* then also it will be able to recognize them separately. Of course, the common subset of features *bc* will be associated to one of the categories (say, *bcd*) (due to competition in the hidden layer), but there would be no inhibition from any of the hidden nodes. Due to the differential support from the rest of the features (i.e., *a*) the activation level of the category *abc* will stabilize to a high activation value. This is also supported by the results provided in Chapter 4.

Let us now consider another example where the network has learned only three categories say, *ab*, *abc* and *cd*. If a new pattern *abcd* is presented to the network then it will recognize it as a combination of *abc* and *cd*. This is due to the fact

that initial activation of *abc* would be higher than that of *ab* and consequently *ab* will have only loser hidden nodes in the settling process. (Here the present system differs from the system developed by Marshall [144], [145] where *ab* and *cd* would be recognized. The expression for the activation level in the output layer is presented in Section 5.3.2. ◇

The following theorems reveal some properties related to the recognition ability of the network in the presence of mixed patterns. We have tried to prove the ability of recognition by inductive reasoning.

Several notations are consistently followed in these theorems. An object class is denoted by $\mathcal{C}$. The feature sets are denoted by F while the individual features have been denoted by f. The initial outputs (before stabilization of the network) are denoted by O. Note that, if a network learns some object class then the initial output of the node corresponding to that class will be maximum among all the output nodes whenever a pattern from that class is presented to the network. This is due to the fact that the dot product of the pattern vector and the corresponding weight vector will be maximum (this was the basis for designing the learning rules).

Theorem 5.1 *If the network is presented with one of its learnt templates at the input layer then it will recognize only the category corresponding to that template.*

Proof : Let us assume that the network has learnt object class $\mathcal{C}_p$. The value of ϵ has been chosen sufficiently small. Let a sample from the object class $\mathcal{C}_p$ be now presented to the network. The initial output of the node corresponding to $\mathcal{C}_p$ is greater than the output of any other node. Mathematically, $O_p > O_i, \forall i \neq p$. O_p denotes the initial output of the corresponding output node before stabilization.

Let an arbitrary input feature f_k belong to object p. In the set of hidden nodes which are connected to the node corresponding to feature f_k, the node connected to p^{th} output will have maximum feedback (provided the feature is not noisy, i.e., the weight of the top-down link is very close to unity). Therefore, the particular hidden node wins over the other hidden nodes in the competition. In the recognition process, the winner-take-all hidden node will send differential activation (e) to object p only. All other objects sharing the feature f_k will not get any active support from it. Similar phenomena will occur for all other features belonging to object p. By this process the outputs of objects other than p will decrease more and

more due to the self-negation and absence of any active support from the feature level. Moreover, the features which do not belong to object p will negate the activations of spurious objects. Only the object p has both self-negation and some positive activation from feature level. The other objects have only self-negation. As a result, the response of object p will stabilize to some positive value while the responses of other objects will come down to zero. Therefore, if the threshold T is selected suitably then the object p can be claimed to have been recognized. □

Theorem 5.2 *If the network is presented with a overlapped pattern of two of its learnt categories such that the feature set of one is not a proper subset of that of the other then it will recognize at least two categories.*

Proof : Suppose two objects a and b have been presented to the network. Let F_a and F_b be the respective feature sets of the objects classes a and b. There can be three different cases :

- There does not exist any other object whose feature set is a subset of $F_a \bigcup F_b$.
- There exists no other object c for which $O_c > O_a$ or O_b.
- There exists at least one object c such that $F_c \subset F_a \bigcup F_b$ and $O_c > O_a$ or O_b.

In the first case it is evident that the outputs corresponding to a and b will be stable to some positive value. But since there exists no other object dependent only on the subset of the features, the outputs of other objects will get some negation from the feature level itself. Therefore, the outputs of the spurious objects will reduce to zero.

In the second case even if there exists some object which is dependent entirely on a subset of the features present at the input, its initial output is less than those of the true objects present. [1] Therefore the spurious objects will share no active feature and consequently, the support from the feature level will be zero. Due to the self-negation, the outputs of the spurious objects will decrease and eventually reduce to zero.

[1]The word "true" means only those objects which are presented to the network. In the present case "true" objects are a and b.

In the third case, the initial output of the spurious object c is greater than those of the true objects. Therefore the spurious object will share some active features. Again, the true objects will have some active features. Therefore, the true objects and the spurious object(s) will coexist in the final output. □

Now we will consider a property regarding the recognition of mixed objects.

Theorem 5.3 *Suppose, a network can recognize a set of n objects simultaneously (i.e. input pattern is the overlapped feature set corresponding to this n objects). If another object (i.e., the $(n+1)^{th}$ object), now presented to the network, is such that*

- *it has some features not belonging to either of the n objects and*
- *there exists no other object having a feature set which is a subset of the union of all the features of $(n+1)$ objects and whose initial output is greater than any of those of the $(n+1)$ objects,*

then the network will be able to recognize only these $(n+1)$ objects simultaneously.

Proof : Let the network be capable of recognizing a set of n objects (say, objects 1 to n) perfectly. Therefore there does not exist any other object p such that F_p is a subset of $F = \bigcup_{i=1}^{n} F_i$ and $O_p > O_1$ or O_2 or O_3 or ... O_n. Because in that case the object p will share some of the active features and p may also coexist. The system, in that case, would not recognize exactly these n objects.

Again there cannot exist any other object p such that F_p is a subset of the union of F_{n+1} and the feature sets of any other n - 1 or less number of objects, and its initial output is greater than that of any of these objects. Because in that case if only those objects were presented to the network (number of objects in that case is less than or equal to n) the network could not be able to eliminate the object p, which violates the basic condition.

There can be another case. The object p is such that $F_p \not\subset F$ but $F_p \subset F \bigcup F_{n+1}$, and $O_p > O_1$ or O_2 or O_3 or ... O_{n+1}. In that case the object p will share some of the active features and in the stable state the object p will coexist. On the other hand, if the initial response of the object p is less than those of the other objects present, then no spurious objects will be detected. □

Theorem 5.4 *The network is capable of recognizing any set of n objects exactly if the objects and the corresponding feature sets are such that there exist no other object p for which*

- $F_p \subset \bigcup F_{k_i}$ *where* k_i *can be anything in the entire object set and*
- $O_p > O_{k_i}$ *for some value of* k_i

Proof : The statement is true for two objects (Theorem 5.2). Since this is true for two objects it is true for three objects (Theorem 5.3) and so on. It is, therefore, true for any set of n objects. □

5.3 Categorization

Depending on the properties of the network described in section 5.2, the theory of categorization (self-organization) and the corresponding architecture have been developed here.

5.3.1 Overall Methodology and Categorization Architecture

The underlying concept of the proposed categorization technique is as follows. Whenever a pattern (either corresponding to a single category or mixed categories) appears, it produces output responses for the categories which are already learnt. These outputs actually yield a measure indicating how well the feature set is being interpreted by the network. If the activation of a particular input node does not match with the support received from the output categories, then there will be an ambiguity corresponding to that feature. If a single pattern or a set of mixed patterns from a new class appears, the features will not be fully interpreted by the learnt categories. If the total ambiguity for all features is greater than some threshold then the pattern presented to the network is considered to belong to a new category. In designing the architecture for such self-organization(categorization), a portion of the network must therefore be able to monitor the performance of categorization and to determine the ambiguity value.

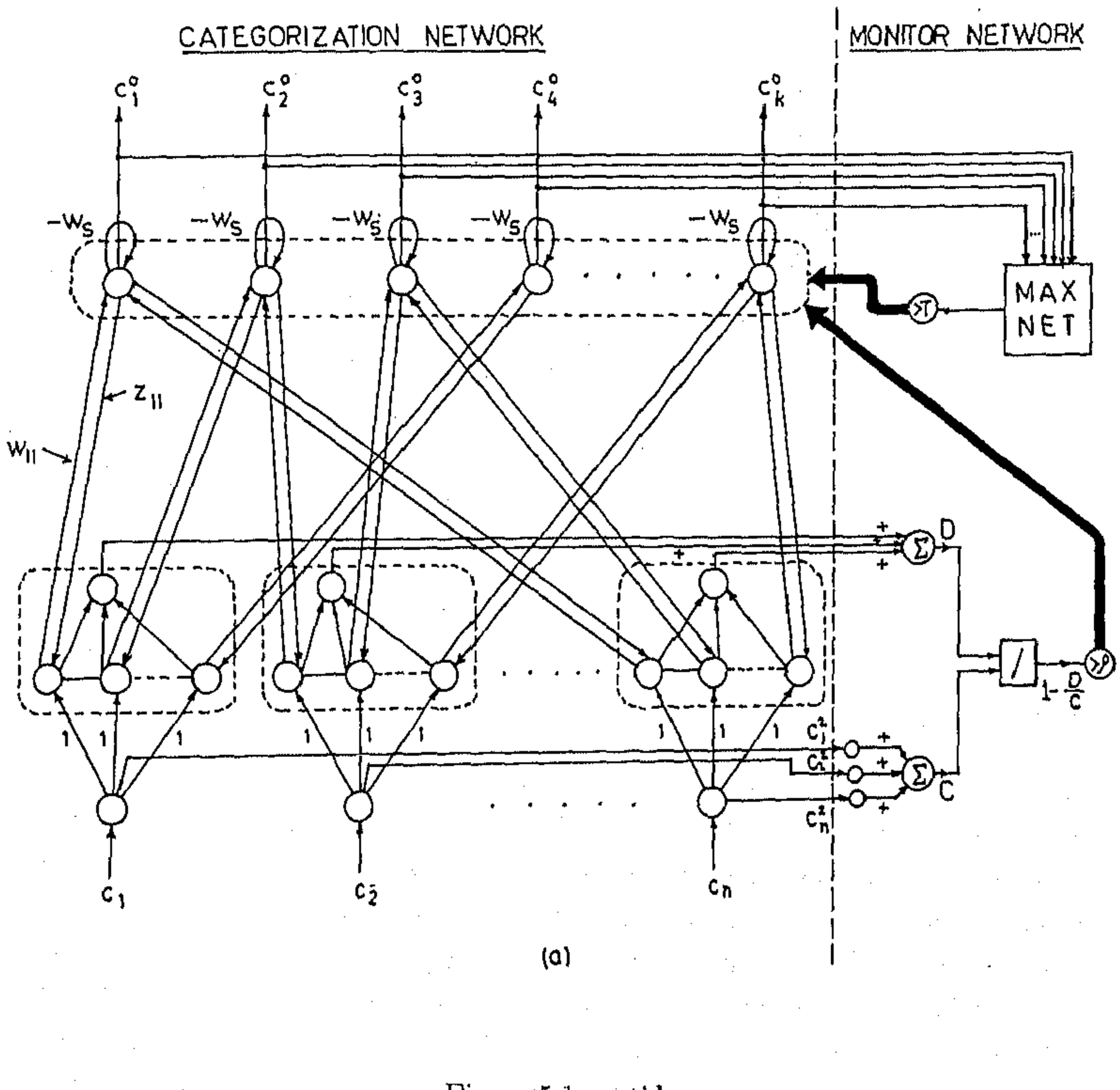


Figure 5.1 cont'd.

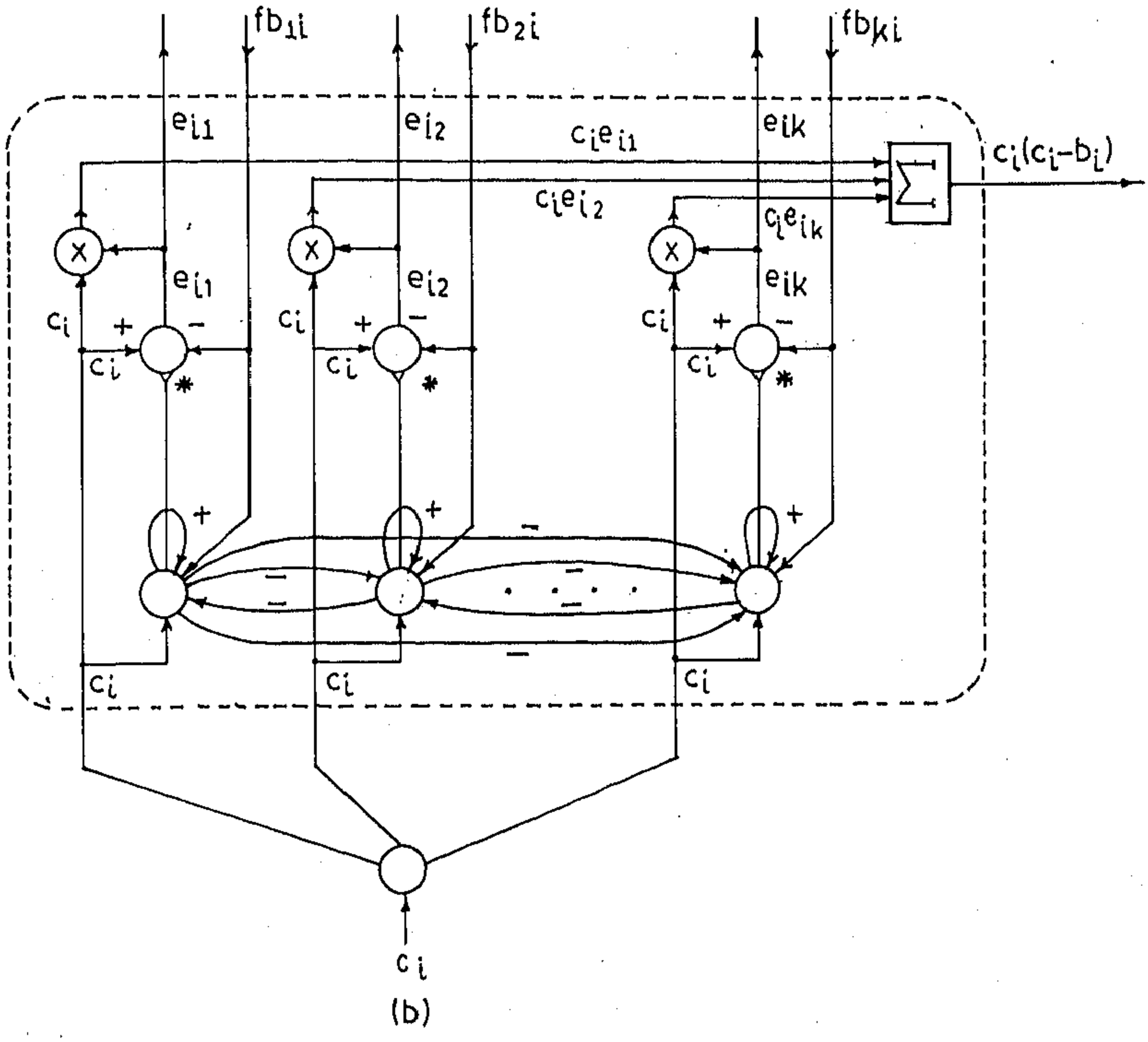


Figure 5.1: (a) Structure of the connectionist model for self-organization. Bold lines represent the control paths from the monitor network to the categorization network. Another node is added in each box of hidden nodes which measures the ambiguity at the corresponding feature (considering the set of hidden nodes). Two adders are used in conjunction with hidden and input layers to compute D and C respectively. The node denoted as "/" computes the certainty factor $1 - \frac{D}{C}$. (b) Structure and connections of the hidden nodes in the network. The symbols have similar meanings as in Fig. 1(b). $c_i e_{il}$ measures the ambiguity for an individual hidden node which is either zero (if e_{il} is zero, i.e., loser hidden node in the competition) or equal to $c_i(c_i - b_i)$ (only for the winner-take-all hidden node). Note that the output of adder will also be $c_i(c_i - b_i)$.

In the architecture for categorization, an extra network is added to monitor the performance of the network for some given pattern. The network architecture is shown in Fig.5.1. With each hidden node there is another node which computes the ambiguity at that particular node. These ambiguities and the input activations are propagated to the monitor where the certainty factor is calculated. The certainty factor is considered to have a linear relation with the total ambiguity. If the certainty factor is greater than some threshold then the pattern is considered to be already present, and the weights of the links are accordingly updated. If the certainty factor is less than the threshold then a new output node is allocated for the input pattern. This threshold is called the *vigilance threshold* (ρ) which is supplied externally. This vigilance threshold is analogous to the vigilance factor used in adaptive resonance theory [130].

In the monitor network there is another part which checks the maximum activation present at the output layer. The maximum activation is checked if it is greater than a threshold T [2]. If the condition holds true then only the network is allowed to learn the pattern present at the input. Otherwise, even if the certainty factor is greater than the vigilance threshold the learning is not allowed. Whenever a new output node is allocated in the output layer the upper part of the monitor network is activated and the network learns the new pattern.

The method of self-organization works as follows :

Step 1 Present a new pattern. The pattern may be a representative of some single category or may be caused by the presence of more than one category.

Step 2 Measure the *Certainty Factor* (CF). Measure the maximum output (x) in the output layer. Note that there does not exist any output node initially. In that case, CF and x are considered as zero. The monitor network measures these two factors.

Step 3 If $CF > \rho$ and $x < T$ then go to step 8;

if $CF > \rho$ and $x > T$ then make all the output nodes whose activations are greater than T fully active, and go to step 6.

Step 4 Allocate a new output node in the output layer. If the total number of

[2]Note that T was used for template type in Chapter 2, however we use T in this chapter to refer a threshold value.

output nodes is greater than the capacity of the network then exit.

Step 5 Allocate hidden nodes for having input-output associations corresponding to the features which are present at the input. Connect each newly created hidden node to the corresponding input node (with a link of unit weight), and to the newly created output node (with bottom-up and top-down links). Initialize the weights of the newly created links i.e., weights of the bottom-up links are set either to zero or to a small value. The weights of the top-down links are set equal to the activation level of the corresponding input node (i.e., $z_{li} = c_i$ where l is newly created output node and i is the input node). Make the newly created output node fully active.

Go to Step 7.

Step 6 If a single node is activated in the output layer, check if there exists any input node for which there is no highly activated hidden node (which indicates that the corresponding feature did not appear in the previous examples). If any such input node exists then create a hidden node to represent the association between the input node and the activated output node. Connect the hidden node with the input node (with a link of unit weight), to other hidden nodes connected to the same input node (with inhibitory links), to the activated output node (with bottom-up and top-down links). Initialize the weight of the bottom-up link to zero or a very small value. Initialize the weight of the top-down link equal to the activation value of the input node.

Step 7 Learn the weights of the links, i.e., iterate the weights till they converge (become less than some threshold ϵ(say)).

Step 8 Present another new pattern. Go to step 2.

The categorization process always takes an input pattern and self-organize accordingly. Therefore, it is able to do its task with as many number of input patterns so long as there is no restriction about the size of the network. It always self-organizes in on-line mode and does not consider any batch-mode operation.

5.3.2 Output Response

The response of an output node is now considered. Let each input feature be represented as a tuple (c_i, w_i, z_i). Here c_i is the confidence value of the feature. w_i and z_i represent the weights of the bottom-up and top-down links from the $(i,l)^{th}$ hidden node to the particular output node (l) and from that output node (l) to the $(i,l)^{th}$ hidden node respectively (here we suppress the subscript l for the output node because response of any arbitrary node l has been considered). Suppose, the output of the node stabilizes to a value x. Let the negative self-feedback of each output node be w_s. Without loss of generality, we can consider all features to be active, i.e. sending differential activation (e) to the output node. If some features are shared by other objects and they do not send any activation to the output nodes then that triplet can be omitted from the set. Under stable condition, the positive and negative signals at the output node will cancel each other. Therefore, from Equn. (4.7), under stable condition,

$$w_s x = \sum_{i=0}^{n} (c_i - z_i x) w_i \tag{5.1}$$

By simple algebraic manipulation it can be shown that

$$x = \frac{\sum_{i=0}^{n} w_i c_i}{w_s + \sum_{i=0}^{n} w_i z_i} \tag{5.2}$$

When the network has learnt a particular category, the value of $\sum_{i=0}^{n} w_i z_i$ is nearly unity (the learning rules are designed in such a way). If w_s is small enough and all c_i values are nearly unity then the output x will reach unity. Depending on the value of self inhibition w_s the value of T can be selected.

Example 1 : Consider an object for which all w_is (at some stage of learning) are equal to w. Let the object have k features each of equal importance to the object, and the weights of all top-down links be unity. Then the output is given as

$$x = \frac{kw}{w_s + kw}$$

If only k_1 features are active and the rest are shared by others so that they cannot send any differential support, the output becomes

$$x_1 = \frac{k_1 w}{w_s + k_1 w}$$

To recognize the object, even for k_1 features being present at the input, we must have

$$\frac{k_1 w}{w_s + k_1 w} > T$$

or,

$$w_s < k_1 w \left(\frac{1}{T} - 1\right)$$

Consider $kw = K$, then the above equation becomes

$$w_s < \frac{k_1}{k} \left(\frac{1}{T} - 1\right) K$$

If the network has to recognize the object even for 10% of the active features (i.e., *rest* 90% *features are shared by other objects*) then (considering K = 1.0) w_s becomes

$$w_s < 0.1 \left(\frac{1}{T} - 1\right)$$

or,

$$T < \frac{1}{1 + 10 w_s}$$

If $w_s = 0.02$ then $T < 0.83$, i.e. a threshold of 0.8 will work well enough to recognize objects under heavy occlusion.

Note that, in the network design the value of w_s should be selected in such a way that the network does not take long time to stabilize and the output threshold T is not too low.

5.3.3 Principles of Ambiguity Measure

It is clear from the discussion made so far that the network will produce some stable output for a set of active features. For a particular output, whether the object will be considered to have been recognized or not will depend on the design of the system. But if an object is present in the scene, and the network has learnt the object then the features must get proper support from the output layer. This is the very basic concept over which the categorization (self-organization) strategy has been developed.

For example, consider the previous case (Example 1) where an object has k features each connected to the corresponding output node by a bottom-up link of weight w and a top-down link of unit weight. Then the output will be $\frac{kw}{w_s + kw}$. Therefore

each feature will get the same support. If w_s is small enough, the input confidence value will be almost equal to that of the top-down support (since output response and, therefore, the top-down support are very near to unity, and the input confidence value is considered to be unity). If some features are shared by some other objects the mismatch between the input confidence and the top-down support will increase. Ideally, in a noiseless scene there should be no mismatch between the input confidence and the top-down support corresponding to a feature. Due to the presence of negative self-feedback, top-down feedback will be slightly reduced even for a perfect feature. Moreover, if more than one object is present in the scene and they share common subsets of features then, as seen before, the outputs will degrade gracefully. In that case also, the mismatch between the input confidence and the top-down feedback at the feature level slightly increases.

There can also be other cases. Suppose, a feature has suffered from noise or some kind of degradation and its confidence value is less than unity. In that case the top-down feedback may be greater than the input confidence. It may be noted here that the importance of the mismatch for a noisy feature is not the same as the importance of mismatch for a perfect feature. The entire ambiguity in the task of recognition depends on how far the features are being interpreted (or explained or emphasized) by the network. If the recognition is perfect then all the features will be well interpreted. If the top-down support is less than the input confidence then the feature is not properly interpreted by the network. If the top-down support is more than the input confidence then the feature is over-emphasized. Perfect interpretation means a perfect matching. If the average interpretation over the entire feature set can be quantified (i.e. a measure of average ambiguity can be provided) then it will certainly yield a quantitative index for the recognition criterion.

Suppose, the network is presented with a set of n features with input confidence values given by $[c_1, c_2, \cdots, c_n]$. Let, after the network has stabilized, the top-down feedback corresponding to these features be $[b_1, b_2, \cdots, b_n]$. Here, we always consider the maximum top-down support to a feature, i.e., the feedback received by the corresponding WTA hidden node. The total ambiguity D corresponding to the entire feature set can be defined as

$$D = \sum_{i=1}^{n} c_i(c_i - b_i) \tag{5.3}$$

Note that, the mismatch between the input confidence c_i and the top-down feed-

back b_i in each feature is modulated by the confidence value of the feature itself, thereby setting the relative importance of the mismatch. If $(c_i - b_i)$ increases, the value of D also increases, and vice-versa. Note that, $D \geq 0$, and possess a maximum value of $\sum_{i=1}^{n} c_i^2 = C$ (say).

The normalized ambiguity measure is given by

$$\mu = \frac{D}{C} \tag{5.4}$$

5.3.4 Principles of Categorization

In the previous section it has been discussed that the recognition criteria can be evaluated on the basis of the normalized ambiguity measure μ. It is clear that if the normalized ambiguity is high then the recognition is very poor and vice-versa. Therefore, the certainty of a decision based on the interpretation of a set of features, may be quantified as

$$CF = 1 - \mu$$

A high certainty factor (CF) means the system is more confident about its decision, while a small value of CF indicates that the system is less confident or confused in making a decision. (Note that there are similar concepts in pattern recognition problem e.g., see [227] where certainty factor indicates the degree of firmness i.e., meaningfulness or validity of a decision regarding belonging of a feature set to a class. On the other hand, the certainty factor in the present case refers to decision regarding interpretation of a feature set for single/mixed category. However, in the case of single category, both these concepts become analogous.)

Consider the case in the previous example where the network is presented with n features with confidence values $[c_1, c_2, \cdots, c_n]$. Let us assume that all these features belong to a single object. In that case all features will be active for that particular object. The corresponding output after stabilization will be

$$x = \frac{\sum_{i=1}^{n} w_i c_i}{w_s + \sum_{i=1}^{n} w_i z_i}. \tag{5.5}$$

Therefore the total ambiguity is given by

$$D = \sum_{i=1}^{n} c_i(c_i - z_i x). \tag{5.6}$$

Substituting the value of x,

$$D = \frac{(w_s + A)\sum_{i=1}^{n} c_i^2}{w_s + \sum_{i=1}^{n} w_i z_i} \tag{5.7}$$

where,

$$A = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(w_j z_j c_i^2 + w_i z_i c_j^2 - (w_i z_j + w_j z_i)\, c_i c_j\right)}{2\sum_{i=1}^{n} c_i^2}.$$

The normalized ambiguity is given by

$$\mu = \frac{w_s + A}{w_s + \sum_{i=1}^{n} w_i z_i} \tag{5.8}$$

Let

$$W = \sum_{i=1}^{n} w_i z_i$$

The certainty factor is, therefore,

$$CF = 1 - \frac{w_s + A}{w_s + W}$$

or,

$$CF = \frac{W - A}{w_s + W}$$

or,

$$CF = \frac{W}{w_s + W}\left(1 - \frac{A}{W}\right). \tag{5.9}$$

Note that, if a perfect pattern from a learnt category is presented to the network then there will be no ambiguity which means that the value of A will be zero. In that case also the certainty factor will be less than unity. This is due to the fact that some negative feedback is present at the output layer for which the output, even for a perfect pattern, will be less than unity under stable condition. Mathematically,

$$CF = \rho_0 cf \tag{5.10}$$

where,

$$\rho_0 = \frac{W}{w_s + W} \tag{5.11}$$

is the certainty factor for a perfect pattern, and

$$cf = 1 - \frac{A}{W} \tag{5.12}$$

is the measure of degradation in certainty factor, whenever some noisy or unknown pattern is presented to the network.

Case 1 : Suppose an object has k_1 features. Under learnt condition all bottom-up links have weights w and the top-down links have weights unity. Let a new pattern with $k_1 + k_2$ features be presented to the network. The new feature set consists of all k_1 features of the object. It is considered that the weights of all bottom-up links from these extra features to the output layer are zero, i.e. these extra features are not learned by the network. In that case the features will receive no top- down feedback. Here, the value of A is given as

$$A = \frac{k_1 k_2 w}{k_1 + k_2}.$$

The value of W is

$$W = k_1 w$$

Therefore, from Equn. 5.12

$$cf = 1 - \frac{k_1 k_2 w}{(k_1 + k_2) k_1 w}$$

or

$$cf = \frac{k_1}{k_1 + k_2}.$$

The certainty factor (from Equn. 5.10) can be written as

$$CF_1 = \rho_0 \frac{k}{k_a} \tag{5.13}$$

where k is the number of features of a perfect pattern in the category. In this case this is the same as k_1. k_a is the number of features of the pattern when some extra features are added to it (i.e., $k_a = k_1 + k_2$).

Case 2 : Suppose, the network has learnt an object with $k_1 + k_3$ features. That is, the object is fully activated when all the $k_1 + k_3$ features are presented at the input. Now if only k_1 features are presented to the input, then the question is : what will be the certainty of decision.

In this case the value of A is given by

$$A = \frac{k_1 k_3 w}{k_1}$$

i.e.,

$$A = k_3 w.$$

Now,

$$W = (k_1 + k_3)w.$$

From Equn. 5.12, the value of cf is given as

$$cf = 1 - \frac{k_3 w}{(k_1 + k_3)w}$$

or,

$$cf = \frac{k_1}{k_1 + k_3}$$

Therefore, from Equn. 5.10, the certainty factor is given as

$$CF_2 = \rho_0 \frac{k_s}{k} \tag{5.14}$$

where k has the same meaning as in *case 1*. k_s is the total number of features when some features are deleted from the feature set of the perfect pattern.

Case 3 : Consider a case where both the situations cited above occur simultaneously. That is, an object consists of $k_1 + k_3$ features out of which only k_1 features have been presented. Moreover, a set of k_2 features is presented to the network which is not at all learnt by the network. In that case,

The value of A is given as

$$A = \frac{(k_1 k_2 + k_2 k_3 + k_3 k_1)w}{k_1 + k_2},$$

and

$$W = (k_1 + k_3)w.$$

From Equn. 5.12, the value of cf is given as

$$cf = 1 - \frac{(k_1 k_2 + k_2 k_3 + k_3 k_1)w}{(k_1 + k_2)(k_1 + k_3)w}$$

or,

$$cf = \frac{k_1^2}{(k_1 + k_2)(k_1 + k_3)}.$$

In this case $k_1 = k_s$, $k = k_1 + k_3$, and $k_1 + k_2 + k_3 = k_a$. Therefore, the certainty factor is given as (from Equn. 5.10)

$$CF_3 = \rho_0 \frac{k_s^2}{k(k_s + k_a - k)} \tag{5.15}$$

Some logical reasoning can be derived from the examples cited above. In the first case, some extra features have been presented. Therefore, the entire feature set either collectively corresponds to some other object or it can correspond to more than one object. But the network is only certain about the learnt feature set. Therefore, the certainty factor is the ratio of the learnt feature set and the new input feature set (which is substantiated by Equn. 5.13). The addition of extra features may also be caused by the presence of some kind of additive noise. For example, in a visual recognition system, some extra (undesirable) features may be detected in the preprocessing stage due to noisy environment. In that case also the network will behave in the same way as cited in *case 1* [3].

In the second case, some features have been taken out from the feature set of a perfect pattern. Therefore, the features which were present in the pattern but not in the new input, will be over-emphasized (this is due to the fact that no external input is present for these features but the network will try to support them). Therefore the discarded features will negate the output activation, but the activation will never reduce to zero due to the presence of other features. As a result, there will always be a confusion or ambiguity in the discarded features. Consequently, the certainty of the decision will decrease. Note that, this phenomenon must not be confused with the case of sharing of features. If a feature is shared then the feature may not send activation to some of the output nodes, but the presence of the feature would possibly be perfectly interpreted. On the other hand, in *case 2*, some features are lost altogether. The decrease in certainty about the decision intuitively happens to be the ratio of the size of the new input feature set to the size of the stored template which is substantiated by Equn. 5.14. Note that, the loss of features may be due to the presence of a pattern from a new category. It can also happen due to the presence of subtractive noise. For example, in the case of visual pattern recognition some features may not be detected in the preprocessing task due to presence of noise.

In the third case, some features have been discarded, as well as some extra features have been added. This simultaneous loss of features and presence of extra features may be caused by the presence of a pattern from a new category or due to the presence of both additive and subtractive noise[4]. Equn. 5.15 illustrates the effect

[3] Note that, the structural information necessary for visual recognition is not embedded here. The example is presented only to establish the effect of noise.

[4] Now onwards the presence of both additive and subtractive noise will be referred as the presence of mixed noise

of such kind of mixed noise.

From the above discussion it is clear that the categorization process (whether a new category code has to be formed or not for a given pattern) should be guided by the fact that how well the features are being interpreted i.e., on the certainty factor of the decision regarding the feature set. If a set of features is presented at the input of a network and it is found that the certainty factor is low enough then it can be inferred that the network is not able to interpret (explain) the set of features with the current templates stored in its links. The set of features then can be considered to represent a new object class altogether. In this network whenever a feature set is presented, the certainty factor, after stabilization, is compared with some threshold (ρ) called the vigilance threshold. The way of selecting the threshold will be discussed in the next paragraph. Note that the vigilance threshold has similar effect as that of the vigilance factor in the ART [130].

5.3.5 Vigilance Threshold and Noise Characteristics

The principle of categorization has been discussed logically. In order to select the exact vigilance threshold certain characteristics of categorization need to be investigated. The noise tolerance of the categorization will be determined by the vigilance threshold. *If the vigilance threshold* ρ *is very* high then the system will be very sensitive to noise, and due to the presence of noise a large number of categories will be formed for the same class of patterns. On the other hand, if ρ is very low then more than one class may fall into the same category and therefore, it will affect the weights of the links also, which eventually may give rise to oscillation among categories for the same class of patterns.

Suppose a pattern is contaminated with the same level of different types of noise (additive, subtractive or mixed). Same level of noise means that the number of extra features added to a pattern in case of additive noise, the number of features discarded in case of subtractive noise, and the total number of features which are discarded and added in case of mixed noise are same. In that case,

$$CF_1 > CF_3 > CF_2 \tag{5.16}$$

Proof : Suppose the noise level is $l(< 1)$ and the total number of features is k. Then the total number of features added or discarded (or both) will be kl.

Therefore the CF values in the first two cases can be written as (from Equn. 5.13, 5.14)

$$CF_1 = \rho_0 \frac{1}{1+l} \tag{5.17}$$

and

$$CF_2 = \rho_0(1-l) \tag{5.18}$$

In the 3rd case, let the number of features discarded be pkl. Then the number of extra features added is $(1-pl)k$, since the sum of the numbers of features discarded and added is kl. Therefore, the noise level can be written as (from Equn. 5.15)

$$CF_3 = \rho_0 \frac{(1-pl)^2}{1+(1-2p)l} \tag{5.19}$$

To prove $CF_3 > CF_2$, we have to prove

$$(1-l)(1+l(1-2p)) < (1-pl)^2$$

By algebraic manipulation this reduces to

$$(1-p)^2 l^2 > 0$$

which is true for any real value of p.

Similarly, to prove $CF_3 < CF_1$ we have to prove

$$(1+l)(1-pl)^2 < 1+(1-2p)l$$

By algebraic manipulation this reduces to

$$p^2 - 2p + p^2 l < 0$$

or,

$$p(1+l) < 2$$

Since $l < 1$, and $p < 1$, the expression holds true. Hence proved. □

Suppose, a single pattern with k perfect features each of equal importance has been presented to the network. Then according to Equn 5.11

$$\begin{aligned} \rho_0 &= \frac{kw}{w_s + kw} \\ &> 1 - \frac{w_s}{kw} \end{aligned}$$

This gives an idea about the maximum value of ρ (or ρ_0) that can be allowed for categorizing a perfect pattern. For example, if we consider $kw = 1$ and $w_s = 0.05$ then the value of ρ_0 is less than 0.95.

In *case 1*, the certainty factor of a decision reduces due to the presence of some extra features. These extra features represent some kind of additive noise in a pattern. The value of certainty factor in the case of additive noise is, in fact,

$$CF = \rho_0 cf$$

Suppose the system tolerates 20% additive noise. In that case, value of cf will be (from Equn. 5.13) 1/1.2. Therefore, ρ should be selected less than 0.95/1.2 or 0.792.

In *case 2*, the certainty factor reduces due to the absence of some features which can be viewed as a sort of subtractive noise. Let the system be able to tolerate 20% subtractive noise. From Equn. 5.14 it is clear that cf is equal to 0.8. Therefore, the value of ρ should be less than 0.95×0.8 or 0.76.

In *case 3*, the certainty factor reduces due to presence of both additive and subtractive noise. If the pattern suffers 10% additive and 10% subtractive noise then the value of cf becomes $0.9 \times 0.9/(1.1 + 0.9 - 1)$ or 0.81. Therefore, ρ has to be selected less than 0.95×0.81 or 0.769 which is in between the other two values.

If the system has to tolerate 20% noise in any form then ρ has to be selected nearly 0.76.

Again, the value of ρ determines the capability of the network in distinguishing two different patterns. For low value of ρ, the network may decide two different patterns are from the same category. Therefore the noise tolerance and the distinction capability set the upper and lower bounds for the selection of ρ. If two patterns are sufficiently overlapped then high noise tolerance cannot be achieved which is a common problem in pattern recognition also.

If multiple or mixed (occluded) patterns appear in the scene during categorization then, as seen before, the output will reduce gracefully for some objects. But the decrease in output is not as severe as the decrease in output due to the presence of noise. Therefore, if several objects appear together they are checked in the same way and if the certainty value is less than ρ then the entire feature set is considered to represent a pattern from a new object class.

5.3.6 Issues of Stability

X-tron is able to form stable category codes even under the presence of mixed categories. Unlike SONNET [150], it is not designed to form stable codes for embedded patterns. However it is able to operate under high amount of overlapping and also under noisy environment.

The learning rules (Section 4.4.2) reveal the fact that the initial rate of learning for a new category is very high (because ε_l is nearly unity and $g'(u_l)$ is very low in equation (4.23)). This enables the network to code the incoming pattern at a faster rate. But with several presentations the agility factor of the node decreases and learning rate becomes slow for that particular node. However, in this model there is no such separate scheme for fast and slow learning; the same learning rule behaves differently for different nodes according to the age of that node.

The categorization technique is based on the assumption that the templates presented to the network are perfectly learned. Therefore whenever a new object class is detected by the network, it is iterated for a number of learning trials until the rate of change of weights becomes less than a small quantity ϵ (say). When the change falls below ϵ, the network is ready to accept a new pattern. Learning a particular category for several trials ensures stability in the process of categorization.

This is also necessary because no order search, as done in ART [130], is performed here. The certainty factor is determined completely on the basis of output response and therefore, the network must ensure that the output responses produced are nearly perfect for learnt set of categories. If the weights are not iterated for several trials then patterns from a single class may fall to different categories at different times making the system virtually unstable or indecisive. The system is also able to handle both the additive and the subtractive noise.

The stability of categorization process can be explained as follows. If an existing pattern is presented to the network, the corresponding output node becomes highly active. An output node gets activated only when the weights are learned properly. If the weights are learned properly then the output category would be able to interpret the features also. If the features are properly interpreted then the ambiguity in the hidden layer would be low and no new category will be formed.

In the case of mixed categories also, if the pattern corresponds to more than one

category and the activation values of the corresponding output nodes are high (i.e. exemplar patterns have been learned) then they will be able to interpret the features. The ambiguity is measured based on the maximum feedback that a set of hidden nodes (for a single input node) receive. Therefore the feature will be interpreted by any one of the object categories to which it belongs. As a result, the ambiguity will be low and no new category will be formed.

Let us now consider that the network is presented with a pattern which do not corresponds any single or any combination of the learned exemplar patterns. In that case no single output node will be highly activated because some of its features would be absent and they would inhibit its activation value. Therefore, the maximum feedback from the output layer to a set of hidden nodes (connected to a single input node) would not be very high. Moreover, there may be some features (in the new pattern) which do not belong to any object. For these features there will be no feedback from output layer (since the connections had not been formed). As a result, there will be a high amount of ambiguity in the interpretation of the features, i.e., certainty factor would be low. In that case the network is able to decide that a new pattern has appeared and consequently it allocates a new output node. This enables the network to have its plasticity property.

The maximum output value is compared with a threshold T as an auxiliary condition. This prevents the network from getting confused by ghost patterns. For example, consider that the network is presented with a learned exemplar pattern with very low feature confidence values (say, a pattern 1 1 0 0 1 is presented in the form .3 .3 0 0 .3). In that case the output of the corresponding node will be low, but the outputs of other nodes will be even lower. When the output is fed back to the hidden layer, the mismatch between the input confidence value and the feedback will also be low and as a result, the ambiguity will be low. In that case the network may confuse the pattern as a case of the exemplar and flag that it has recognized it, but this is not desirable. This can be prevented (as mentioned in Section 5.3) by comparing the maximum output with the threshold T.

The most appealing part of the system is that it is able to stably categorize patterns even when they appear in mixed form (i.e. mixed set of patterns). This is because of the fact that no single object is flagged as winner in the output layer. Rather, some of the hidden nodes representing the input-output associations are flagged as winners.

5.4 Results and Analysis

The connectionist model was simulated on SUN 3/60 workstations. Both binary pattern and visual pattern were considered as input. The purpose of the present investigation is to establish the power of the network in categorizing these inputs. Note that, the application of the network in practical domain like 2D object recognition or medical diagnosis etc. needs domain specific knowledge and that is not discussed in this chapter.

5.4.1 Binary Strings

Table 5.1: Patterns are presented against features. Full confidence is represented by 1 and no confidence as 0.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
pat_1	0	0	0	1	0	0	0	1	0	0	0	1	1	1
pat_2	1	1	0	0	1	1	1	0	1	0	0	0	1	0
pat_3	1	0	0	0	0	1	0	0	1	1	1	0	0	0
pat_4	0	1	1	0	1	0	1	0	0	1	0	0	1	1
pat_5	1	1	0	1	1	0	0	0	0	0	0	0	0	1
pat_6	1	1	0	1	1	0	0	1	1	0	0	0	0	0

The set of binary patterns which was presented to the network is shown in Table 5.1. The patterns are chosen arbitrarily. At first, the patterns are presented to the network without any noise. The self-feedback was chosen to be 0.05. The value of γ was selected as 0.15. The constant γ was used to realize Weber's law (as done in ART [130]) in the weight updating rules of bottom-up links as discussed in Section 4.4.1. The threshold T, above which an output category would be considered to be present, was selected as 0.8.

The categorization (self-organization) property of the network was tested for different vigilance thresholds like 0.8, 0.85 and 0.9. Since the patterns were perfect in all cases they were categorized correctly. As an illustration, the weights of the links attained after 600 presentations are shown in Table 5.2. The weights of the bottom-up links represent the relative importance of the features with respect to output categories. Note that, the weights of the top-down links, created in the

network, were unity since the patterns were perfect. Therefore the weights of the top-down links are not presented separately.

Table 5.2: Weights are presented against features and classes. A zero weight indicates that the corresponding link is absent in the network.

	C_1	C_2	C_3	C_4	C_5	C_6
f_1	0.0	0.003	0.087	0.0	0.162	0.121
f_2	0.0	0.003	0.0	0.069	0.170	0.127
f_3	0.0	0.0	0.0	0.316	0.0	0.0
f_4	0.117	0.0	0.0	0.0	0.20	0.149
f_5	0.0	0.003	0.0	0.069	0.170	0.127
f_6	0.0	0.007	0.195	0.0	0.0	0.0
f_7	0.0	0.007	0.0	0.164	0.0	0.0
f_8	0.184	0.0	0.0	0.0	0.0	0.235
f_9	0.0	0.004	0.124	0.0	0.0	0.172
f_{10}	0.0	0.0	0.194	0.146	0.0	0.0
f_{11}	0.0	0.0	0.344	0.0	0.0	0.0
f_{12}	0.366	0.0	0.0	0.0	0.0	0.0
f_{13}	0.146	0.005	0.0	0.101	0.0	0.0
f_{14}	0.126	0.0	0.0	0.088	0.216	0.0

Effect of Noise on CF

Table 5.3: Certainty factors for different noise levels on the patterns

case	*pattern*														*certainty factor*
1	0	0	0	1	0	0	0	1	0	0	0	1	1	1	0.9494
2	0	0	0	0.9	0	0	0	0.9	0	0	0	0.9	0.9	0.9	0.9494
3	0.1	0.1	0.1	1	0.1	0.1	0.1	1	0.1	0.1	0.1	1	1	1	0.9492
4	0.1	0.1	0.1	0.9	0.1	0.1	0.1	0.9	0.1	0.1	0.1	0.9	0.9	0.9	0.9491
5	1	0	0	1	0	0	0	1	0	0	0	1	1	1	0.8401
6	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0.8220
7	1	0	0	1	0	0	0	1	0	0	0	1	1	0	0.7163
8	1	0	0	0	0	1	0	0	1	1	1	0	0	0	0.9497
9	0.9	0	0	0	0	0.9	0	0	0.9	0.9	0.9	0	0	0	0.9497
10	1	0.1	0.1	0.1	0.1	1	0.1	0.1	1	1	1	0.1	0.1	0.1	0.9493
11	0.9	0.1	0.1	0.1	0.1	0.9	0.1	0.1	0.9	0.9	0.9	0.1	0.1	0.1	0.9492
12	1	0	0	0	0	1	0	0	1	1	1	0	0	1	0.8361
13	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0.8249
14	1	0	0	0	0	1	0	0	0	1	1	0	0	1	0.7136

In the next phase of experiment the effect of noise on a particular pattern was studied. For this purpose input patterns were contaminated with various levels of additive and subtractive noise. The effect of noise on the certainty factor for the concerned pattern is presented in Table 5.3. From this table it is clear that

when a perfect pattern (case 1 in Table 5.3) from the first category of Table 5.1 was presented then CF was 0.9494 which is the value of ρ_0 (Equn. 5.11). The confidence values of the features were replaced by 0.9 and it was found that CF retains the same value. This is due to the fact that in finding CF, the total ambiguity is always normalized by the total input confidence (Equn. 5.12). Therefore, apparently it seems that even if the confidence values of the features, which were present in the perfect pattern, go on decreasing, the value of CF will retain its original value (ρ_0). This is the basic reason for which the value of output threshold was always considered in the process of self-organization. Although CF retains its original value, the maximum output goes on decreasing with the decrease in the confidence values of the features. Since the output decreases proportionately with the confidence values of the features, the average ambiguity remains constant and thereby CF retains its value.

The pattern was again contaminated with some additive noise where all the features which were actually absent have now confidence value equal to 0.1(case 3). In that case it (Table 5.3) shows that CF slightly decreases. When both the noise are present simultaneously then also CF decreases slightly. In fact, in all three cases the maximum output in the output layer will determine if the noisy pattern is from the present category or from a new one, during the process self-organization.

We have then checked the performance of the network by adding one extra feature to the pattern, namely the first feature. Results (case 5) show that CF decreases by a great extent, from 0.94 to 0.84. This is due to the fact that there was no top-down support to that feature from the category concerned. Therefore the ambiguity at that feature is very high and hence the average ambiguity also increases and, as a result, CF decreases. Note that, in case 4 (Table 5.3), the effect of noise was not very high to any particular feature, and therefore there was a graceful degradation in the performance, which is not true in the present one (case 5). Case 6 shows the performance when a feature is deleted from the pattern, namely, the last feature. Result indicates a drastic reduction in CF from 0.94 to 0.82. This has also the same explanation as in case 5. Case 7 presents the result when an extra feature is added and a genuine feature is discarded. In effect, CF reduces from 0.94 to 0.72.

Note that the objective of the investigations performed through cases 1-7 is to establish the nature of CF and not the nature of maximum output. Because in the categorization process the output threshold is always kept fixed (chosen as 0.8 with negative self-feedback of 0.05), and the categorization property was studied

with different vigilance thresholds. Cases 8-14 in Table 5.3 show the similar effects on another pattern, namely the pattern from 3rd category in Table 5.1.

Categorization of Individual Patterns

Table 5.4: Results of categorization when patterns are presented individually with vigilance factor = 0.9 and noise level = 0.3.

Actual class	*category*																	
1	1	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	6	0	1	1	26	0	0	8	0	0	1
3	0	0	1	0	0	0	0	0	41	0	0	0	0	0	0	0	1	0
4	0	1	0	39	0	0	0	0	0	0	0	0	1	0	0	1	0	2
5	0	0	2	0	32	0	0	0	0	0	0	0	0	10	0	0	0	0
6	0	0	0	0	0	44	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.5: Results of categorization when patterns are presented individually with vigilance threshold = 0.65 and noise level = 0.3.

Actual class	*category*						
1	1	0	0	0	0	49	0
2	0	50	0	0	0	0	0
3	0	0	50	0	0	0	0
4	0	0	0	50	0	0	0
5	0	0	0	0	1	0	49
6	50	0	0	0	0	0	0

In this experiment, the values of self-feedback and γ are chosen to be the same as before. It is found that when the patterns are contaminated with 10% and 20% random noise, the network is able to categorize them perfectly with vigilance thresholds 0.8 and 0.9. Here, 10% noise means the maximum level of noise is 10% of the maximum confidence value that can occur. For example, in the present experiment, the maximum confidence value is unity, and hence maximum noise level (additive or subtractive) can be 0.1. When the patterns were contaminated with 30% noise the network is able to categorize correctly with a vigilance threshold (ρ) equal to 0.8. On the other hand, if ρ is selected as 0.9 then it is seen that a large number of categories are formed. Since perfect categorization occurs in the

first two cases the results are not presented here. Table 5.4 shows the results of categorization with 30% noise and ρ equal to 0.9. It may be noted that T is always kept fixed at 0.8. From the table it is clear that although quite a few redundant categories have been formed, there are only six distinct categories into which the patterns fall most of the times. This shows that although the network is initially confused with such a high noise level and high value of ρ, it has been able to associate a particular node with a particular category finally.

The network behavior is then studied with a low value of vigilant threshold (ρ). The patterns were again contaminated with a high noise level of 30% and ρ was selected to be 0.65. Table 5.5 shows the categorization performance of the network. The results show that the network has been able to categorize almost perfectly. Only two classes (1 and 6) fell into the same category initially, but afterwards, class 1 was allocated to some new category.

Categorization of Mixed Patterns

Table 5.6: Results of categorization when individual and mixed patterns appear randomly to the network. 'cls' stands for class and 'ctg' stands for category.

	ctg1	ctg2	ctg3	ctg4	ctg5	ctg6	ctg7
cls1	0	0	0	0	58	0	0
cls2	0	0	0	0	0	63	0
cls3	0	0	60	0	0	0	0
cls4	74	0	0	0	0	0	0
cls5	0	0	0	56	0	0	0
cls6	0	1	0	0	0	0	58
cls1&2	0	0	0	1	6	5	3
cls1&3	0	0	6	0	6	0	0
cls1&4	6	0	0	0	6	0	0
cls1&5	0	0	0	3	3	0	0
cls1&6	0	0	0	1	4	0	3
cls2&3	0	0	4	0	0	4	0
cls2&4	7	0	0	0	0	7	0
cls2&5	0	0	0	6	0	6	0
cls2&6	0	0	0	0	0	7	7
cls3&4	5	0	5	0	0	3	0
cls3&5	0	0	8	8	0	0	0
cls3&6	0	0	3	0	0	0	3
cls4&5	6	0	0	6	0	0	0
cls4&6	2	0	0	0	0	0	2
cls5&6	0	0	0	5	0	0	7

Here the capability of the network to categorize both the individual and mixed patterns simultaneously is studied. The results corresponding to 10% noise level and 0.8 vigilance threshold are shown in Table 5.6 when the patterns were presented in random order. Note that if a mixed pattern appears it was treated either as a new category or as a superposition of two different categories. In Table 5.6 the output categories are presented against the actual classes.

It is clear from Table 5.6 that whenever the pattern from an individual class appears it has been categorized perfectly. For example, all 74 patterns from class 4 have been identified as the first category. Only one pattern from class 6 fell into second category which is really a redundant one. Whenever, mixed patterns appear, they are identified either as a new one or as a mixture of the learned categories. For example, whenever a mixture of patterns from classes 1 and 5 appears, it is identified by the network as a mixture of categories 4 and 5. This is, of course, a correct decision because all patterns from class 1 fell into 5th category and those from class 5 went to 4th category. In Table 5.6, the most confusing result corresponds to the entries for the mixture of classes 1 and 2. Note that, patterns from class 1 and class 2 went to categories 5 and 6 respectively. This reveals the fact that these categories were learned after the first four (one of them is redundant) which correspond to classes 3, 4 and 5. Initially, when a mixture from classes 1 and 2 appears, the network was not able identify them as a mixed category and therefore a new category was allocated (category 4). Later, this category is taken over by the patterns from class 5. Whenever, the network becomes able to categorize the patterns from classes 1 and 2, the mixture of 1 and 2 is no more confusing. The network is able to identify the mixture 5 times (from Table 5.6). Results show that the network is able to identify the mixtures of other patterns also.

5.4.2 Visual Patterns

The visual patterns presented to X-tron for categorization, are already shown in Fig. 4.2. Here, the parameters of the network, namely, amount of self-feedback, γ are selected to be the same as before. The visual patterns are contaminated by both additive and subtractive noise and then categorized by the network with a vigilance threshold equal to 0.8 and output threshold 0.8. It is found that the network is capable of correctly categorizing the patterns (both individual and mixed) even in the presence of 30% noise. As an illustration, the results for 20% contamination

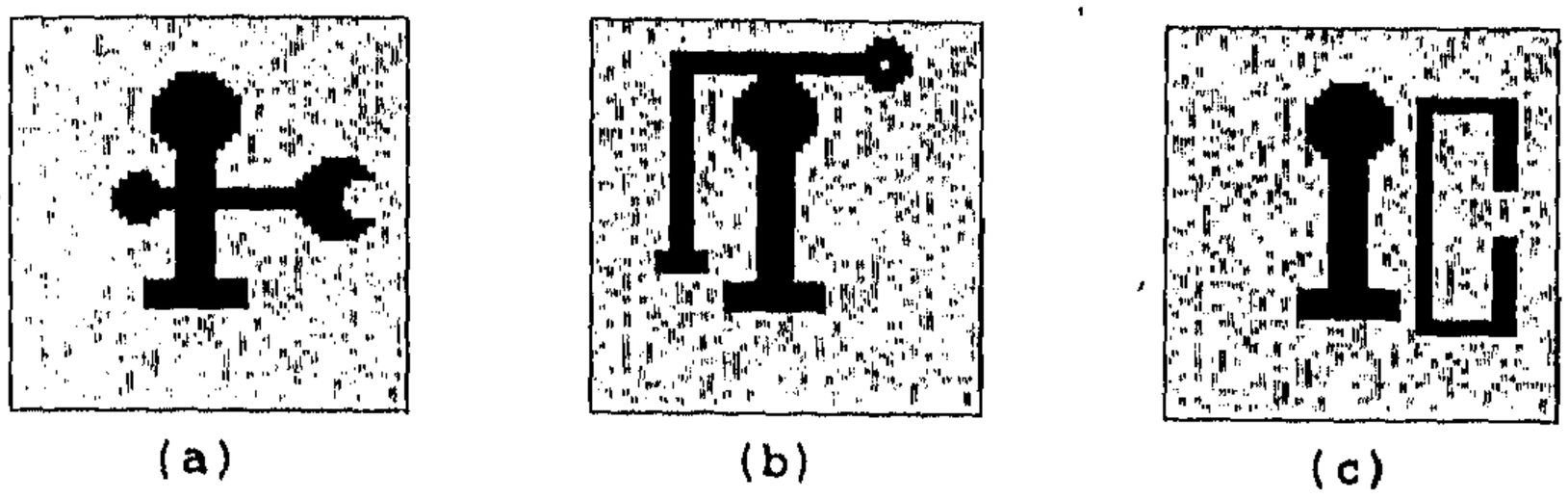


Figure 5.2: Mixed patterns with 20% noise level. (a),(b),(c),(d),(e) and (f) represent the mixture of objects 1 & 2, 1 & 3, 1 & 4, 2 & 3, 2 & 4, and 3 & 4 respectively.

Table 5.7: Confusion matrix when visual patterns are presented with vigilance threshold = 0.8 and noise level = 0.2. 'cls' stands for class and 'ctg' stands for category.

	ctg1	ctg2	ctg3	ctg4
cls1	0	12	0	0
cls2	23	0	0	0
cls3	0	0	19	0
cls4	0	0	0	18
cls1& 2	7	7	0	0
cls1& 3	0	3	3	0
cls1& 4	0	5	0	5
cls2& 3	4	0	4	0
cls2& 4	2	0	0	2
cls3& 4	0	0	7	7

(Fig. 5.2) are presented in Table 5.7.

5.5 Conclusions and Discussion

In this chapter, the capability of X-tron in categorizing different patterns either in single or mixed form has been investigated. An ambiguity measure, depending on the interpretation of the input features, is defined, based on which the categorization process has been formulated. The network architecture is also accordingly modified. The network automatically adjusts the number of nodes in the hidden and output layers, depending on the complexity or nature of overlap between the patterns. Note that the entire structure of X-tron basically follows from the mathematical formulation (equation 4.8) which was derived for properly interpreting a feature set.

The effectiveness of the method is demonstrated for both binary and visual patterns in presence of additive, subtractive and mixed noise. Note that, the binary strings considered in the present investigation, have a higher degree of overlap between themselves compared to the visual patterns. As a consequence, the categorization of the binary strings (individual or mixed patterns) seemed to be more difficult than the visual patterns. That is why the characteristics of the network has been studied in detail on the binary strings in order to establish the exactitude of the self-organizing capability. The categorization results for visual inputs also have been provided.

The characteristics of the proposed model well compares with the self-organization property as shown in adaptive resonance theory. First, in ART an order search is necessary among the output categories to find if one of them interprets (explains) the input pattern. In the proposed system no such search is necessary. This is due to the fact that the ambiguity is in the featural level and the activations of all output nodes are kept in tact; none of them are disabled. Second, ART does not deal with multiple or mixed categories. On the other hand, if a mixture of known categories appears at the input, the present network is able to detect it and no new category is formed. Note that the system provides output in continuum grades (fuzzy). The concept of continuum grades was also considered in ART2, ART3 and fuzzy ARTMAP for forming stable codes for individual categories.

X-tron is able to self-organize in the presence of mixed categories. The model proposed by Cho and Reggia [142] can also do the mixed category recognition, but the learning process, in that model, operates only under supervised mode. Moreover, the way X-tron operates is entirely different from that proposed in [142]. The system developed by Marshall [144], [145], [146] does the task of mixed category perception by limiting the competition process within similar types of nodes, whereas X-tron performs this task (with high overlap between the patterns) by dissociating the competition process from the output layer. SONNET [149], [150], [152] is able to form stable categories for embedded patterns. X-tron forms stable category only when it finds the pattern separately (and the problem of embedded patterns has not been addressed). Note that SONNET uses a concept of competition between the links for associating a feature with an object, whereas the process of competition takes place in the hidden layer of X-tron. Cohen and Grossberg [147], [148] have considered the mixed category recognition problem using masking fields, but their network does not form any stable category.

X-tron is also comparable with other self-organization models (which do not perform the task of mixed category recognition) in a similar line. For example, in Neocognitron [175], the case where the feature set of a category is a proper subset of the feature set of another category, was solved by employing inhibitory connections. On the other hand, the same phenomenon has been solved here by using Weber's law as in the case of ART. If inhibitory connections were used then the activation of the genuine categories would have reduced if a pattern representing a set of mixed categories is presented to the network. This is due to the fact that the inhibitory connection will ensure that if a redundant feature is presented at

the input, it will reduce the output of a category. In the case of a mixed pattern, the features of one pattern would play the role of redundant features with respect to other patterns (provided the feature is not shared). Therefore, the activation of the true categories would decrease to a great extent if the mechanism of inhibitory connections were embedded.

Although the network has been tested on synthetic data (binary and visual patterns), it can be used to deal with the real life object recognition problem under occlusion, partial information loss or noisy environment. It may be mentioned here that in practical visual recognition problem, the degree of high overlap, as considered for binary strings, may be rarely possible. For example, if we use 'corner' as features for object recognition then it is unlikely that a particular corner feature is shared by more than one object in the scene. The results on binary strings indicate the fact that if the structural information from the feature space to the object space can be embedded into the connections of the network, then it can possibly be applied to object recognition tasks. ♠

In Chapters 6 and 7, connectionist systems for object recognition are developed by exploiting the principle of X-tron and incorporating the structural information from feature space to object space.

Chapter 6

Primitive Extraction, Structural Learning and Recognition

6.1 Introduction

As mentioned in Chapter 1 and Chapter 3, recognition of the structures involves the matching of a candidate structure with some prototype structures stored in the model-base. The main difficulty in structure matching problems is that the presence of noise (and/or vagueness) may change the description of some of the primitives, thereby affecting the matching performance. Assigning some weights to the primitives and to the relations (reflecting their importance in characterizing various classes) may therefore help to an extent in achieving noise tolerance and in handling impreciseness in input. These weights will be higher for the primitives and relations which are more consistent (i.e. important) in characterizing a class.

Structural description is widely used in different problems like shape matching, stereo matching, character recognition etc. In all these problems descriptions should be such that the effect of noise gracefully degrades the performance of the system. Therefore, to design a recognition system based on structural description, one should pay attention to the proper extraction of the primitives and assignment of weights to the primitives and the relations. The extracted primitives (features) should be as robust as possible. Moreover, the system should be able to assign these weights automatically (supervised or unsupervised learning).

In this connection, we mention the method, described in Chapter 3 for matching structural descriptions between prototype and candidate objects using Hopfield model, where learning the saliencies of the primitives (i.e., automatically acquiring the degree of importance of the primitives) was not possible. Moreover, in the process of construction of the attributed relational graphs, the linguistic descriptions of the primitives were considered, and no technique for automatic extraction of the primitives from an image had been mentioned.

The investigation, made in this chapter, has two parts. In the first part, we present a method for simultaneous extraction of multiple linear segments (primitives/features) from an image by integrating the principles of X-tron (Chapter 4) and principle of edge/line linking (Chapter 2) [212]–[216]. The significance of the primitives, thus extracted, for structural learning and recognition, is then established in the second part with a multilayer perceptron. The problem of handwritten Bengali character recognition has been considered as a candidate for demonstrating the effectiveness of this system. Particularly, characters of similar shapes and their various distorted versions are considered for illustrating the discriminating ability of the system.

In literature, there exist various approaches based on neural networks for character recognition problem including the attempts made by Fukushima [175], [171]. These can be classified into two groups. In the first category [228], [229], [230], [231], [232], classification or matching task is based on the derived features from the character images. Here, it is necessary that the features should be invariant under noisy, ambiguous environment. The second category [175], [171], [233], [234], [235], on the other hand, does not need to extract the features separately; the classification is performed directly from the pixel level information. Here, most of the methods use MLP for generating complex decision regions.

In our proposed method, we have exploited the merits of robust feature extraction and the characteristics of MLP in integrating the features. We have extracted linear structures in an image and these structures are fed to MLP as input. The linear segments are specified by their position, orientation, and extent to the MLP model which recognizes the structures present in the scene depending on the nature of linear segments.

Principle of Hough transform [64] is used to extract linear segments. An iterative verification scheme is used to efficiently implement Hough transform within

a connectionist framework. A three-layered model, which is in principle, similar to X-tron, is developed to extract out linear segments/structures simultaneously (multiple peaks in Hough space are simultaneously detected instead of multiple objects). However, in this case, no learning is required to detect the peaks in Hough space. In literature, there exists some approaches [236] for detection of multiple peaks in the multidimensional parameter space. However, the present approach detects the peaks in association with the line strength and directions present in the image. In other words, the proposed technique verifies the existence of the peaks iteratively to interpret the existence of the line points in the image. On the other hand, the existing approach [236] employs a cooperative and competitive process in the parameter space itself and detects the peaks depending on their distribution in the parameter space. The principle of line linking (as described in Chapter 2) is also used in designing the proposed model to enhance the extraction process. A three-layered MLP model is cascaded with the proposed network to design a six-layered connectionist system for both feature extraction and recognition.

The system accepts skeleton version of images (regions) as input unlike the method described in Chapter 3 where linguistic descriptions of the primitives were used as input. The extracted features (linear structures) are hierarchically integrated with the help of a multilayered perceptron model. In this case, the network has been designed with a view to reducing redundancy in the number of connections. The nearby linear structures (close in position and orientations) have been grouped hierarchically. It may be mentioned here that in an existing approach for character recognition [237], a modified version of Hough transform had been applied to detect strokes in the characters. The characters were recognized by employing a dynamic programming technique in the parameter space. However, the present approach provides an efficient scheme for implementation of Hough transform in connectionist paradigm and also learns the structural patterns with the help of an MLP model.

The rest of this chapter is organized as follows. The overall methodology of simultaneous extraction of multiple linear segments and consequently, the way of using MLP for integrating these segments (for learning and recognition purpose) have been described in Section 6.2. The structure of the system is described in Section 6.3. The way of finding the line points based on the principles described in Chapter 2, is provided in Section 6.4. The technique of simultaneous extraction of multiple linear segments (primitives/features) using the principle of X-tron is

described in Section 6.5. The process of integrating the features with the help of an MLP is described in Section 6.6. Some results on the performance of the system for handwritten Bengali character recognition are provided in Section 6.7. Concluding remarks are provided in Section 6.8.

6.2 Principle of Feature Extraction and Recognition

In describing the principle of structural learning and recognition process, we consider the structures to be composed of mostly linear segments. The line points in a scene can be found out by matching suitable templates, and these line points can be aggregated according to line response values and directions to extract out linear structures present in the scene. The process of finding out line segments can be enhanced by linking line points over some neighborhood. The larger the size of the neighborhood, higher will be the invariance and lower will be the details of the feature information and smaller the size of the neighborhood, higher will be the effect of noise. This argument holds good for any structural shape recognition problem.

Block diagram (Fig.6.1) shows the basic operations of the system. The System

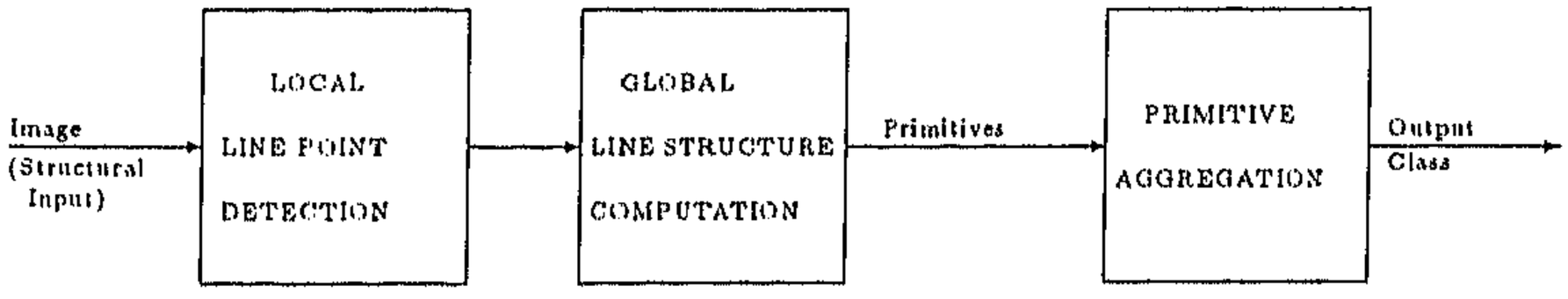


Figure 6.1: Block diagram showing the basic operations of the structural learning and recognition system.

first finds out the local line points present in the skeletonized structures in the image. This is performed with the help of template matching. These templates are efficiently embedded within the links between the first and second layer of the system. The directions and response values of the line points are smoothed by the information present over some neighborhood. The local line points are then grouped according to their orientations in the third layer of the system. This

grouping is performed with the help of Hough transform. An efficient scheme for implementation of Hough transform in the connectionist framework has been designed. The activations in the third layer representing the linear structures are then grouped hierarchically with an MLP model. Let us now briefly describe the concept of Hough transform.

6.2.1 Hough Transform and Feature Extraction

The Hough transform works as follows. Each point lying on a straight line in the image space (corresponding to nonzero pixels [1]) can be expressed as

$$r = x\cos\theta + y\sin\theta, \tag{6.1}$$

where (x, y) is the coordinate of the concerned point in the image space, r and θ are parameters specifying the line, (r is the normal distance of the line from origin and θ is the angle subtended by the normal with the positive x-axis). According to this equation, several (r, θ) values can be computed depending on the (x, y) value of a point. As a result, a point in the image space corresponds to a line (sinusoidal in nature) in (r, θ) space. For each such point in the image space, the cumulative contribution (accumulator value [64]) in the (r, θ) space is computed. If there exists a straight line in the image space then the contribution to a particular (r, θ) value would be very high because all points lying on the line in image space would produce some contribution to that (r, θ) value. In practice, a cluster of activations in the (r, θ) space would be formed due to the presence of a line in the image space. Therefore, if the clusters in (r, θ) space (i.e., parameter space) can be identified then the possible lines in the image space would be detected. In other words, if the local line direction at a pixel in the image space is specified then it should be transformed to a unique point in the Hough space. Let us consider this point (in the Hough space) to be the representative point of that pixel. In the following claim we will consider only the representative points. A straight line in the image space can also be identified as

$$y = mx + c \tag{6.2}$$

where m (slope of the line) and c (intercept of the line with the y-axis) are the parameter values. However the equations (6.1) and (6.2) essentially represent the

[1] In this investigation nonzero pixels correspond to object region

same phenomenon. Next we present an interesting property of Hough transform in (m, c) space.

Claim : If a curve in the image space is continuous and second order differentiable, then the points lying on the curve will lie in a contiguous space after Hough transformation.

Proof : Suppose the image contains a curve given by $y = f(x)$. For the sake of simplicity, we consider the image space to be continuous. Consider a point (x_0, y_0) on the curve. The equation of the tangent to the curve at (x_0, y_0) can be written as

$$y - y_0 = (x - x_0)f^{'}(x_0)$$

i.e.,

$$y = [y_0 - x_0 f^{'}(x_0)] + x f^{'}(x_0). \tag{6.3}$$

In other words, in the parameter space the point (x_0, y_0) will be transformed to (m_1, c_1) where

$$m_1 = f^{'}(x_0)$$

and

$$c_1 = y_0 - x_0 f^{'}(x_0).$$

Now let us consider a nearby point on the curve $(x_0 + \Delta x, y_0 + \Delta y)$. where Δx and Δy are sufficiently small. Suppose this point maps to a point (m_2, c_2) in the parameter space. In a similar way it can be shown that

$$m_2 = f^{'}(x_0 + \Delta x)$$

$$c_2 = y_0 + \Delta y - (x_0 + \Delta x)f^{'}(x_0 + \Delta x).$$

Moreover, Δy can be written as

$$\Delta y = \Delta x f^{'}(x_0)$$

Therefore, it can be written that

$$m_2 - m_1 = \Delta x f^{''}(x_0)$$

and

$$c_2 - c_1 = \Delta y - x_0 \Delta x f^{''}(x_0) - \Delta x f^{'}(x_0 + \Delta x).$$

Since the first and second derivatives exist and are finite, values of $c_2 - c_1$ and $m_2 - m_1$ can be small enough by selecting Δx to be arbitrarily small. In other

words, the nearby points on the curve in the (x, y) space will lie in contiguous space in the transformed region. □

It is to be noted here that instead of considering (m, c) space one can consider (r, θ) space also. In fact the second one has been used in the subsequent discussion. The equation for transforming the image space into (r, θ) space is given by (6.1). It is obvious that if a curve occupies contiguous space in the (m, c) domain then it will occupy contiguous space in the (r, θ) domain also.

If the parameter space is divided into a number of slots, then each slot corresponds to a particular straight line segment. The contribution (accumulator value) present in each slot represents the total number of pixels present in the corresponding line segment in the image space. Because of the above *claim*, a curve in the image space contributes to a contiguous chain of slots in the parameter space. This is equivalent to an approximation of a curve by a sequence of line segments. The level of approximation is dependent on the size of slots.

A linear structure in the image would correspond to a cluster, in addition to some spurious activations in Hough space. It is therefore necessary to extract (segment) the clusters out from the spurious responses. But the selection of the proper threshold for segmentation is a problem. Moreover, since the contribution of a curve in image space is distributed over a sequence of slots in the parameter space, the response values of some of the slots may be low and as a result, get eliminated due to thresholding. Lowering the threshold value, on the other hand, may not be able to eliminate some of the spurious responses. However, Hough transform has an ample scope for massive parallelism because the contribution to each slot can be computed independent of the others.

Since neural networks provide a robust, massively parallel computational framework, Hough transform can possibly be efficiently implemented within a connectionist framework. Moreover, the problem of selecting suitable threshold may be avoided in a connectionist framework with the help of iterative verification method, i.e., using the principle of X-tron. In the connectionist implementation, a neuron is allocated to each slot. The activation of each neuron essentially represents the amount of contributions to that slot. The clusters in the Hough space depend on the (x, y) values of the nonzero pixels in the image space. This can be implemented with a fixed network structure. The dynamics of the network is discussed in Section 6.5 in detail.

6.2.2 Concept of Primitive Aggregation

The activations in the Hough space represent the basic primitives of the structures. The basic primitives should be suitably integrated to represent the higher level features (information) with better noise invariance (robust features). Since the pattern of integrating the features is highly dependent on the type of classes, it will be convenient to learn these patterns in a hierarchical (layered) connectionist model under supervised mode.

To provide better noise invariance the primitives should be grouped over local neighborhood (in the Hough space). This is due to the fact that even if the primitive varies in its position (in the Hough space) due to presence of noise, the effect would be reduced in the next layer of the hierarchy. Moreover, this kind of grouping may provide insensitivity to small amount of orientations of the structures.

The hierarchical structure of the system should also be able to extract out more invariant properties from a group of primitives. Possibly, this can be performed by grouping the primitives over larger neighborhoods. A variation of multilayered perceptron (with suitably selected number of hidden layers and nodes, the connections between the layers being constrained within local neighborhoods) may be used. This is described in the following sections.

6.3 Structure of the System

The proposed connectionist system consists of six layers (Fig. 6.2). The first three layers of the proposed system is structurally similar to X-tron. The input layer of the network contains a two dimensional array of neurons. The size of the array is the same as that of the image (say $I \times J$). Each neuron accepts an activation value equal to the normalized intensity ([0,1]) of the corresponding pixel. In the second layer there are sixteen neurons corresponding to each input neuron (i.e., second layer contains $16 \times I \times J$ neurons). The second layer associates the image space with the parameter space and each group of sixteen neurons in the second layer corresponds to the sixteen templates (the corresponding line segments are shown in Fig.2.1 (Section 2.3.1)). The third layer corresponds to the parameter space. Each neuron in the third layer represents a slot in the parameter space of Hough

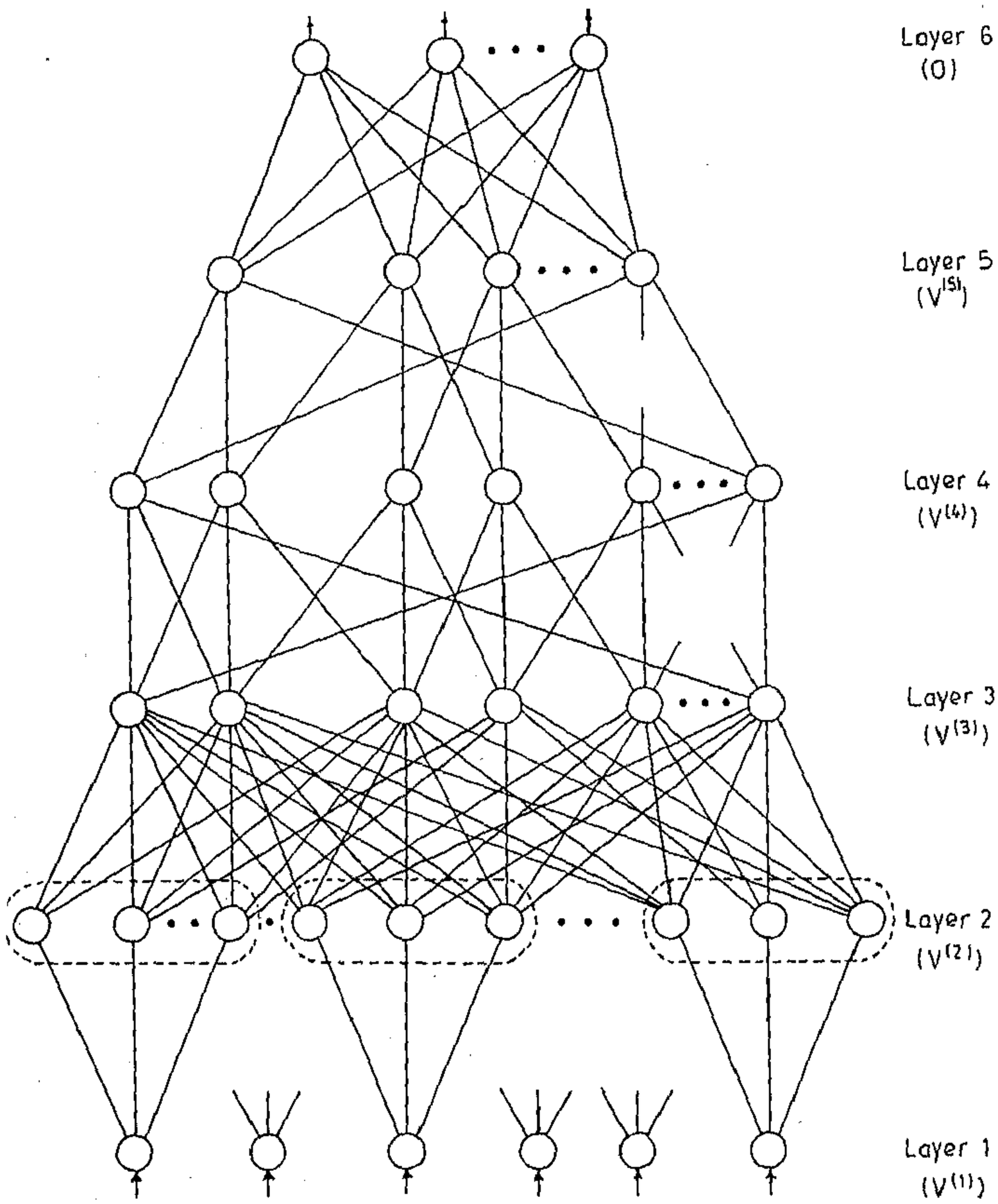


Figure 6.2 cont'd.

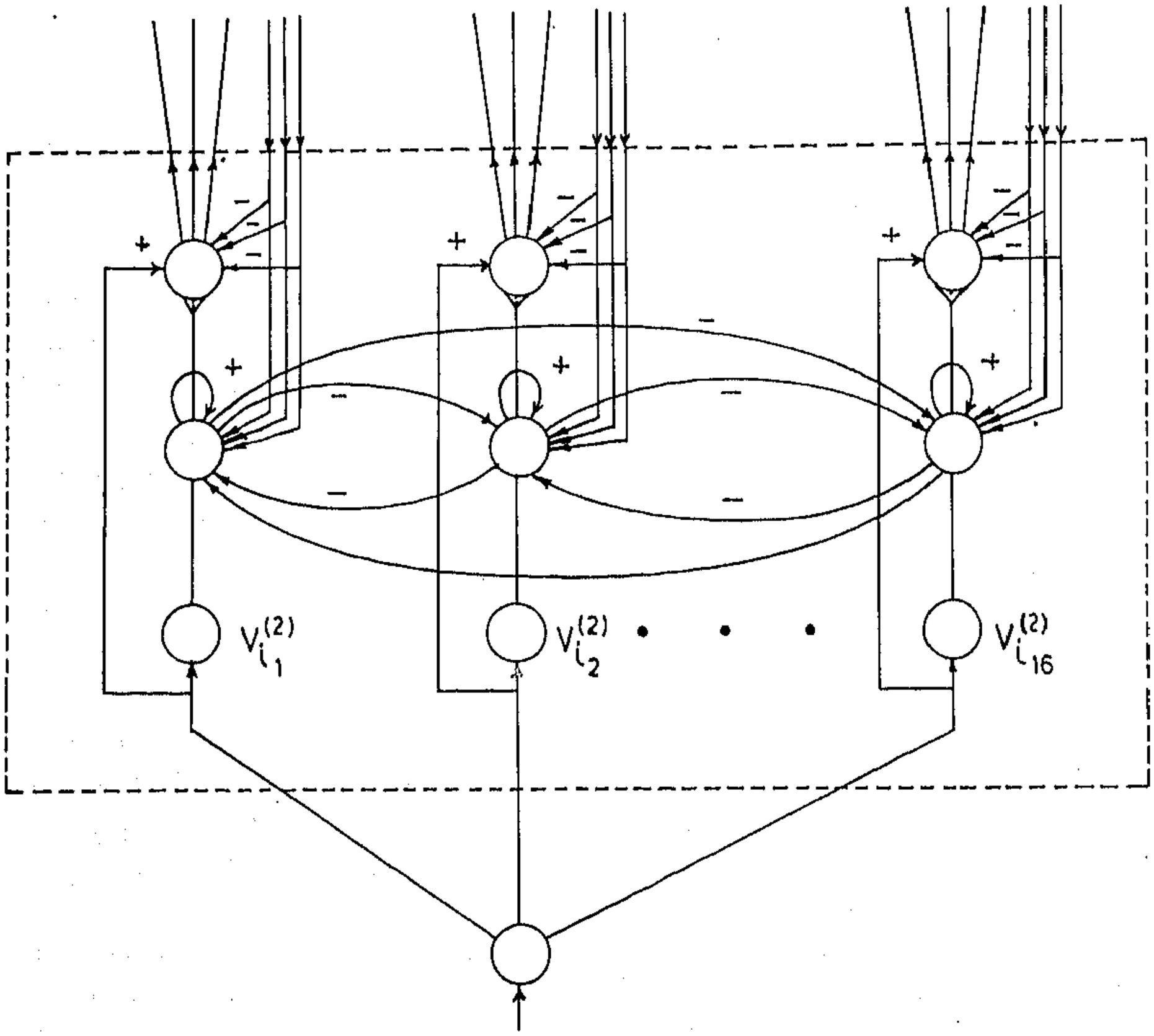


Figure 6.2: (a) Schematic diagram of the structure of the system. The layers of the network are two dimensional. Each link from the second to the third layer represents a bottom-up and a top-down link. The self-feedback connections in the third layer are not explicitly shown. The dashed box in the second layer indicate a group of nodes connected to same input node. (b) Connections between second layer nodes connected to the same input node. The structure of each node is shown explicitly. The part which competes with other nodes is represented as a conjugation of two nodes.

transform. This layer essentially approximates the structures (single pixel thick skeletons of characters) by line segments. The size of the third layer depends on allowed resolution i.e., the chosen slot size in the parameter space.

The connections between first and second layer have been exaggerated in Fig.6.2(b). Each group of sixteen second layer neurons, connected to a single input neuron, is presented within the dotted box. Each second layer neuron has three parts, as shown in the figure. The first part holds the activation value corresponding to the respective line template. The position of a second layer neuron, within a group, distinctly characterize the type of the template, it corresponds to. For example, the first neuron corresponds to the first template, the second one corresponds to the second template and so on. The activation level received by a second layer neuron is determined by the respective template connections to the input layer and input node activations. The second part of each neuron competes with other neurons within the same group. The output of the second part always modulates the activation level of the third part (henceforth, activation level of third part will be referred as the activation level of the second layer neuron itself). If the second part of some neuron loses the competition, then the third part becomes inactive, and if the second part wins, then the third part of the neuron becomes active. The third part of each neuron computes the difference between the signals coming from the input layer and the maximum feedback it is receiving from the third layer. The difference of these two signals are sent to the third layer if the corresponding neuron is a winner within its group.

The fourth layer of the network takes activations from the third and groups (smoothes) them over local neighborhoods. The objective of the fourth layer is to smooth the activation values present in the third layer in order to achieve robustness of the system. The fifth layer also groups activations from the fourth layer over local neighborhoods. This layer is intended to integrate linear segments over local neighborhoods to find some invariant structural properties. The size of the fourth layer is the same as that of the third layer. On the other hand, the size of the fifth layer depends on the chosen neighborhood size and the amount of overlap between local neighborhoods. (The size of the local neighborhood in the fourth and fifth layers, and the amount of overlap in the fifth layer will be discussed in Section 6.5.) The number of neurons in the sixth or the output layer is equal to the number of classes (structures) to be learned and recognized. Each neuron in the output layer has connection with all nodes in the fifth layer.

The input layer accepts images of skeletonized structures. As mentioned before, each neuron in the input layer is connected to sixteen neurons in the second layer. All sixteen neurons within a group in the second layer have competition between them. Each neuron in the second layer is connected to all neurons in the third layer through bottom-up and top-down links. The bottom-up links carry activations from the second to the third layer and the third layer feeds back the activation to the second through the top-down links. Each neuron in both second and third layers has a (r, θ) value associated with it. The (r, θ) value actually determines to which neuron it should send activation and from which it should receive. A neuron in the third layer would receive activation from a neuron in the second layer when there is a match between the (r, θ) values resident in these neurons. Each third layer neuron has also a negative self-feedback connection.

In the initialization process, the local line directions are computed in the second layer using the template weights embedded into the links from first layer to second layer. The line response values and directions are smoothed over some local neighborhoods (this is performed with the same principle as described in Section 2.3.2). The neurons in the third layer are activated from the neurons in the second layer through bottom-up links. After initialization, the nodes in the third layer feed back the activation values to the second layer through the top-down links. The second layer nodes, in turn, send the differential support to the nodes in the third layer. The activations of the nodes in the third layer are updated according to the negative self-feedback and the differential support. In this process, the spurious activations do not receive any differential support and are reduced due to negative self-feedback. On the other hand, the true activations attain stable states when the differential support equals the negative self-feedback.

After stabilization the third layer represents the clusters (corresponding to linear segments in the image space) in the parameter space. The weights of the links from the first to the second and from second to the third layer are fixed. Note that, in X-tron, weights of top-down and bottom-up links are learned depending on the co-occurrences of features and objects. However, in the present system, weights of the links between second and third layers are kept fixed. The weights of the links from the third to fourth, fourth to fifth, and fifth to sixth layer are learned using the backpropagation learning rule. If the size of the neighborhood in the fourth and fifth layers is very large then there will be redundancy in the network and the backpropagation technique would require more time to converge. On the other

hand, if the neighborhood size is very small then the network may not be able to extract out the invariant properties of the structures. As a result, the performance of the network may be deteriorated. The selection of the approximate size of the neighborhood has been discussed in Section 6.5.

The following notations are used in the subsequent discussions. The activation value of the i^{th} input (first layer) neuron is represented by $v_i^{(1)}$. The second layer neurons corresponding to i^{th} input neuron are indexed as i_j (i.e., the j^{th} neuron among the sixteen neurons connected to the i^{th} input neuron). The notations followed here are not exactly the same as that used in X-tron. This is because the number of hidden nodes for each input node, used here, is fixed and do not depend on how many third layer neurons are connected to the hidden layer. The activation value of the i_j^{th} neuron in the second layer is denoted by $v_{i_j}^{(2)}$. The r and θ values present in i_j^{th} neuron in second layer are denoted by $r_{i_j}^{(2)}$ and $\theta_{i_j}^{(2)}$ respectively. Similarly, the activation value, r, and θ stored in the i^{th} third layer neuron are denoted by $v_i^{(3)}$, $r_i^{(3)}$ and $\theta_i^{(3)}$ respectively. Note that indexing of only second layer neurons is done depending on the first layer neurons. The neurons in the other layers are indexed independently. The activation values in the fourth and fifth layers are represented by $v_i^{(4)}$ and $v_i^{(5)}$ respectively corresponding to the i^{th} neuron in both layers. The output layer activations are denoted as o_is.

6.4 Computation of Local Line Points

As mentioned before, the second layer contains information about the local line segments in the image. The local information about the possible line segments are extracted by matching suitable templates at each pixel. The templates ($\mathcal{T}_1$, $\cdots$, $\mathcal{T}_{16}$ as mentioned in Chapter 2 (Fig. 2.1)) are designed considering all possible line segments that can appear over a 3×3 neighborhood in a digital grid. It is evident from the nature of the templates that they can be directly implemented in a connectionist framework by properly assigning the weights of the links. The links from the first layer to the second layer represent the template connections. Note that, more than one template will respond at a junction point. Moreover, even if the concerned point is not a junction point, more than one template may produce nonzero response values. For example, if a vertical line segment is present in the image then the $\mathcal{T}_1$ template will produce full response and $\mathcal{T}_5$, $\mathcal{T}_6$, $\mathcal{T}_7$, $\mathcal{T}_8$ templates

will produce partial responses. The situation may become confusing if no template produces full response, but more than one template produce partial responses. As a result, if a single template type is associated with each pixel to represent the possible line direction at that point then the result may become erroneous. It is, therefore, more reasonable to associate more than one template type alongwith their response values with a single pixel (the situation has already been discussed in Chapter 2).

Another problem of using templates to extract the local line segments is that there can be discontinuity in the template responses of the pixels belonging to the same line. Besides, the template type with highest response (dominant template) present at a particular pixel on a line may widely change due to the presence of a small amount of noise. For example one pixel shift of a point in a vertical line segment may cause the template $\mathcal{T}_{13}$ or $\mathcal{T}_{14}$ (instead of $\mathcal{T}_1$) to dominate at that location. Again, note that the sixteen templates correspond to only eight possible directions in [0, 180] degrees because of the fact that the templates determine the directions only on the basis of 3×3 neighborhood. Fig.6.3 shows how these eight directions of a line segment are measured (Chapter 2). On the other hand, the parameter space in the third layer should be able to represent all possible directions of the line structures with better resolution for effective primitive extraction. This problem can be avoided if the directions represented by different templates are iteratively averaged over local neighborhood.

The templates can be efficiently implemented by embedding the template weights into the links from the first layer to second layer. Each group of sixteen neurons in the second layer, connected to an input neuron. correspond to sixteen different template structures. The position of a neuron in the group of sixteen precisely identifies the template represented by that neuron.

The orientation values (directions of local line segments) are associated with the corresponding neurons in the second layer. Each neuron in the second layer stores two different values v and ϕ to represent the line strength and the orientation of the line corresponding to the respective template. The line strengths computed in the second layer are then fed back to the corresponding input neurons. Each input neuron then accepts the maximum line strength coming from the sixteen second layer neurons connected to it. Thus the input neuron stores the maximum line strength of the corresponding pixel. Depending on the local direction of the line, the possible (r, θ) value in the parameter space is computed at each neuron in the

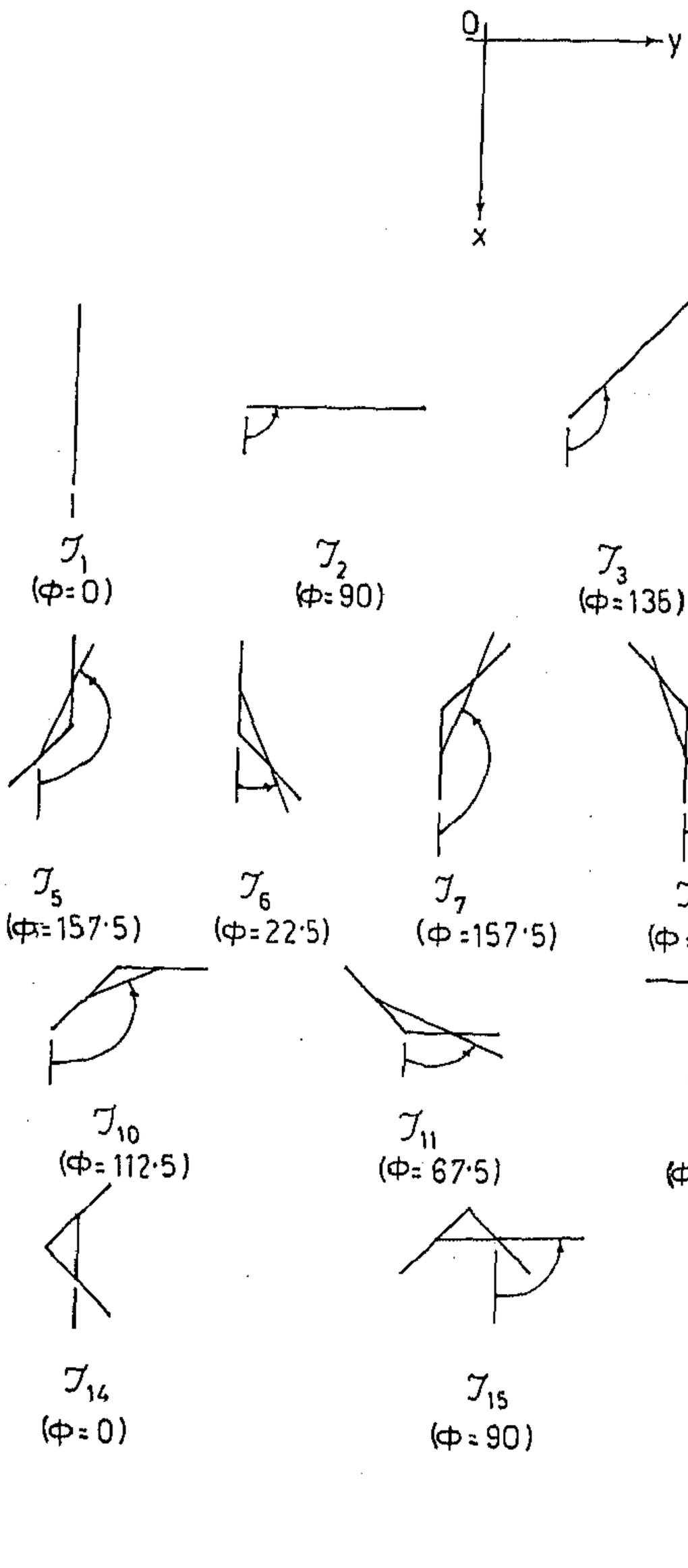
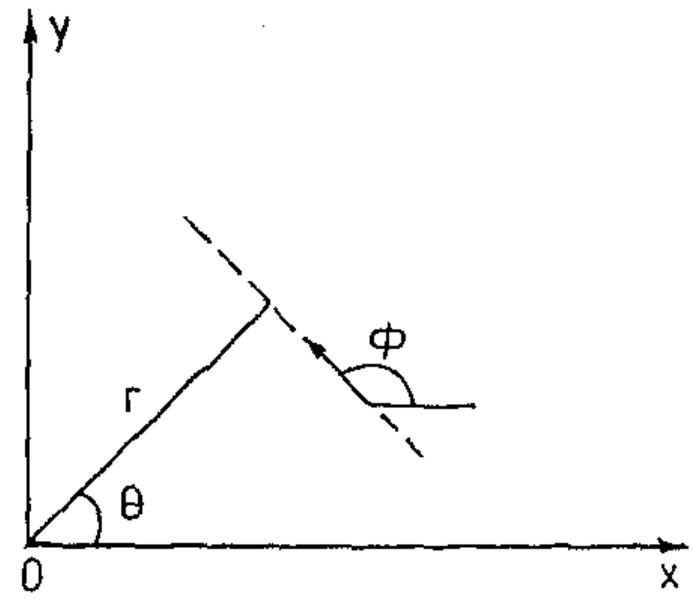
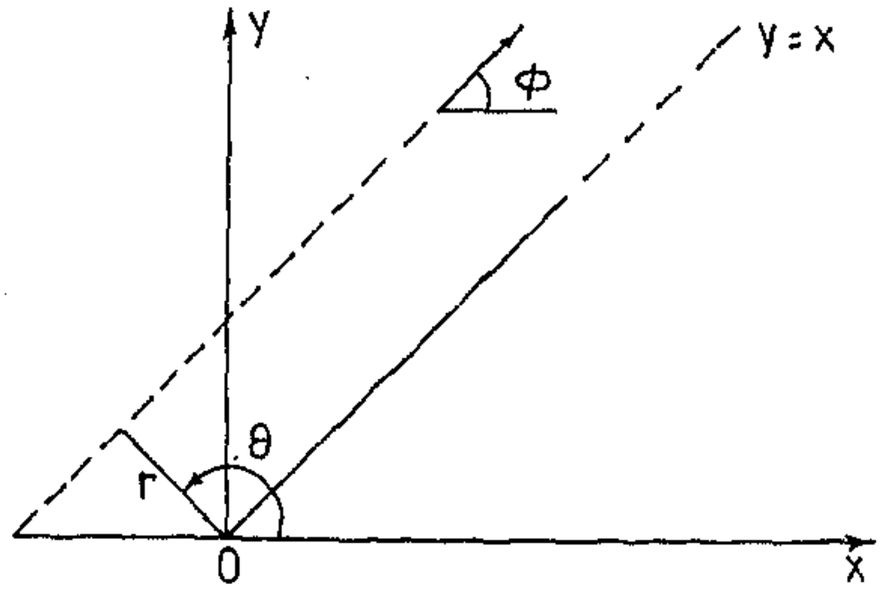


Figure 6.3: Eight possible directions that can be represnted by the digital line segments over 3×3 neighborhood.





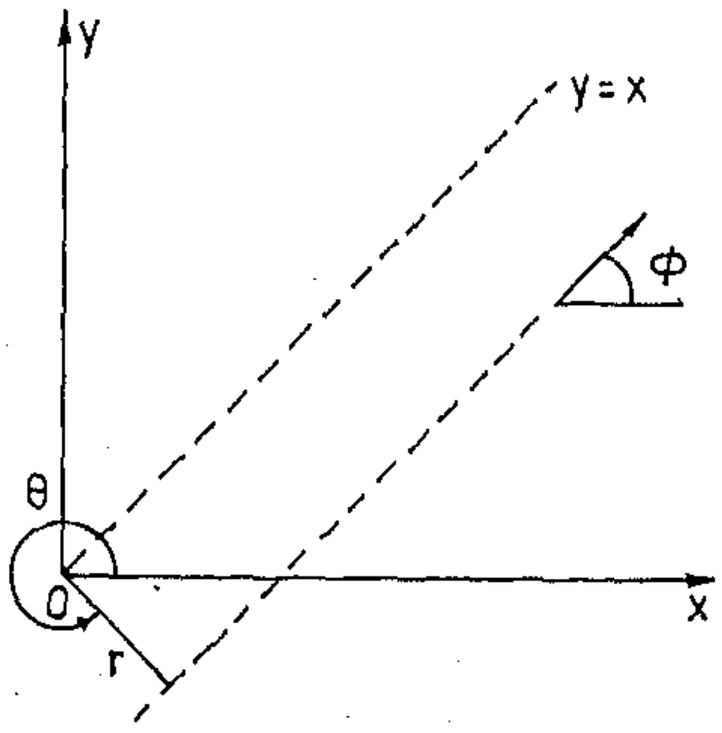


Figure 6.4: Three different cases that can occur for a line segment. In three cases value of θ would have different relations with ϕ.

second layer (Fig.6.4). This is performed with the following equations.

$$\theta = \begin{cases} \phi - 90 & \text{if } \phi \geq 90, \\ \phi + 90 & \text{if } y > x \quad \text{and} \quad \phi < 90, \\ \phi + 270 & \text{otherwise,} \end{cases} \tag{6.4}$$

where (x, y) is the coordinate of the pixel with respect to the image reference frame. Note that, the origin of the image reference frame is fixed at the upper left corner of the image. The value of r is computed once the value of θ has been found out. It is computed by (6.1) which indicates that these calculations involve only local operations and can be implemented with local processors.

The orientation values are averaged according to the following rule.

$$\phi_{i_p}(t+1) = \frac{v_{i_p}^{(2)} \phi_{i_p}(t) + \kappa \sum_{j \neq i} \max_q [v_{j_q}^{(2)} \mathcal{N}(T_{i_p}, T_{j_q})] \phi_{j_s}(t)}{v_{i_p}^{(2)} + \kappa \sum_j \max_q [v_{j_q}^{(2)} \mathcal{N}(T_{i_p}, T_{j_q})]} \tag{6.5}$$

where t represents the number of iterations. κ is a constant which determines the relative importance of activations received from neighborhoods. $\mathcal{N}$ is the neighborhood function which takes values in $\{0, 1\}$ and determines the connections between the nodes in the second layer. T_{i_p} represents the template type corresponding to the i_p^{th} node in the second layer. The template type refers to one of the sixteen templates $(\mathcal{T}_1, \cdots, \mathcal{T}_{16})$ as mentioned before. The use of max() operator takes care of the fact that a second layer node always respond to its strongest neighbor. ϕ_{j_s} is the direction stored in the neighboring second layer neuron with maximum activation value. The neighborhood function is the same as that described in Section 2.3.2 (Equn.(2.3)).

It is to be noted here that the template types are not changed as the orientation values change, i.e., the cooperative connections in the second layer are fixed. After computation of possible direction of the representative line segment in each neuron in the second layer, (r, θ) values are computed according to (6.4) and (6.1). The (r, θ) values are then stored in the second layer.

The size of the input layer is dependent on the size of the input image. For example, if the input image is of size 100×100 then the input layer of the network should have 100×100 neurons. Since the second layer consists of 16 neurons corresponding to each input neuron, the total number of neurons in the second layer is 16×10^4. Each neuron in the second layer represents a possible line direction that can be

present at a pixel. The size of the second layer could have been drastically reduced if four directions corresponding to templates $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ and $\mathcal{T}_4$ were only considered.

6.5 Computation of Global Line Structures

The line structures present in an image are extracted in the third layer. This layer actually aggregates the local line response values (extracted in the second layer) according to their orientations and strengths. The clusters of activations in the parameter space (third layer of the network) are formed after stabilization of the negative self-feedback and the differential support received from the second layer. Each neuron in the second layer is connected to all neurons in the third layer. The weights of the links are fixed.

The output of each third layer neuron also consists of three different values (v, θ, r). The output r and θ values of each neuron in the third layer depend on the position of the neuron. The output r and θ values of each second layer neuron depend on the position of the pixel and the local line direction, which is computed by (6.4) and (6.1).

The updating of the activation values of the neurons in the third layer is derived by minimizing the error of mismatch between the activations of the second layer neurons and the feedback support from the third layer neurons (this is analogous to the error value computed in the X-tron model). The total error between the activations of the second layer neurons and the feedback values is given as

$$E = \frac{1}{2} \sum_i \left[v_i^{(1)} - \max_{i_j} \left(b_{i_j} \frac{v_{i_j}^{(2)}}{v_i^{(1)}} \right) \right]^2 . \tag{6.6}$$

The activation value of a neuron in the first layer is denoted by $v^{(1)}$, that in the second layer is denoted by $v^{(2)}$, and that in the third layer by $v^{(3)}$. The second layer neurons representing the local line response values at the i^{th} pixel are denoted by i_js. The feedback to the i_j^{th} second layer neuron is denoted by b_{i_j}. The amount of feedback to a second layer neuron is dependent on the difference between the resident (r, θ) values of the second and third layer neurons. The difference is modeled by Zadeh's standard π-function [238]. The graphical representation of $\pi(a, b)$ is shown in Fig.6.5. The value of the feedback is given as

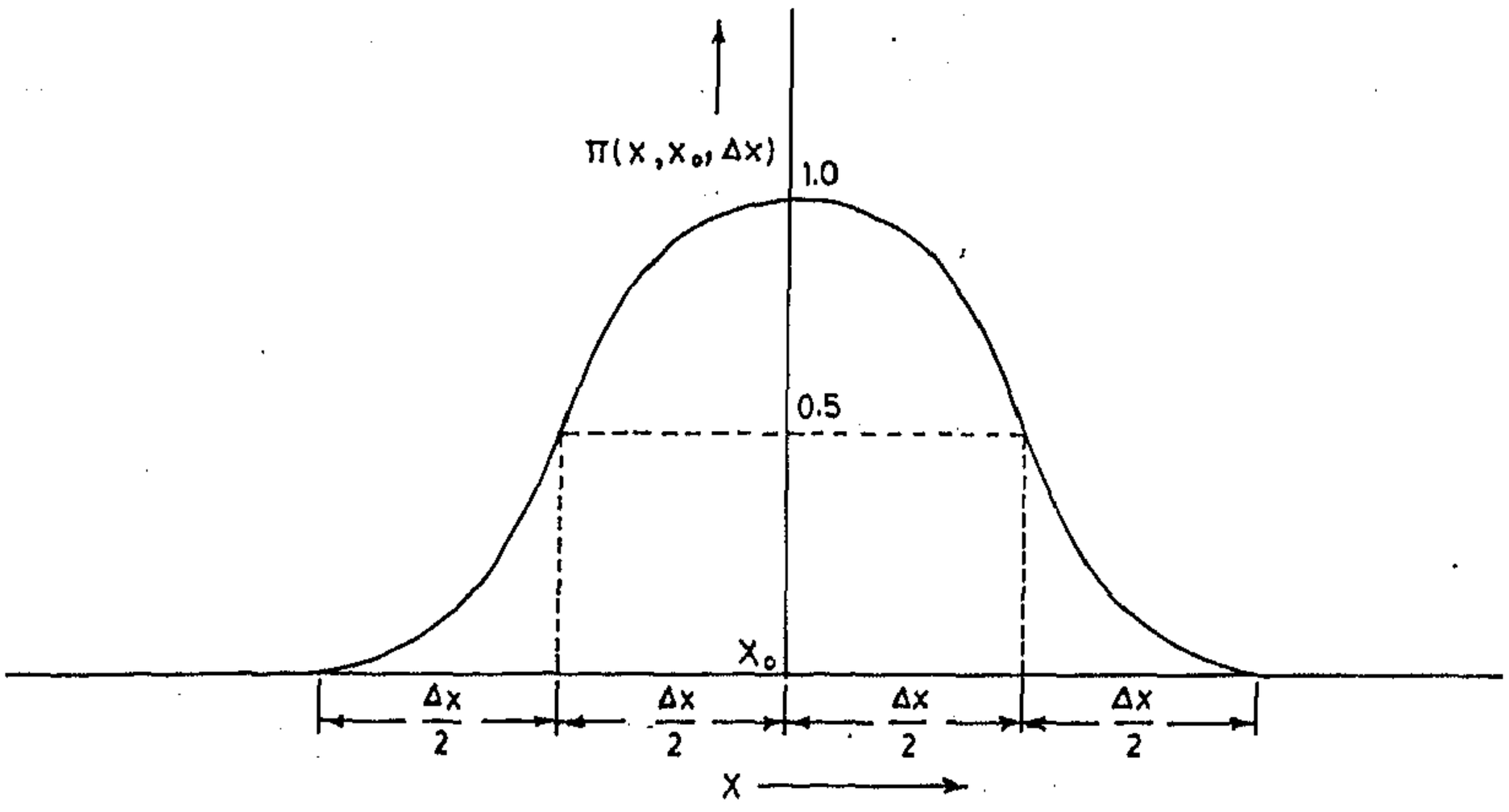


Figure 6.5: The characteristics of a $\pi-$function.

$$b_{ij} = \frac{1}{h} \sum_{k} v_k^{(3)} \pi(\theta_k^{(3)}, \theta_{ij}^{(2)}; \Delta\theta) \pi(r_k^{(3)}, r_{ij}^{(2)}; \Delta r) \tag{6.7}$$

where h is a normalizing constant which determines the average level of activation in the third layer after stabilization. The (r, θ) values stored in the second and third layers are denoted by $(r^{(2)}, \theta^{(2)})$ and $(r^{(3)}, \theta^{(3)})$ respectively. Equation 6.6 indicates the fact that the nodes in the second layer which correspond to the same pixel in the image (i.e., same input node) do not cooperate, rather they compete between themselves. The use of max() operator ensures that only the winner-take-all nodes in the second layer corresponding to each pixel would be able to determine the total error.

The equation 6.6 shows a similar concept of minimizing the mismatch between input and the feedback as used in X-tron. The hidden nodes, here, represent the local guesses about the line directions instead of representing the input-output associations as performed in X-tron. The local guess of a pixel about corresponding line direction is similar to the local guess of a feature about the object to which it should send activation (as in X-tron).

In the error expression (equation 6.6), the feedback value is modulated by the activation value $v_{i_j}^{(2)}$ of the second layer neuron. This is because of the fact that a second layer neuron with a low activation value may receive very high feedback. On the other hand, a neuron with a high activation value may receive a low amount of feedback.. If only the feedback value were considered then the neuron which has low activation value but high feedback would win. This may not be desirable in many cases. To consider effects of both the present activation value and the feedback value, product of these two terms has been used.

In the formulation of error expression, only the mismatch between the activation values present in the input nodes and the feedback support has been considered. The error expression should be modeled in such a way that the redundant activations in the third layer get minimized (this is also performed in X-tron). This can be achieved by adding an extra constraint on the total activation in the third layer. Thus the modified error becomes

$$E = \frac{1}{2}\sum_i \left[v_i^{(1)} - \max_{i_j} \left(b_{i_j} \frac{v_{i_j}^{(2)}}{v_i^{(1)}} \right) \right]^2 + \frac{1}{2} w_s \sum_k (v_k^{(3)})^2 \tag{6.8}$$

where w_s provides the relative effectiveness of the extra constraint. In other words, the error expression can be written as

$$E = \frac{1}{2}\sum_i \left[v_i^{(1)} - \left(b_{i_l} \frac{v_{i_j}^{(2)}}{v_i^{(1)}} \right) \right]^2 + \frac{1}{2} w_s \sum_k (v_k^{(3)})^2 \tag{6.9}$$

where i_l is the winner-take-all node in the second layer corresponding to the i^{th} input node (the competition in the second layer takes place within each group of sixteen nodes).

The changes in the activation values of the third layer neurons are given as

$$\Delta v_k^{(3)} = -\frac{\partial E}{\partial v_k^{(3)}} \Delta t, \tag{6.10}$$

where Δt is a discrete time step. The rule can be derived as

$$\Delta v_k^{(3)} = \gamma \left[\frac{1}{h} \sum_i \left[v_i^{(1)} - b_{i_l} \frac{v_{i_l}^{(2)}}{v_i^{(1)}} \right] \pi(\theta_k^{(3)}, \theta_{i_l}^{(2)}; \Delta\theta) \pi(r_k^{(3)}, r_{i_l}^{(2)}; \Delta r) \frac{v_{i_l}^{(2)}}{v_i} - w_s v_k^{(3)} \right], \tag{6.11}$$

It is therefore seen that w_s acts as the weight of the *negative self-feedback*. Depending on the value of w_s the activation values in the third layer will be determined. If w_s is very high then the proper activations will also be reduced to a great extent. On the other hand, if w_s is small the redundant activations may not be removed.

Equation (6.11) can be interpreted in the following way. Each neuron in the third layer feeds back its activation value to the second layer (which is given by b_{i_l}) through top-down links. A second layer neuron is able to receive the feedback coming from the third layer only when the resident (r, θ) value matches with the (r, θ) value present in the third layer neuron. This matching is modeled in terms of the standard $\pi-$function as shown in (6.7). The way of incorporating the $\pi-$function is described later. Each second layer neuron has two parts. One of them retains the actual activation value $v^{(2)}$. The other computes the modulated activation value ($b_{i_l}\frac{v_{i_l}^{(2)}}{v_i^{(1)}}$). The second part of all second layer nodes (retaining the modulated activation value), connected to the same input node, compete between themselves. After competition is over, only the winner-take-all node becomes able to send activation to the third layer through bottom-up links. The amount of activation from the second layer to the third layer (given as $(v_i^{(1)} - b_{i_l}\frac{v_{i_l}^{(2)}}{v_i^{(1)}})\frac{v_{i_l}^{(2)}}{v_i^{(1)}}$) is dependent on the input activation and modulated feedback activation. The amount of activation is defined as *differential support*. The differential support from a second layer neuron can be mathematically modeled as

$$e_{i_j} = \begin{cases} [v_i^{(1)} - b_{i_j}\frac{v_{i_j}^{(2)}}{v_i^{(1)}}]\frac{v_{i_j}^{(2)}}{v_i^{(1)}} & \text{if } mb_{i_j} > mb_{i_l} \text{ for all } l \neq i \\ 0 & \text{otherwise} \end{cases} \tag{6.12}$$

where,

$$mb_{i_j} = b_{i_j}\left(\frac{v_{i_j}^{(2)}}{v_i^{(1)}}\right).$$

Note that, the differential support has a similar expression as described for X-tron (Equn.(4.7)). However, in X-tron differential support depends only on the difference between input and feedback. Here, on the other hand, feedback, as well as the difference are modulated by the activation value of the corresponding hidden layer node. A neuron in the third layer would be able to receive differential support from a neuron in the second layer, only when the resident (r, θ) values in the second and third layer neurons match. The matching in the (r, θ) values between the second and third layer neurons is also modeled in terms of $\pi-$function

(equation (6.11)). The Δr and $\Delta\theta$ values for matching in the second and third layers are the same (equations (6.7) and (6.11)). The activation values in the third layer are updated according to the difference between the differential supports received from the second layer and the negative self-feedback. It is evident from the derivation of $\Delta v^{(3)}$ (equation (6.10)) that the error value always decreases, i.e., $\Delta E \leq 0$. (The reasoning is exactly the same as that presented in Section 4.3.2). Since the error value is always finite (equation (6.8)), in the limit, $\Delta E \rightarrow 0$ and as a result, $\Delta v_k^{(3)} \rightarrow 0$ for all k, i.e., the system will reach a stable state.

The system needs the standard $\pi-$function to be implemented into the links between second and third layer. The weights of the links are set fixed whenever the second layer neurons get activated and the computation of (r, θ) values is complete. The weights (of both bottom-up and top-down links) are set in the following way.

$$\begin{aligned} w_{i_j k} &= \frac{1}{h}\pi(\theta_k^{(3)}, \theta_{i_j}^{(2)}; \Delta\theta)\pi(r_k^{(3)}, r_{i_j}^{(2)}; \Delta r) \\ z_{k i_j} &= \frac{1}{h}\pi(\theta_k^{(3)}, \theta_{i_j}^{(2)}; \Delta\theta)\pi(r_k^{(3)}, r_{i_j}^{(2)}; \Delta r) \end{aligned}$$

where $w_{i_j k}$ is the weight of the bottom-up link from i_j^{th} second layer node to the k^{th} third layer node, and $z_{k i_j}$ is the weight of the top-down link from the k^{th} third layer node to the i_j^{th} second layer node. In the setting of the weights the (r, θ) values are available at the terminal nodes of the links. (This process of weight setting should not be treated as a learning process.) The process becomes active whenever a new image is presented to the network and the computation of (r, θ) values in the second layer is completed. This is necessary both for learning and recognition. The weights remain fixed so long as the input image is not changed. Whenever a new image is presented to the network the weights are reset and fixed according to the new (r, θ) values, computed in the second layer.

The activation level in the third layer depends on the value of h and self-feedback w_s. In the following discussion we show an empirical relation between the activation level $(v^{(3)})$ in the third layer, h, and w_s. Let an image contains a straight line segment of length l (i.e., l pixels). Let each pixel on the line correspond to a line strength of unity. When the image is mapped onto the connectionist model, all pixels will activate a single third layer neuron under noiseless, ideal condition. Therefore, the change in the activation value of that third layer neuron representing the line segment can be written as

$$\Delta v^{(3)} = \left[\frac{l}{h}(1 - \frac{v^{(3)}}{h}) - w_s v^{(3)}\right] \Delta t. \qquad (6.13)$$

Under stable condition, $\Delta v^{(3)} = 0$. Thus, $v^{(3)}$ will take a value

$$v^{(3)} = \frac{lh}{l + w_s h^2}. \tag{6.14}$$

If $w_s h^2 >> l$ then $v^{(3)} = \frac{l}{w_s h}$, and if $l >> w_s h^2$ then $v^{(3)} = h$. In the first case the information about the length of the line segments are preserved. In other words, with the first condition the activation values in the third layer would correspond to the original accumulator contents (with some scaling), and with the second condition, the activation values in the third layer will have peaks of constant magnitude. This reveals an empirical parametric relation for choosing w_s and h in the connectionist implementation of the Hough transform. Here, the values of h and w_s are chosen in such a way that the information about the line segments in the image is also restored in the activation values of the third layer neurons.

The third layer of the network represents the parameter space of Hough transform. The size of the third layer depends on the chosen resolution of the parameter values used in the Hough transform. In the proposed connectionist framework, the problem of selecting suitable threshold to segment out the peaks in Hough space has been omitted. However, in the present system, we need to choose the parameter values like w_s and h. But since an empirical relation between these parameters has already been derived (equation 6.14), it may provide a better parameter selection criteria. Moreover, the present scheme provides an alternative way to select peaks instead of using a thresholding scheme, although selection of proper thresholds in the standard Hough transform technique may provide comparable outputs.

6.6 Feature Integration

The activation values in the third layer represent the line segments present in the input image structures. The activation values also store the information about the length of the line segments with properly selected parameter values. These linear segments are hierarchically integrated to recognize the shapes of the structures presented to the network. The integration of the segments is performed with a multilayered perceptron model which accepts the activation values from the third layer of the system.

The MLP we considered has two hidden layers (fourth and fifth layers of the system) and one output layer (i.e. the sixth layer of the system). The connections between

two consecutive layers in the MLP model are restricted over local neighborhoods (Fig.6.6). The restriction of the links rids off the network from unnecessary details,

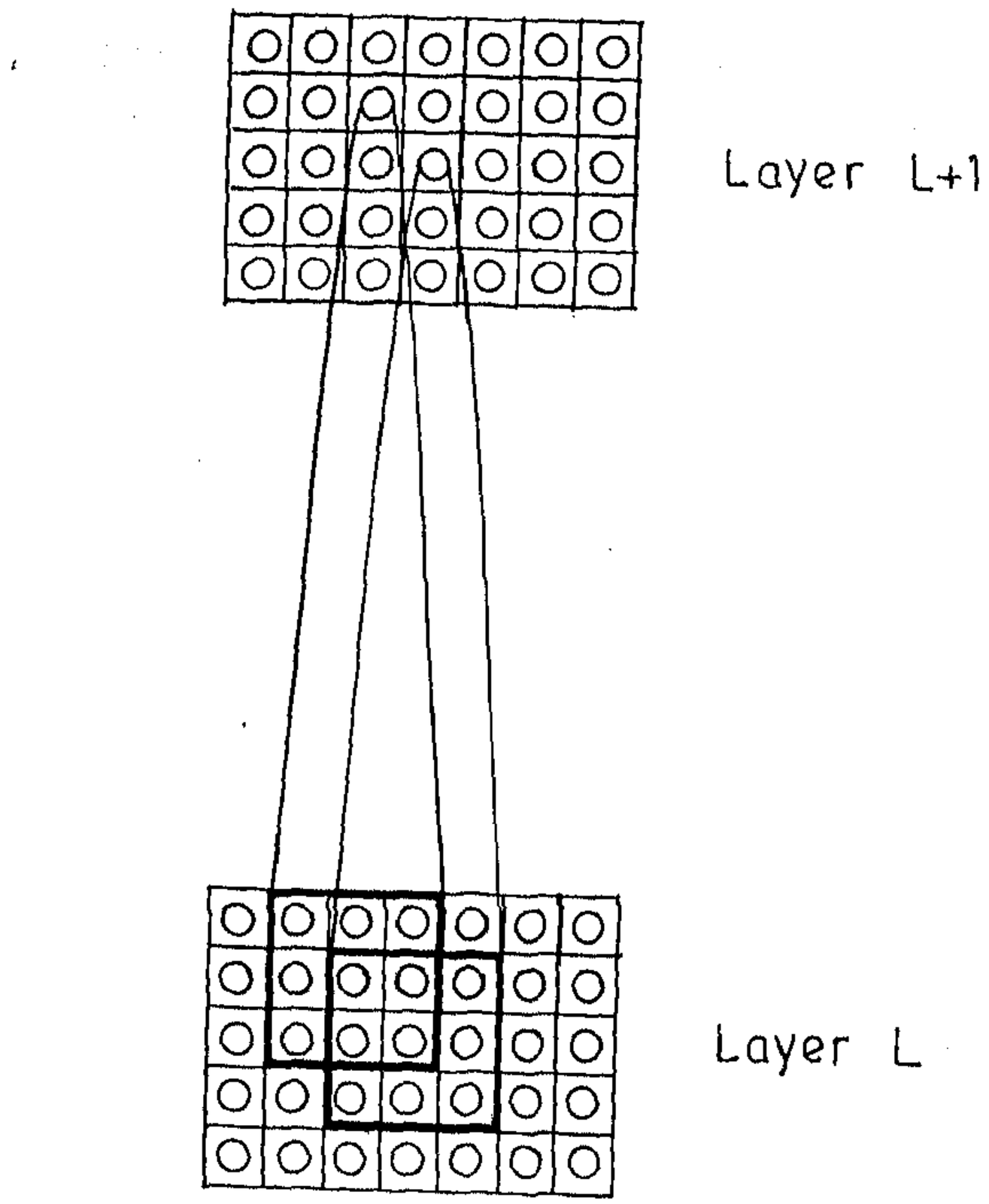


Figure 6.6: The neighborhood connections between two different layers. This is valid for connections from third to fourth and fourth to fifth layers.

and enables it to learn the structures with a greater speed. The size of each hidden layer and the nature of connections with its previous layer are determined according to the desired function of the layer. The way of selecting the size of hidden layers are discussed below.

The fourth layer of the system groups activation values from the third layer over local neighborhoods. The purpose of this layer is to smooth the activation values present in the third layer over local neighborhoods. The size of this layer is exactly

the same as that of the third layer. The size of the third layer is 54×35. The size depends on the chosen resolution of the parameter space (described in the next section). Every node in the fourth layer is connected to a 7×5 neighborhood of the corresponding node in the third layer. The weights of the links are set during the backpropagation learning process.

The fifth layer groups the activation values from the fourth layer over local neighborhoods. The purpose of the fifth layer is to extract out the structural properties within the neighborhood of a primitive. The neighborhoods in the fourth layer are selected in such a way that there exist sufficient overlap between two neighboring regions of activities. In fact, the size of the fifth layer would depend both on the nature of the desired overlap and the size of the neighborhoods. The relationship between the size of the layer, the size of the neighborhood and the nature of overlap is derived below.

The shape of neighborhoods is chosen to be rectangular. Let each neighborhood be of size $m \times n$. Let p_x and p_y be the fraction of overlaps in the two orthogonal directions respectively, which means that $p_x m$ and $p_y n$ neurons (of fourth layer) send activations to two neighboring neurons in the fifth layer. Let the size of the fourth layer be $M \times N$. In that case, each pair of two neighboring neurons in the fifth layer corresponds to a gap of $m(1 - p_x)$ neurons in the fourth layer in one direction and $n(1 - p_y)$ neurons in the other direction. Therefore, the size of the fifth layer (say, $M' \times N'$) is given by

$$M' = \frac{M}{m(1 - p_x)}$$

and

$$N' = \frac{N}{n(1 - p_y)}.$$

The sixth or the output layer finds out the global structure of the character. This layer accepts the activation values from all neurons in the fifth layer.

The rule used for back propagation is given as

$$\Delta w_{ji}^{(l-1)} = \eta \delta_j^{(l)} v_j^{(l-1)} \tag{6.15}$$

where $w_{ji}^{(l-1)}$ stands for the weight of the link connecting the j^{th} node in layer $l-1$ to the i^{th} node in layer l. The δ values are given by

$$\delta_j^{(6)} = (t_j - o_j) f'(u_j^{(6)})$$

where $u_j^{(6)}$ is the total input to the j^{th} node in the sixth or output layer. $f()$ is the transfer function of the nodes. For other layers, the δ values are given by

$$\delta_j^{(l)} = f'(u_j^{(l)}) \sum_k \delta_k^{(l+1)} w_{kj}^{(l)}$$

Since the size of the third layer depends only on the resolution of the Hough space, it is virtually independent of the size of the input image. This indicates that the backpropagation learning rule, taking place from third layer to sixth layer, can be independent of the image size. If the image size is increased, the values w_s or h can be increased accordingly so that the activation levels in the third layer remain unaffected. For example, if image size is doubled in both x- and y- directions, then also the activation level in the third layer remain unchanged if the value of w_s is doubled. This indicates one novelty of this system. Once the system learns the input structure set, the same system can be used to recognize structures of different sizes. The modifications only need to be done in the first and second layers, and this modification does not involve any learning process.

6.7 Results and Analysis

The methodology described in the previous sections has been implemented on handwritten Bengali (one of the major Indian language) character recognition problem. Twelve different Bengali characters are chosen for learning and recognition. The experiment has been performed in two separate phases. In the first phase, only seventeen samples of each character have been taken and the system is trained with these samples. The performance of the system is then tested with noisy versions of the samples. In the second phase of the experiment, we have enlarged our data set. The data set consists of 30 samples for each character and noisy versions of each of them. The data set, thus obtained, is divided into training and test sets. It has been found that with increase in the number of samples, the performance of the system is improved. Ideally, for such kind of tasks dealing with handwritten characters, the number of training samples should be very high. But due to the limited computational facility, we have restricted to small data set and have shown a clear improvement in performance with the increase in number of samples.

In the first part of experiment, characters were written by seventeen different persons. Therefore, the character set contains 17×12, i.e., 204 characters from 12

different categories. Note that, the character set we considered contain linear structures. Moreover, some of them have similar shapes. This particular set is considered in order to establish the discriminating ability of the proposed system even within less variant categories. In other words, the result we get is more meaningful.

The characters are preprocessed before presentation to the network. The graylevel images are segmented to get binary images using graylevel thresholding. The output is then smoothed and cleaned to remove noise. This is performed by growing and shrinking operations over eight-neighborhood [9]. The image is then normalized in size (100×100). The two tone images are then thinned using the thinning algorithm as presented by Rosenfeld and Kak [9]. The thinned versions of some of the characters are shown in Fig. 6.7.

The thinned versions are presented as input to the proposed system. The system transforms the input image into the parameter space (as described in Sections 6.4 and 6.5). In the present case for sake of accuracy and robustness, all sixteen templates have been considered. But in the actual simulation process there is no need to allocate space for all neurons. During simulation, neurons have been allocated only for the nonzero pixels; and for each nonzero pixel the first four prominent directions have been considered. The value of κ (equation 6.5) is chosen as 0.5, this makes the contribution of the concerned pixel and those of its neighbors to be the same during the smoothing process of line directions. The line directions present in the second layer neurons are iterated 10 times.

Considering the structural complexity of Bengali characters, the resolution in the Hough space is chosen as $\delta r = 4$ and $\delta\theta = 5$. Note that the values of Δr and $\Delta\theta$ (equation 6.11) may be different from the resolution, i.e., δr and $\delta\theta$ respectively. However, in the present work they are chosen to be the same. For most of the Indian character set this may provide good results. For English character set, these values would certainly work because the English alphabets are simpler in structure compared to Bengali alphabets. The number of slots the parameter space along the θ axis is determined as 72. Since during the computation of Hough transform the origin of the image coordinate system is considered at one particular corner of the image, the θ values lying in the third quadrant are redundant. As a result, the required number of slots in the Hough space along the θ axis becomes 54. Parameter r represents the normal distance of a line in the image space from the origin of the image coordinate system. The maximum distance can occur along

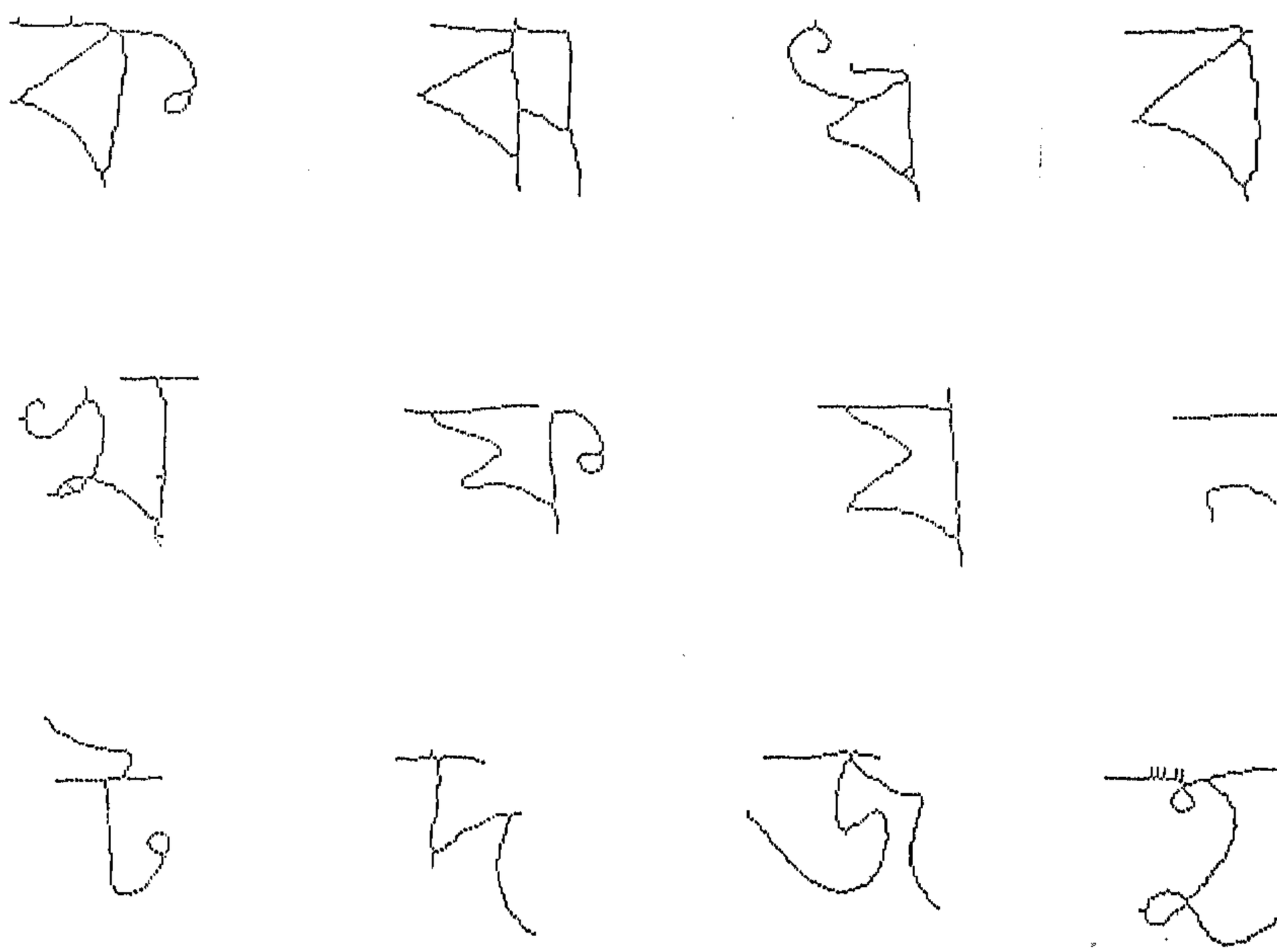


Figure 6.7: A sample character set after thinning.

the diagonal of the image. Therefore, in the present case it can be $\frac{100}{4} \times \sqrt{2}$ which is approximately 35. Thus size of the third layer is chosen as 54×35. It is to be noted here that the size of the third layer can be kept fixed independent of the size of the image.

The values of h and w_s are chosen in such a way that the activation values in the output layer retains information about the lengths of the linear structures. In other words, the condition $w_s h^2 >> l$ is considered. In this image we have $l < 100$ (note that l denotes the number of pixels in a linear structure). The values of h and w_s are selected depending on the maximum length of the image. The value of h is chosen as 1000, i.e., $l/h < 0.1$. Note that, the factor γ/h determines the rate of updating of the states of the nodes in the network. Therefore, if γ/h is too small then the first three layers of the system would take a long time to converge, and on the other hand, if γ/h is large then there can be oscillations in the updating process. The value of γ/h is selected as 0.1 and $w_s h$ is selected as 0.5. Thus, the values of w_s and γ become 5×10^{-4} and 100 respectively. The activation values in the third layer of the system were found to stabilize after few iterations. We have iterated 50 times for each character.

The size of the fourth layer is considered to be of same size as the third layer. With a neighborhood size of 7×5 this gives a overlap of approximately 82% between adjacent neighborhoods in the third layer. In the fifth fourth layer, approximately 50% overlap (between adjacent neighborhoods) is considered with a neighborhood size of 9×7. Therefore, the size of the fifth layer becomes 12×10. We used neighborhoods of size 13×11, which provides slightly more than 50% overlap. Since in the present work only 12 characters are used the output layer consists of 12 nodes. Each node is connected to all neurons in the fifth layer.

The value of η (rate of learning) in the backpropagation rule is varied from a higher value to a smaller one. The learning starts with η equals to 0.5. After each 20 iterations it is decreased by 0.1 until it becomes 0.2. Then η is decreased to 0.1 after 30 iterations and to 0.05 after another 30 iterations. The final tuning of the weights is performed in another 30 iterations with a value of η equal to 0.05. The network is trained with on-line learning, and the change in weights of the links is noted after every epoch. Finally, the normalized change in weights reduces below 0.00005 after 150 iterations. The total processing time for learning the character set is found to be approximately 25 hours (24 hours 59 minutes 38 seconds) on a SPARC 1 workstation (without floating point coprocessor). The training was

performed with the entire character set (204 samples). After the training is over, the same set of characters was presented to the network for recognition, and it was found that the system is able to correctly recognize all characters.

We have also demonstrated the effectiveness of the trained system in identifying distorted structural patterns. In order to generate distorted versions, each pixel can be randomly shifted (with some probability) within its eight neighborhood preserving the connectivity. This process can also be done iteratively to provide severe distortion.

The recognition score of the trained system is shown in Table 1 when different distorted versions (generated with different probabilities and iterations) were given as input. Some of the distorted versions are shown in Fig. 6.8. From Table 6.1, it is clear that the performance of the network gracefully degrades with the level of distortion. Note that the effect of two iterations with a low probability value (0.1) is more severe than that in a single iteration with a higher probability value (0.2). The connectionist system is, therefore, seen to be able to recognize the handwritten characters even when some of their basic linear structures are distorted.

Table 6.1: Recognition score for distorted characters (probability value indicating level of distortion).

iter no	*probability value*		
	0.05	0.1	0.2
1	96.08	93.14	85.78
2	93.63	80.88	72.55

In the second part of the experiment, the data set has been extended to include 30 different samples of each handwritten character. Noise has been injected into these 30 samples (with a noise level of 20% (i.e., with probability 0.2) with two iterations) to generate another 30 samples for each character. We have taken 25 original samples and 30 noisy samples for training the system (i.e., $55 \times 12 = 660$ characters). The set of weight vectors obtained in the first part of the experiment, has been used as a starting point in the second part. We have started with a high value of η equal to 0.5 and it is reduced to 0.1 in 62 epochs. This part of the experiment (training phase) took approximately 85 hours of CPU time on the same SPARC workstation.

After only 62 epochs, the network was able to recognize 98% of the training samples

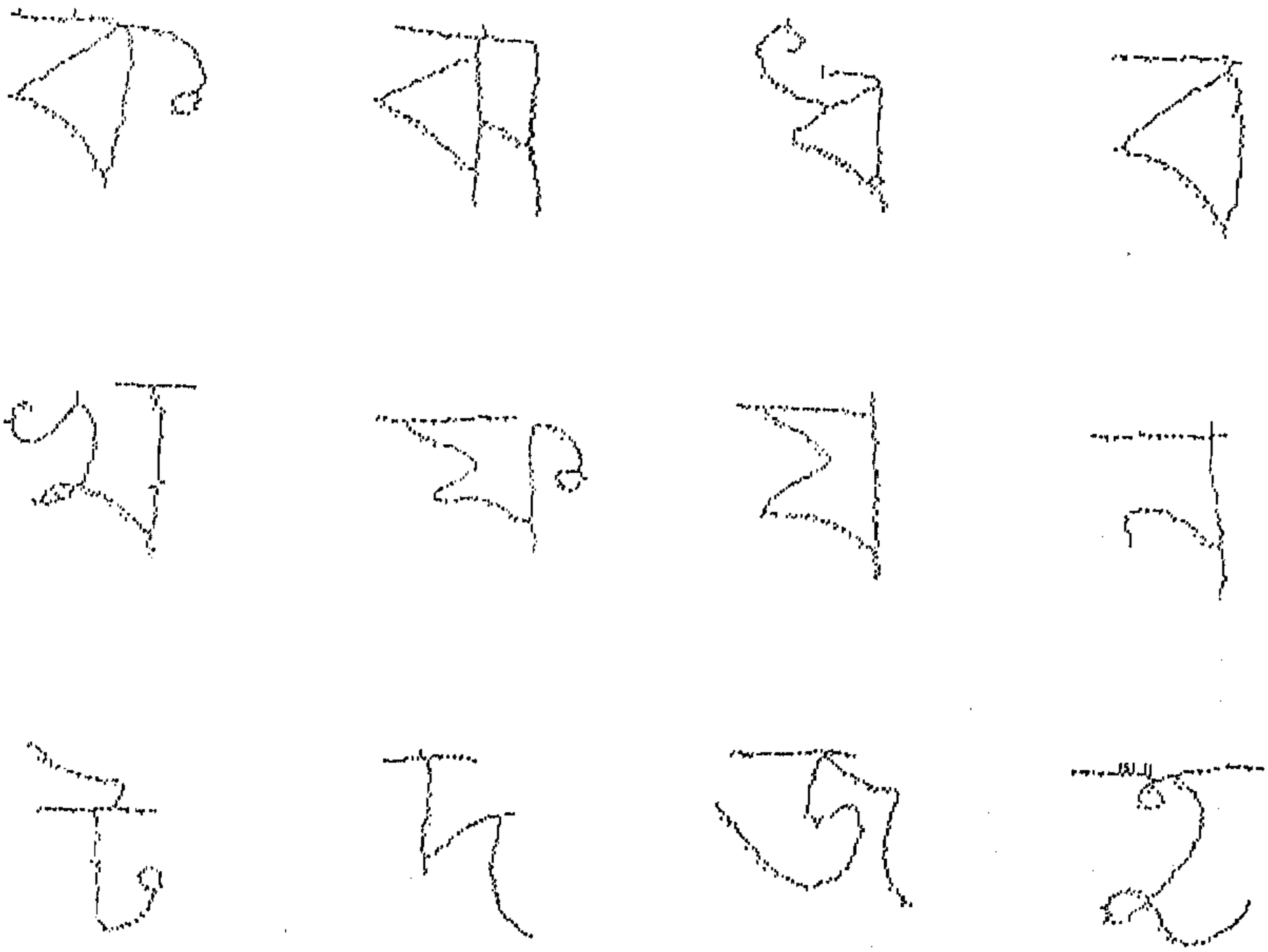


Figure 6.8: Distorted version of the sample character set.

correctly. However, the generalization capability of the system can be checked with data sets that are not used for training. To show the effectiveness of the network, we did the following : we mentioned earlier that our original data set has 30 samples of each character, of which 25 are used for training. We tested the network with the remaining 5 samples of each character (i.e., a total of $5 \times 12 = 60$ characters) and recognition score is found to be 100%. We then generated a new data set by adding 20% noise to each of 30 samples (i.e., $30 \times 12 = 360$ characters) of each character with one iteration. (Please note that for the training set also we added 20% noise, but it was repeated twice). The recognition score on these 360 noisy characters is found to be nearly 86%. The investigation indicates that the performance of the proposed system can be enhanced with larger training data set and more learning (more epochs).

6.8 Conclusions and Discussion

A connectionist system for learning and recognition of structural patterns has been presented here. This includes a scheme for robust primitive extraction using the principles of X-tron (Chapter 4) and line linking (Chapter 2), and integration of features using multilayered perceptron. The first three layers of the system exploit the iterative verification scheme as described in Chapter 4. The line response values and their directions are smoothed considering the neighborhood information, as described in Chapter 2.

The merits of the proposed system lie in the fact that it can select peaks automatically in the Hough space without using any threshold selection scheme, and can perform selective integration of the features during learning. Another advantage of this system is that it is able to learn structures independent of their size. If the sizes of the structures are increased, the parameters in the third layer of the system can be adjusted to get the same activation levels in the third layer. Therefore the same MLP model (i.e., from third to sixth layer of the system) can be used for further learning. Since the system approximates curves with line segments using Hough transform, it seems to be efficient in handling broken line segments also.

To demonstrate the effectiveness of the system, we considered, as an example, the problem of recognizing handwritten Bengali characters of similar shapes. It has been found that the performance gracefully degrades with the distortion level in

input. The methodology can also be applied for recognition of other structural patterns such as industrial objects. In that case, skeletonized images of the industrial parts are to be presented with a suitable frame of reference.

Chapter 7

PsyCOP : Object Identification and Localization

7.1 Introduction

In this chapter, we are going to present a new connectionist system (PsyCOP) which is built using the principle of X-tron, for learning and simultaneous recognition of multiple flat industrial objects (like hammer, spanner, plier etc.) with orientation and shift invariance [217], [218]. Note that, in the development of PsyCOP, both the problems of identification and localization of objects are considered.

Note that, some of the connectionist models for object recognition (mentioned in Section 1.3.3) consider pose identification (localization) problem, but at the same time they use multiple copies of the same entity at each location [188], [189]. Some models [186], [187] consider pose identification with the help of selective attention mechanism, but do not consider the structural relationships between the features and objects (as necessary for orientation and shift invariant industrial parts recognition).

Although the psychological and neurological findings indicate the possibility of existence of two interconnected pathways for entity and pose identification [202], [203], the existing connectionist models do not incorporate this fact in the strategy for recognition. It is to be noted that the necessary behavioral experiment in

support of this separate processing zones has also been provided in the literature of psychology [239].

There are also various studies on the selective attention mechanism for high level vision [240], [241], [242], [243] in the literature of psychology. The theory of selective attention states that different parts in a scene are attended at different times depending on the visual cues present in the scene. The mechanism of selective attention involves the feedback through top-down paths which gates the receptive field of lower level neurons. The selective gating of the lower level signals takes place with the help of attention director. There exist two different theories of visual attention, namely, early and late selection theories. In early selection theory [240], [241], attention control occurs before recognition of the object (e.g. color, length etc. of an object). On the other hand, the late selection theory [243] indicates that the attention control occurs after recognition of the object (e.g., reading alphanumeric texts etc.). Bundsen [244] presented a unified theory of selective attention mechanism where both early and late selection processes occur.

Psychological studies reveal some properties of the cognitive behavior of the animals, while the classical computational techniques try to extract out objects from digital images (a brief survey has been presented in Section 1.2.2). One of the mostly used computational technique to generate initial hypotheses is generalized Hough transform [245], [93], where the boundary points of an object are transformed to the parameter space. In this technique the object can be found even when some portion of the data is missing. One way to take the advantages of psychological findings and the advantages of using the transformations (Generalized Hough transformation) is to use the connectionist framework of computation.

The objective of investigation in this chapter is not to explain the psychological behavior [202], [203] of cognition, but to develop an efficient connectionist system for object recognition by integrating cognitive findings with the generalized Hough transform technique. PsyCOP has been developed by considering the psychological fact that the identification and pose estimation of objects occur in two different regions in visual cortex. It has been found that usage of such a phenomenon in the design of the connectionist system results in reduced number of neurons. (The system is named as PsyCOP which stands for a Psychologically motivated Connectionist system for Object Perception).

Some principles of connectionist computing presented by Feldman [26], [27], [200]

are applied in the implementation of PsyCOP. The idea of incorporating spatial information in the visual domain with less number of neurons used in [27] has also been exploited in the design of connectionist architecture. PsyCOP uses the same principle as X-tron for recognizing mixed objects using iterative verification scheme. It uses the same learning rules, as presented in Section 4.4.2, to capture the importance values of the features with respect to the objects depending on the environment. PsyCOP is also able to learn the structural relationships between the features and the objects. Several new connectionist modules are also designed for the effective design of PsyCOP.

The rest of this chapter is organized as follows. The strategy for recognition of flat rigid industrial objects using two separate processing zones is described in Section 7.2. The architecture of the system is described in Section 7.3. Section 7.4 describes the overall mechanism for object identification and localization alongwith some new connectionist modules that have been introduced in the architectural design. A two-stage learning paradigm is described in Section 7.5 for capturing the structural relationships between the features and the objects, and for capturing the degree of importance of the features with respect to the objects. Section 7.6 presents the way of implementation of the system and the recognition performance for real-life objects (including overlapped industrial parts). A theoretical analysis of the robustness of this method is presented in Section 7.7. Finally, some concluding remarks are provided in Section 7.8.

7.2 Strategy for Recognition

In the proposed methodology, objects are localized by indicating their position and orientation. The position of an object denotes the position of the object centroid, and the orientation is the orientation of the principal axis of the object with respect to some standard reference frame. The features are also attributed by the feature type, feature position and orientation. Here, we have used polygonal approximations of 2D objects and the corners are used as the features. Note that many other features could have been used; however, we have restricted to corner features only. The position and orientation of a corner feature is represented by the position of the corner and orientation of the angular bisector of the corner with respect to some standard reference frame. The reference frame is fixed for all

features detected in the scene and the model produces output showing the position and orientation of an object with respect to the same reference frame.

It is to be noted in this respect that we have considered rigid objects only. For any rigid object, so far as the scale remains unchanged, the relative position and orientation of an object with respect to a constituent feature remains unchanged. This has been illustrated in Fig.7.1. Therefore, if the values of (r, θ, ϕ) are stored

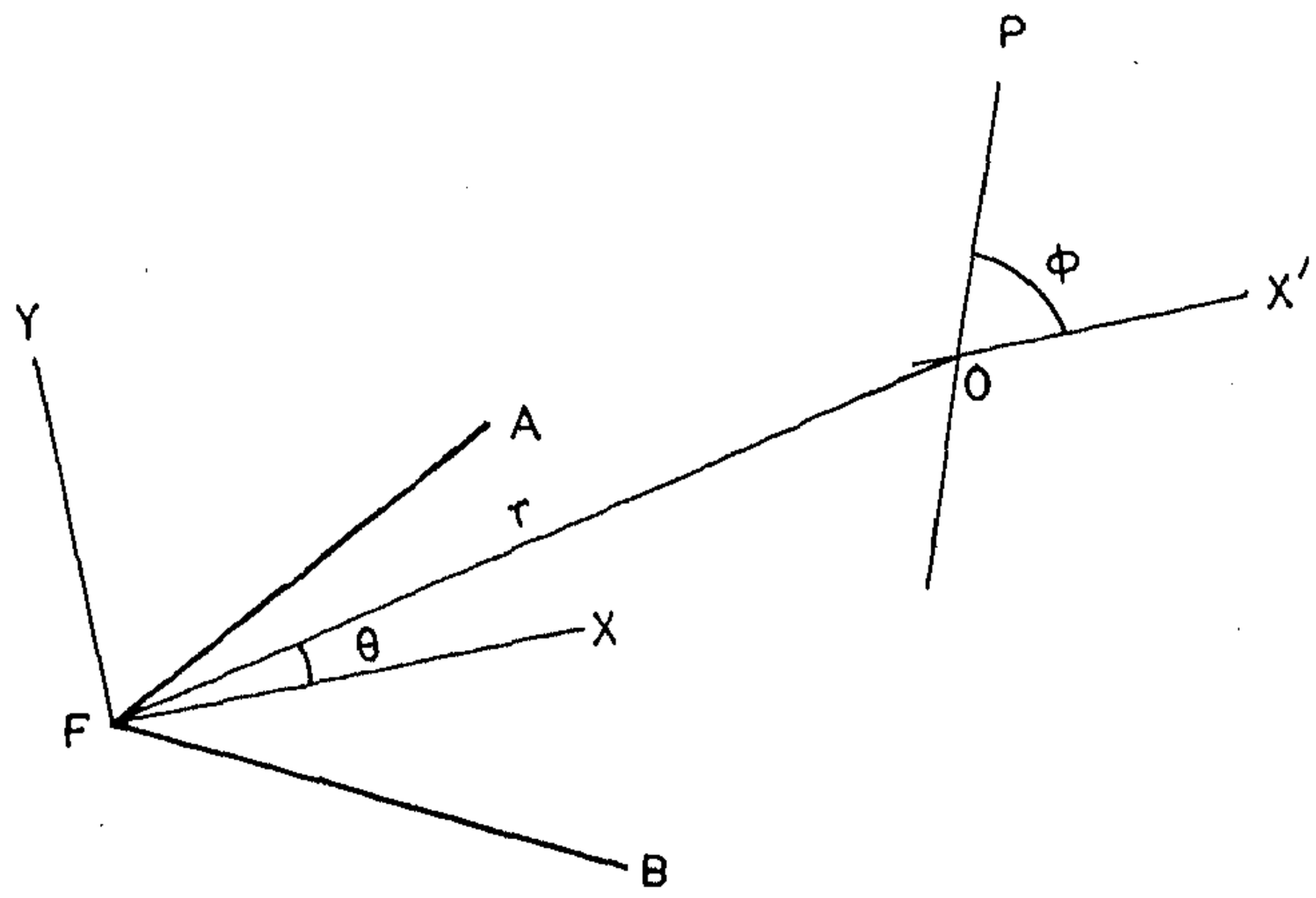


Figure 7.1: **F** and **O** are the positions of the feature and object respectively. The corner has an angle **AFB**. **FX** is the angular bisector. The feature reference frame is **(FX, FY)**, where **FY** is normal to **FX**. The principal axis of the object is **OP**. (r, θ, ϕ) denotes the location of the object with respect to the feature reference frame.

for a particular feature-object combination, then the corresponding feature would be able to predict the position and orientation of the corresponding object in the scene [1]. This is very similar to the idea of generalized Hough transform (GHT) technique [93].

[1] r, θ, and ϕ values are used in this chapter to specify the relative position and orientation of an object with respect to a feature, although the same notations have been used for different purpose in the previous chapter

In GHT, the entire image space is quantized into $m \times n$ (say) slots. The orientation values are also quantized into p (say) slots. An accumulator array $Ac[N][m][n][p]$ is maintained where N is the total number of objects in the database. For every feature in the scene (originally, boundary pixels were considered as the features), the positions and orientations of the objects to which the feature belongs are calculated and the corresponding accumulator values are incremented. For example, let a feature f_1 belong to three objects o_1, o_2 and o_3 and the corresponding positions and orientations are (m_1, n_1, p_1), (m_2, n_2, p_2) and (m_3, n_3, p_3) respectively in the quantized slots. In that case, the accumulator values, $Ac[1][m_1][n_1][p_1]$, $Ac[2][m_2][n_2][p_2]$ and $Ac[3][m_3][n_3][p_3]$ are incremented by unity. After considering all features in the scene, the peaks in the accumulator space are found out by some thresholding mechanism and these peaks essentially represent the identities and locations of the objects present in the scene.

Even if we consider rigid objects with fixed scale, GHT has a couple of disadvantages. First of all, the peak selection process has to be accurate. A high value of threshold may result in removal of peaks due to the genuine objects and a low value of threshold may cause some spurious peaks to remain. Moreover, the required threshold to properly segment the peaks may be different in different regions in the accumulator space. Second, the size of the accumulator space drastically increases with the increase in the number of objects in the model-base. Moreover, the importance of different features with respect to the objects are not considered in GHT. Using connectionist framework of computation, such kind of problems can be dealt with in a better way. The design of such a connectionist system is also motivated by the psychological findings.

In the proposed system, two different channels (block diagram is shown in Fig.7.4) have been used to represent the object identities and their locations. "An entity or an object has appeared at a particular location" : this can be represented in the form of a coupling between two nodes, one representing the object identity and the other representing the location. Let us call the cells used to represent the 'what it is' part as A-cells, and the cells used to represent the 'where it is' as B-cells.

Two special type of connections are used to build the structured connectionist model. These are

A : modifier of the links,

B : sigma-pi connections of the links.

The concept of modifier of the links has been presented in [27]. Two types of modifiers, namely, stimulating and inhibitory modifiers, have been used in the present network (as shown in Fig.7.2(a)). The link emanating from node C stimulates (in Fig.7.2(a)) or deactivates (in Fig.7.2(b)) the link connecting the nodes A and B. In

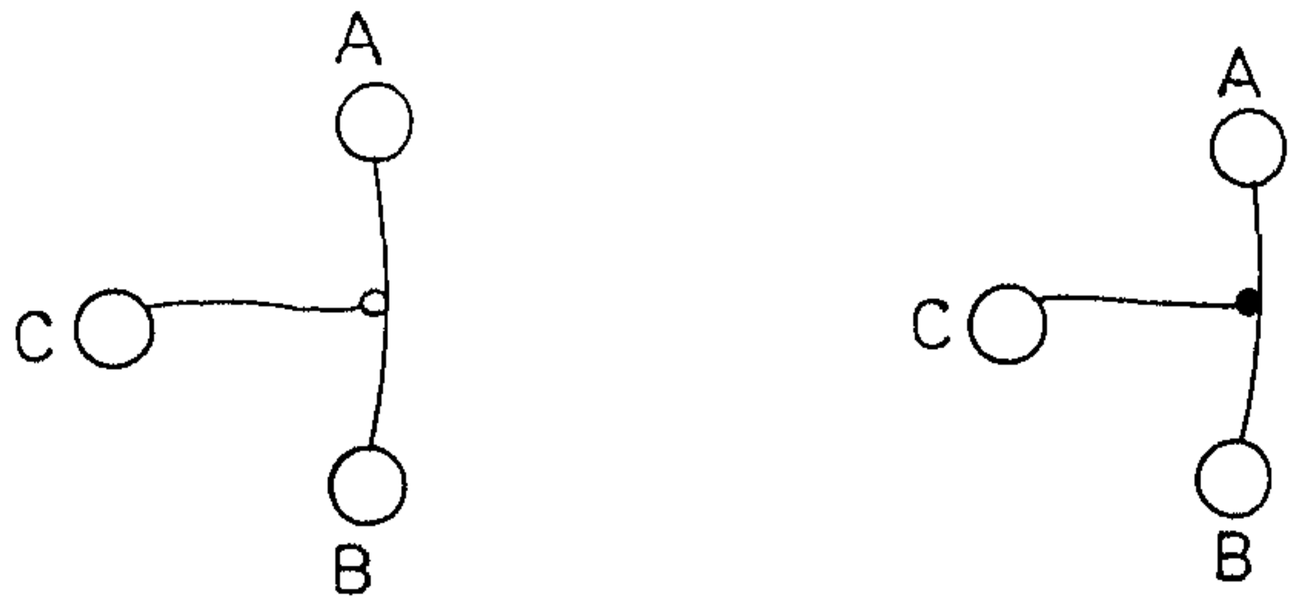


Figure 7.2: (a) A stimulating modifier of the link connecting the nodes A and B, (b) A deactivating or inhibitory modifier of the link connecting the nodes A and B.

the first case, A is able to send a signal to B (or otherwise, i.e., B sends to A) only when the connecting link is stimulated by some positive signal from C through the modifier. In the second case, A is not able to send any signal to B (and also B is not able to send to A) so long as the connecting link is deactivated or inhibited by some positive signal from C through the inhibitory modifier, even if the connecting link is stimulated by some other modifier.

The concept of pi-connection was introduced in [28]. In this case, a node receives the product of more than one signals. For example, a node is connected to a group of links having weights $(w_{11},\ldots,w_{1m}),\ldots,(w_{n1},\ldots,w_{nm})$ such that every n-tuple of links have pi-connection. The links are connected to nodes which hold activation values given as $(x_{11},\ldots,x_{1m}),\ldots,(x_{n1},\ldots,x_{nm})$. In that case the total input received by the node (u) will be

$$u = \sum_{i=1}^{n}\prod_{j=1}^{m} w_{ij}x_{ij}$$

Fig.7.3 shows a typical case that we have used in the proposed model. Here, the

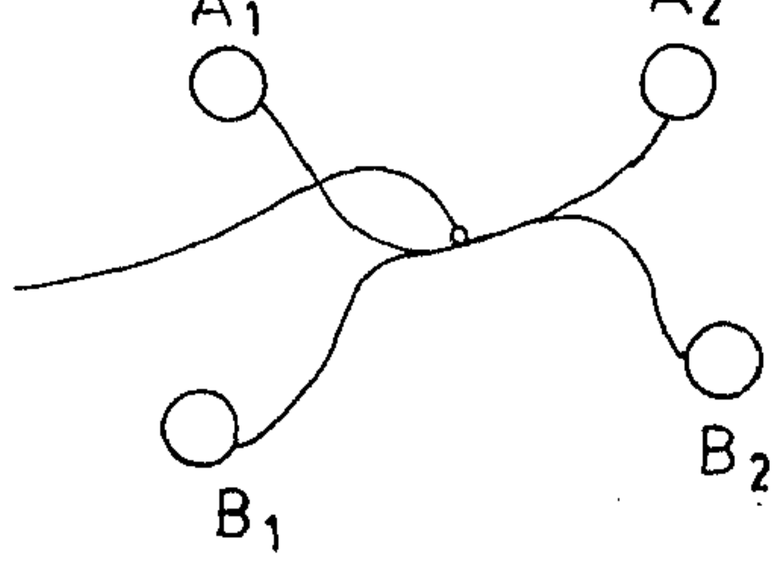


Figure 7.3: The sigma-pi connection between two links emanating from the nodes A1 and B1. The link branches out to two other nodes A2 and B2.

links emanating from two nodes A1 and B1 get conjunctively or pi-connected and then branch out to two other nodes A2 and B2. The conjunctive connection also get stimulated by another signal through the modifier. The nodes A2 and B2 receives the weighted product of the outputs of A1 and B1 when the conjunctive connection is stimulated. The sigma-pi and the modifier connections play important roles in separating the two channels to represent the 'what it is' and 'where it is'.

The model employs the technique of iterative hypotheses verification which is in principle, the same as that used in X-tron (described in Chapter 4). The input layer of the network consists of a set of neurons to represent the entire set of features that can appear in the objects to be recognized. For example, if we consider the corners as the features of the polygonal objects, then the entire range of the angles of the corners are divided into a number of slots, and each slot represents a particular feature. The relative locations of the objects with respect to the constituent features $(r, \theta, \phi$ values) are stored in the links.

Whenever a set of features and their positions are specified to the system, each feature instantiates a set of possible candidate objects and their respective locations depending on the (r, θ, ϕ) values stored in the links. All these activations are

represented in the form of stable coalitions where an activated B-cell representing the location gets connected through an activated link to an A-cell representing the entity (either feature or object). The activations corresponding to the object instantiation are grouped by the system to produce initial hypotheses.

After the formation of the initial hypotheses of the object categories, the iterative process of verification takes place when each candidate coalition feeds back its activation to the feature level. If the feature activation is less than the feedback activation then the activation level of the candidate object category is decreased. On the other hand, if the feature activation level is greater than the feedback then the activation level of the candidate object category is increased. The system stabilizes when a match between the input and the feedback activation is achieved. Let us now describe the architecture of the system in detail.

7.3 Structure of the Model

The system consists of three different layers, namely, input layer, output layer, and hidden layer (Fig.7.4). Input layer corresponds to the features, output layer corresponds to the objects, and the hidden layer corresponds to the feature-object associations. (Note that, feature-object association has the same meaning as that used in X-tron. However, here information about location is also considered.) The activation of an A-cell in each layer is either 1 or 0 representing if the corresponding entity is present or absent, whereas the activation level of a B-cell represents the confidence about the presence of some entity at the corresponding location. The A-cells are arranged in linear arrays representing the maximum possible number of features in the input layer, the maximum possible number of objects that the system can learn in the output layer, and the maximum possible number of feature object associations that have occurred in the hidden layer. The B-cells are arranged in the form of a 3D array (i.e., columns of B-cells are arranged over 2D grid), where each cell specifies a particular position and orientation of an entity (feature or object or feature-object association).

The network has three different types of links (Fig.7.5). Type 1 links connect the A-cells of one layer with the A-cells of another layer. Type 2 links connect the B-cells of one layer to the B-cells of another layer. The A-cells and the B-cells of the same layer are connected by the type 3 links. For example, the hidden layer

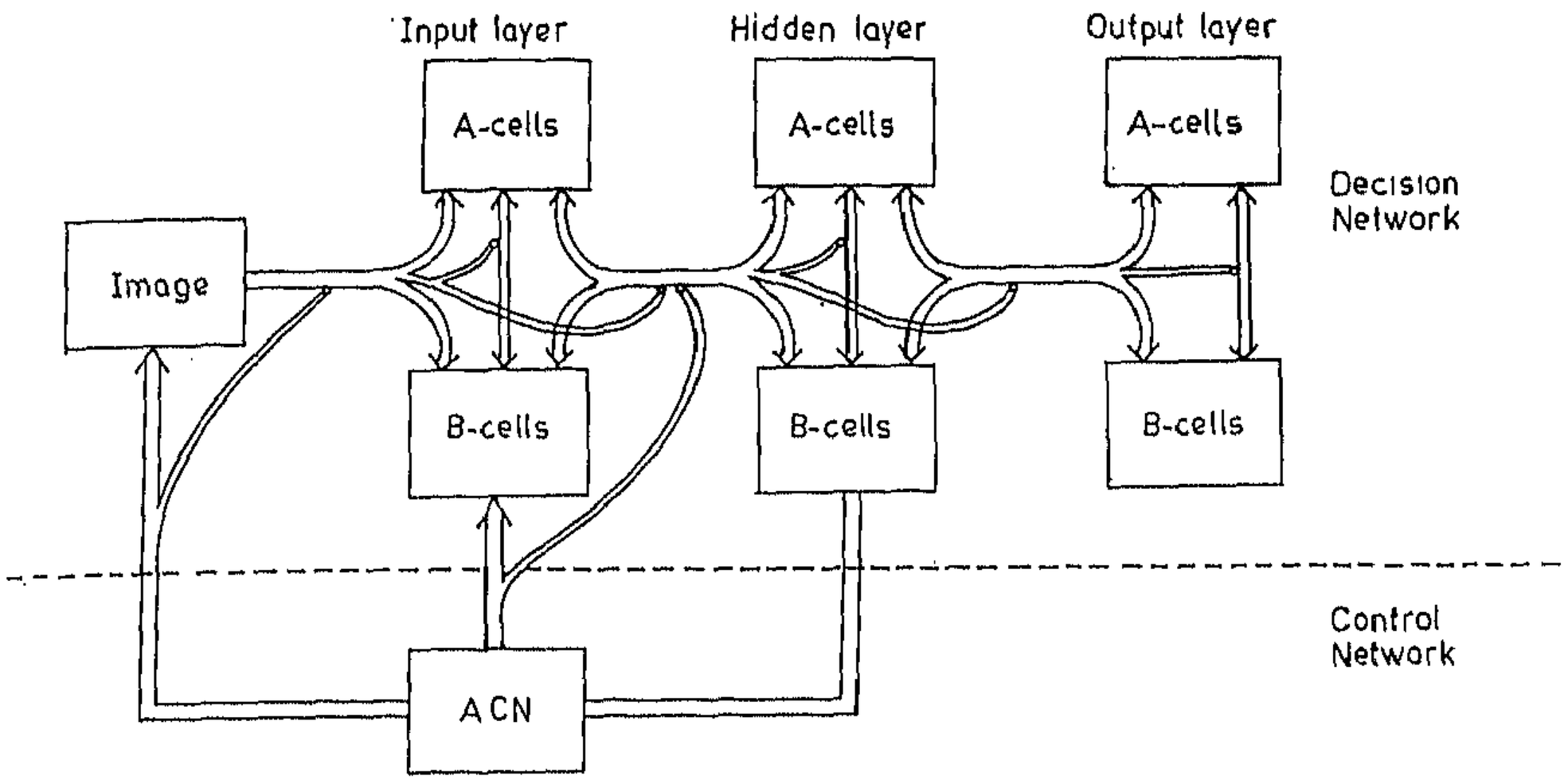


Figure 7.4: Block diagram of PsyCOP. The decision network determines the location and identity of an object while the control network controls the selective attention mechanism.

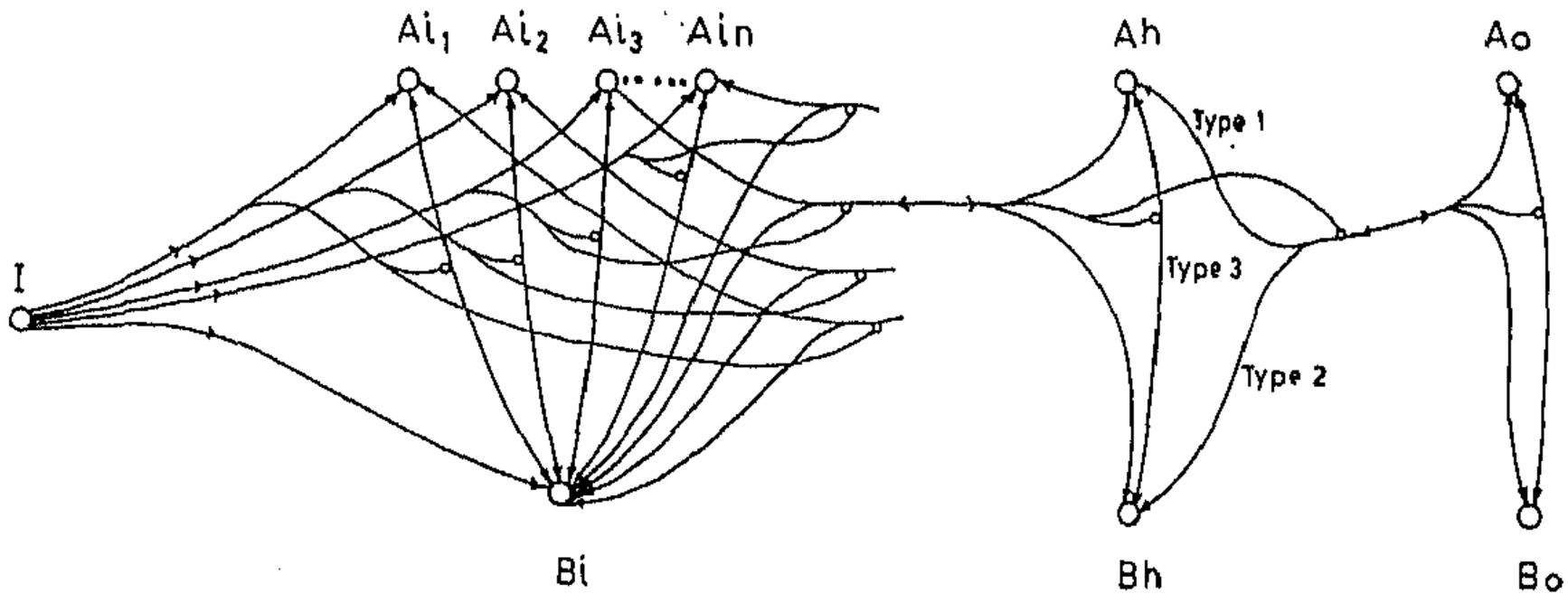


Figure 7.5: Connections between the A and B cells of the input, hidden and output layers. 'I' represents a feature from which the input nodes are activated.

A-cells representing the feature-object associations are connected by type 3 links with the hidden layer B-cells.

The type 3 links help in the formation of couplings between A and B cells. If a B-cell is activated then the activated A-cells that can be accessed from that B-cell through type 3 links represent the entities present at the location corresponding to the B-cell. Similarly, if an A-cell is activated, the activated B-cells that can be accessed through type 3 links, represent the locations where the entity, corresponding to the A-cell, is present. But there will be false alarming in the output whenever more than one object is present in the scene. For example, let two objects O1 and O2 be present in the scene at P1(x_1, y_1, ξ_1)[2] and P2(x_2, y_2, ξ_2) respectively. In that case the A-cell corresponding to O1 would be connected to both the B-cells corresponding to P1 and P2. Similarly, the A-cell for O2 would be connected to both the B-cells corresponding to P1 and P2. As a result, the network would infer that O1 is present at P1 and P2, and O2 is also present at both the locations. Consequently there will be a confusion in the output.

To prevent such a situation, type 3 links are selectively stimulated by using modifiers. An A-cell will be able to access B-cells through only those type 3 links which are stimulated by the modifiers. The type 3 links are modified by the conjunction of type 1 and type 2 links coming from the lower layer. A and B cells in each layer also get activated by the conjunctive or pi-connection from the lower layers.

The type 1 links, emanating from the input layer A-cells, and the type 2 links, emanating from the input layer B-cells, get conjunctively connected to form the type 12 links (henceforth, the conjunctive or pi-connection between type 1 and type 2 links will be denoted by type 12 links). The type 12 links then branch out and enter the hidden layer A-cells and hidden layer B-cells as input, and connect to the hidden layer type 3 links as modifiers. Another branch of the type 12 link emanating from the input layer goes to modify the type 12 link emanating from the hidden layer. Similarly, the hidden layer type 12 links branch out to the output layer A-cells, B-cells and output layer type 3 links.

Between the hidden and output layers, there are two kinds of type 12 links. The bottom-up type 12 links carry the activation values of the hidden nodes to the output nodes, while the top-down type 12 links carry the activation values of the output nodes down to the hidden nodes. Type 1 links between hidden and output

[2]The symbol ξ is used to represent angle of orientation

layer store the relative importance of the feature-object associations, while the type 2 links have some fixed weights. The type 1 links from the input layer to the hidden layer store the transformational offsets (r, θ, and ϕ values), whereas the type 2 links between input and hidden layer have some information regarding physical offsets. The details have been discussed in the next section.

Corresponding to each input A-cell (representing a feature), the hidden layer contains a number of hidden A-cells (which is equal to the number of objects to which the feature belongs, and each hidden A-cell represents a feature-object pair). Whenever, the feature set corresponding to an object is mapped onto the input layer, each input coalition tries to activate hidden cells depending on (r, θ, ϕ) values stored in the links. As a result, hidden B-cells are activated within a cluster. To map the feature set onto input layer and to form the cluster of activations, a sequential scanning mechanism is employed which has much similarity to the selective attention mechanism described in the literature of psychology. The selective attention mechanism is realized with the help of a special network, namely, *attention control network* (ACN). The attention control network is coupled with the type 2 links from the input layer B-cells to the hidden layer B-cells. ACN also uses the mechanism of modifiers to selectively stimulate the type 2 links emanating from the zone of attention. The detailed mechanism of selective attention will be discussed in the next section.

7.4 Object Recognition Mechanism

Let us now describe the overall process of recognition with the proposed network. The features (e.g., corners) extracted from the graylevel image are mapped onto the input layer sequentially with the help of attention control network (ACN). When the image (containing the input features) is scanned, the features in the zone of attention activate the input A-cells and the B-cells depending on the type, position and orientation of the features. It is to be mentioned here that the feature extraction process should take care of the fact that at most one feature can appear in a region equal to the input grid size. The size of the zone of attention depends on the sparseness of the features. It is chosen to be greater than or approximately equal to the input grid size.

The input A-cells have special type of transfer functions which produce maximum

output (unity) for certain range of input activation and zero for rest of the input. Note that the activation level of A-cells represent the presence or absence of some entity and not the confidence level about its presence. The input B-cells have two parts : one of them holds the activation value representing the confidence about the presence of a feature and the other has a special type of transfer function (radial basis function [29] tuned at certain orientation). One of the B-cells at the input grid location of a feature gets maximally activated depending on the orientation of that feature. A local competition mechanism within a column of B-cells helps the maximally activated cell to become winner and others lose their activation.

The type 3 links and the conjunctive connections of type 1 and type 2 links, connected to the input A and B cells also get stimulated (guided by the ACN). As a result the activated A and B cells form coalitions through the stimulated type 3 links and also become able to send activation to the hidden layer through the stimulated type 12 links.

Although the A and B cells get selectively activated from the input, the connecting type 3 and conjunctive type 12 links seem to remain stimulated which may cause formation of false coalitions. To prevent this, the links have a special property where if either of the cells of a stimulated link remains inactive for sometime, the link loses the signal carrying capability unless it is further stimulated by some signal. The time over which the stimulated links lose their signal carrying capability is set in such a way that the link gets deactivated before the attention control network switches to the next zone of attention. (Let us term this property as attenuable LTM).

The hidden layer A and B cells corresponding to a feature are activated from the input coalition through the stimulated type 12 links. The locations of the hidden B-cells to be activated from an input coalition depends on the transformational offsets stored in the type 1 links and the physical offsets stored in the type 2 links between input and hidden layer. Whenever a hidden B-cell is activated it competes with the other B-cells connected to the same A-cell, and the winner represents the approximate position of the feature-object pair. Once a hidden B-cell establishes itself as the winner for an input activation, it gets biased to that conjunctive connection coming from the input layer (biased connection) and do not take part in the competition process with other hidden B-cells so long as the scanning of the input image is not complete. The principle of biased connection is explained in the next section (Fig.7.6). Due to the property of attenuable LTM, the links

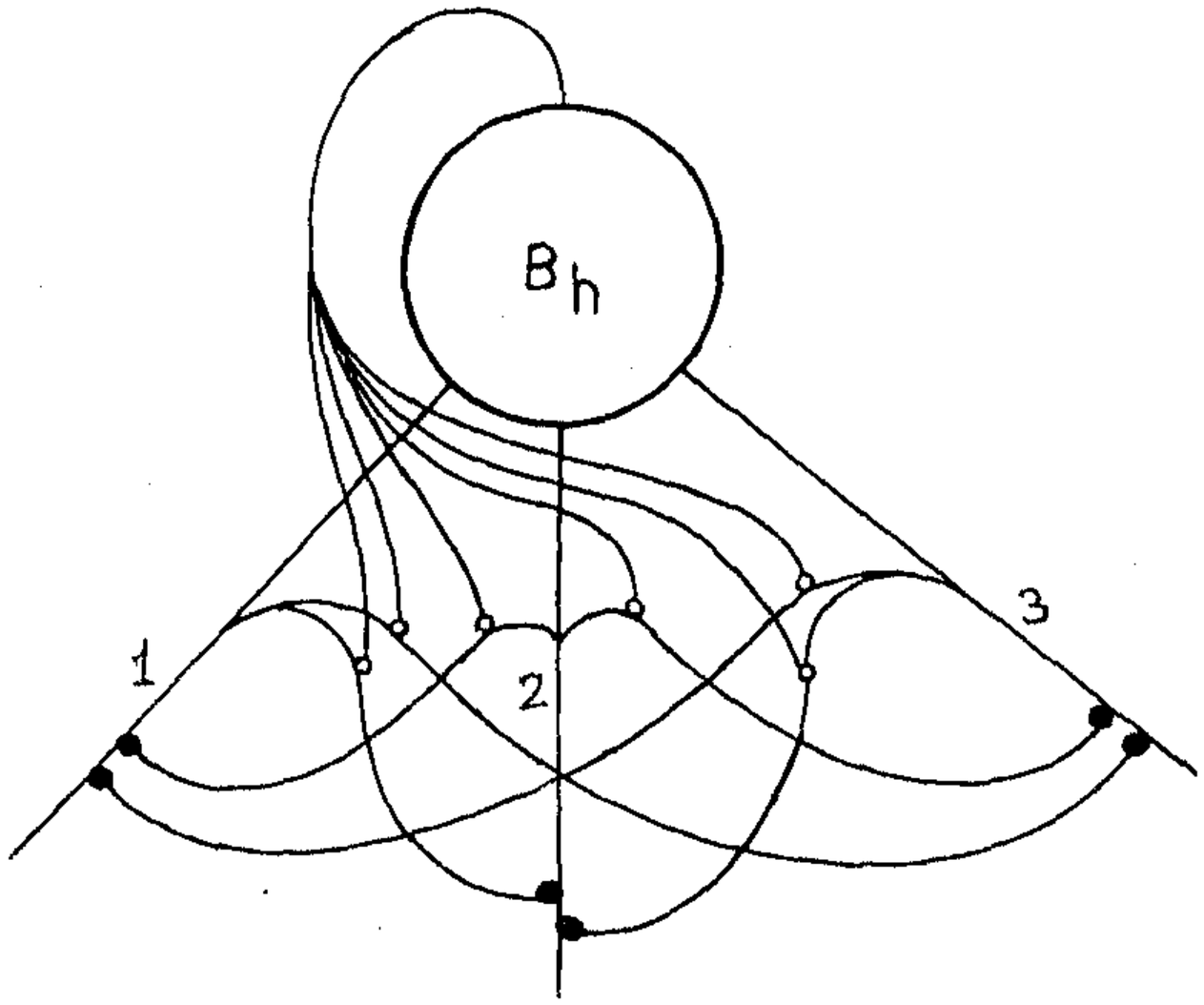


Figure 7.6: A network for biased connection of the links to the node Bh. Each link inhibits the other links with the help of inhibitory modifiers. The inhibitory modifiers are in turn stimulated from the output of the node.

connected to hidden cells other than the winner get deactivated.

In this process of scanning the image, the features are mapped onto input layer and feature-object pairs and corresponding locations are activated in the hidden layer. Each input coalition try to activate a hidden B-cell at a location corresponding to the object and since the input coalitions are formed sequentially in the scanning process, a cluster of activations is formed in the hidden layer B-cells corresponding to an object.

Once the scanning of the image is over, the hidden layer A and B cells conjunctively send their activation values to the output layer through bottom-up links. Each output B-cell is connected to a group of hidden B-cells in proximity (as shown in Figs.7.7 and 7.8). Let us term this group of cells as the purview of an output cell. Each output node collects the activation values from its purview in the hidden layer through the bottom-up links. Each output node (B-cell) has a negative self-feedback associated with it. As soon as an output B-cell gets activated, it sends back its activation value down to the hidden layer through the top-down links. Once the hidden B-cells receive feedback activation, the nodes which are activated from the same input pair start competing between themselves. We call this as selective competition process which has been explained in Fig.7.9. (Note that the hidden B-cells compete within certain neighborhood during the scanning process of the image. However, in the settling process, the B-cells compete selectively and this is not confined within neighborhoods.) The hidden A-cell coupled with the winner hidden B-cell (in the selective competition process), corresponding to an input coalition, represents the most likely object to which the corresponding feature belongs. The winner B-cell computes the difference between input activation (since there is biased connection, as mentioned before, the hidden B-cell cannot receive activation from any other input coalition) and the feedback activation, and send the *differential support* to the output node through bottom-up link. Here, we like to mention once again that we are not considering the activation values of the A-cells, because the A-cells only modulate the signals by zero or unity, and the actual confidence about the presence of an entity is represented by the activation level of the corresponding B-cell.

In the output layer, B-cells having overlapped purview are activated due to the presence of a cluster in the hidden layer. The output B-cells compete within a small neighborhood and the winner represents the exact location of the object. Each hidden B-cell gets biased to an output B-cell from which it is getting maximum

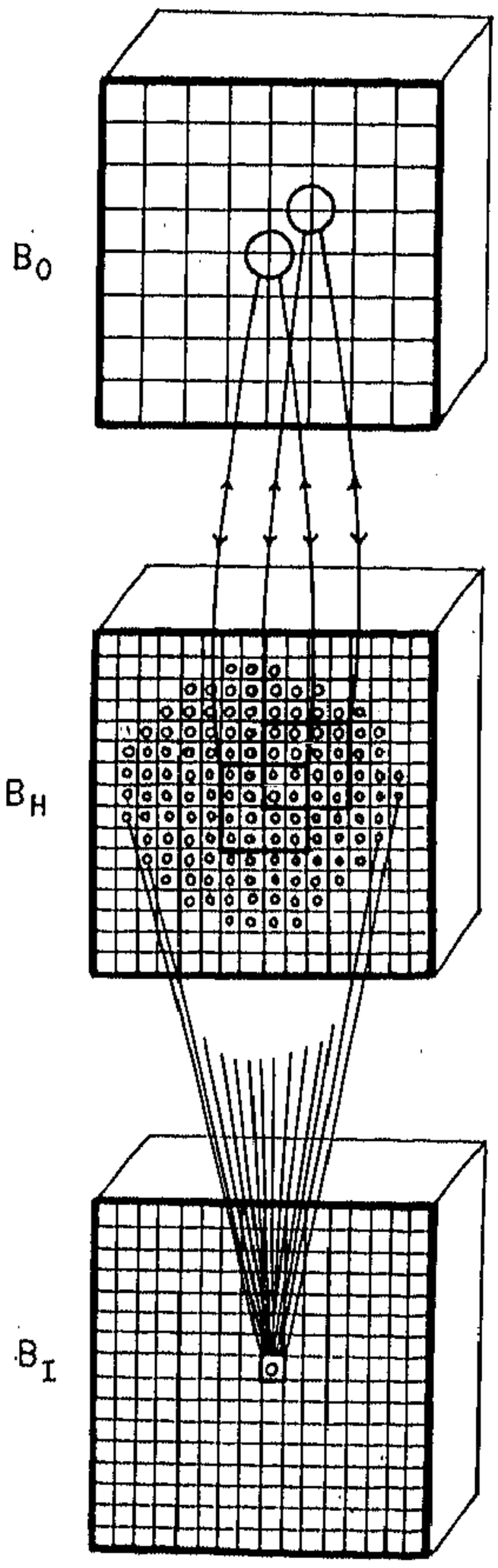


Figure 7.7: Connections from the input layer B-cells to the hidden layer B-cells and from hidden layer B-cells to the output layer B-cells. Links from an input B-cell connect to the hidden B-cells over a cone.

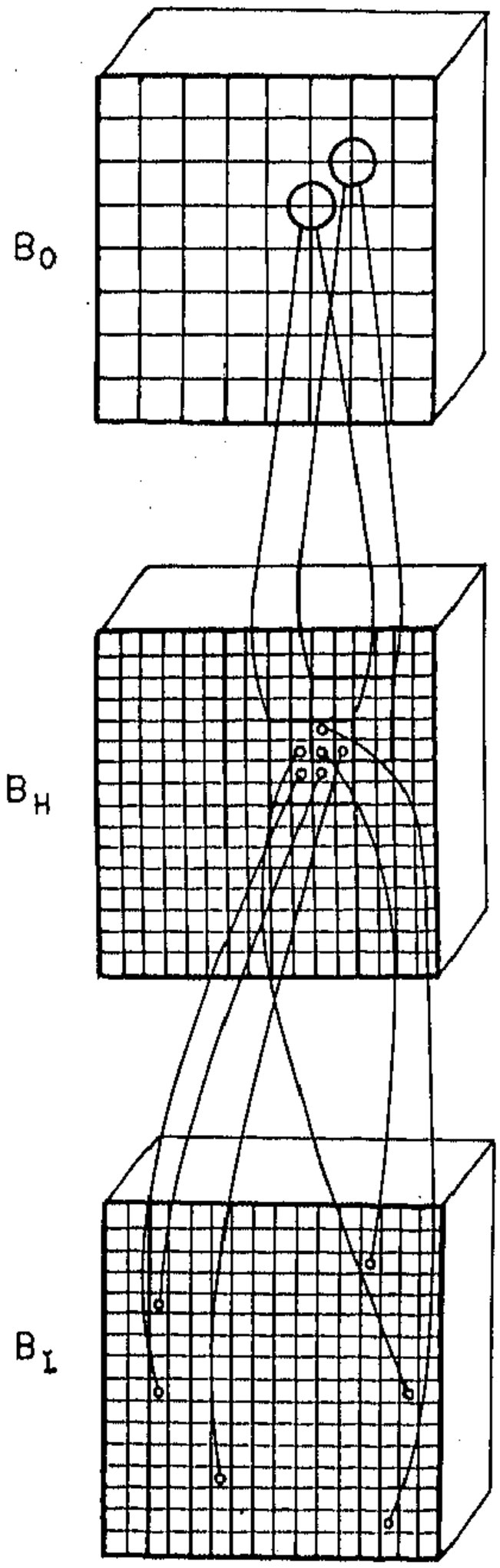


Figure 7.8: The relative position of an object with respect to its constituent features. For each input node, at least one hidden node is activated. The hidden node activations are collected at the output layer Bo.

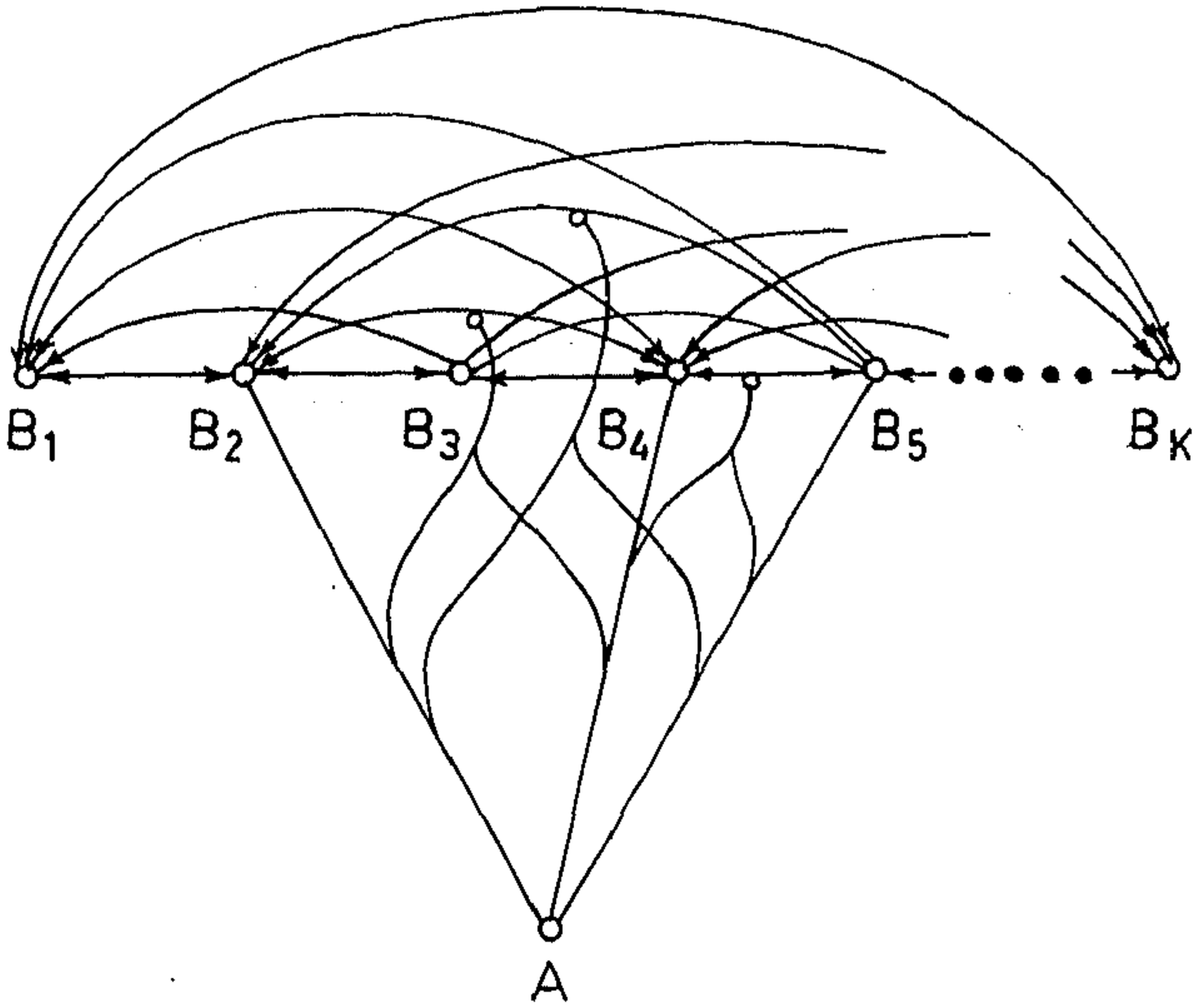


Figure 7.9: The connections for selective competition. The cells $B2$, $B4$, and $B5$ compete between themselves because only those particular internode links are stimulated by modifiers.

feedback (this happens with the same principle as the input bias). The differential activation is sent to the output B-cell to which the hidden B-cell is biased. Each output B-cell receives differential support and negative feedback, and it updates its activation value. The same process repeats and the activation values of the output nodes stabilize when the differential support and negative self-feedback become equal. After stability, the activation levels of the output B-cells represent the confidence about the presence of the corresponding objects (determined by the activated A-cells coupled with them) at that locations.

Now let us discuss in detail about the instantiation of hidden nodes (transformation embedding), special modules (like biased connections, selective competition, and competition within neighborhood), selective attention, and finally the dynamic behavior of the network.

7.4.1 Instantiation of Hidden Nodes

Type 1 links store the actual values of positional and orientational offsets (r, θ, ϕ) between features (represented by input A-cells) and objects to which the features belong (represented by feature-object combinations, corresponding to hidden layer A-cells). Type 2 links between the input layer and the hidden layer have fixed weights and the weights decrease with the increase in physical offsets between the locations of input B-cells and hidden B-cells. Each type 1 and type 2 link can be viewed as a composition of three links, one to represent r, one to correspond θ and the other to correspond ϕ. Let us denote the weights of the type 1 and type 2 links from input to hidden layer by $W1$ and $W2$ respectively. The values of $W2$ are mathematically given as

$$W2^* = \frac{\epsilon_*}{\epsilon_r^2 + \epsilon_\theta^2 + \epsilon_\phi^2} \tag{7.1}$$

where, '*' stands for r, θ or ϕ, and $(\epsilon_r, \epsilon_\theta, \epsilon_\phi)$ represents the physical offsets.

Let the i^{th} input A-cell and j^{th} B-cell conjunctively activate the k^{th} hidden A-cell and l^{th} hidden B-cell. Then the activation received by the hidden B-cell (ubh) is given as

$$ubh_l = (W1^r_{ik}.W2^r_{jl} + W1^\theta_{ik}.W2^\theta_{jl} + W1^\phi_{ik}.W2^\phi_{jl})xa_i.xb_j \tag{7.2}$$

where xa and xb are the activation values of the input A and B cells respectively (the activation received by hidden k^{th} A-cell is also equal to ubh). It is clear from

the Equn.(7.2) that the activation of the hidden layer B-cell would be maximum for which the stored transformations in $W1$ links perfectly match with the physical offsets represented by $W2$. In other words, this kind of activation helps the network to form cluster of activations in the hidden layer B-cells at a particular location when an object is present at that location. The transformation values are also incorporated in other networks [189]. However, the way it has been incorporated here is different from that used in [189].

The type 2 links carry activations from the input to the hidden layer over a cone, where the weights of the links decrease as the distance between the hidden nodes and the input node increase. If a hidden node has the same location as that of the input node then according to the equation 7.1 of the weights would become infinitely large. In order to have finite set of weights in such cases the hidden layer position and orientation can always be defined in such a way that there exists a minimum deviation by $(|\Delta x/2|, |\Delta y/2|, |\Delta \xi/2|)$ from the input layer, where Δx, Δy, and $\Delta \xi$ are the size of slots in position and orientation represented by a neuron.

7.4.2 Selective Attention

As described before, selective attention is a pseudo-parallel mechanism where different parts of an object are attended sequentially to get an idea about the object. This technique has been used in the proposed model. Features mapped onto the input layer are scanned sequentially to activate the proper hidden nodes.

In the scanning process, each time a particular zone is attended. During scanning, features appearing in the zone of attention are mapped onto input layer, and input coalitions activate the hidden nodes. The size of the zone of attention approximately depends on the sparseness of the features. The zone of attention should be such that more than one feature belonging to the same object do not appear simultaneously. Whenever more than one feature belonging to the same object appear simultaneously, they would be mapped into the same location in the hidden layer. Since each hidden B-cell can take care of only one feature, there will be a collision. Each B-cell in the hidden layer is provided with a mechanism for collision detection which in turn, helps in controlling the size of the zone of attention.

Attention Control Network

The attention control network (ACN) is coupled with the links from the input layer B-cells to the hidden layer B-cells. A link would be able to carry activation from the input layer to the hidden layer only when it is stimulated by ACN. The mechanism of attention is applied only in the phase of initialization of the network. Once the network is initialized the network is able to update the states of the nodes simultaneously.

The size of the zone of attention has a default value. Whenever a conflict arises in any hidden B-cell, it informs ACN and the ACN reduces the size of the zone of attention. The process of reducing the size of attention is stopped whenever the signal from the hidden layer indicates that there exists no conflict. The size of attention zone gradually increases as the ACN shifts the zone of attention.

Conflict Detection

Each hidden B-cell is able to detect a conflict whenever it arises between more than one nonzero signal coming from the input layer. To detect a conflict situation and consequently to inform ACN, each hidden node has a separate part, namely, conflict detector (CD). CD functions in the following way.

Let $\mathbf{u} = [u_1, u_2, \cdots, u_l]$ be the input vector to a particular B-cell. If there exists no conflict then

$$u_i u_j = 0, \quad \text{for all } i, j.$$

In other words,

$$\max_{i \neq j} \{u_i u_j\} = 0.$$

The conflict detector of each hidden B-cell has a transfer function given as

$$v1 = f(\max_{i \neq j} \{u_i u_j\} - \varepsilon), \tag{7.3}$$

where $f(.)$ is a step function which is given as

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

A small positive constant ε is used for noise tolerance (ε may also be chosen as zero). The ACN takes the outputs from CDs of all hidden B-cells and detects the

maximum signal. If the maximum signal is nonzero then it infers that a conflict exists at some hidden B-cell and consequently, it reduces the size of the attention zone.

7.4.3 Special Modules

Biased Connection

The selective attention mechanism remains active only during the initialization process of the network. In the settling process, during the updating of activation values of the nodes, the selective attention mechanism is no more active. On the other hand, the hidden B-cells which have already been fired, receive signal only through that particular connection which activates the hidden B-cell in the initialization process. In other words, each hidden B-cell has the property of being biased to a particular link through which it will accept the activation in the settling process. This particular property can be embedded into the links connected to a hidden node using a set of deactivating modifiers as shown in Fig.7.6. Each link connected to a hidden node inhibits the other links (connected to the same node) to carry activations with the help of deactivating modifiers. The node, in turn, stimulates the deactivating links. The mechanism works as follows. Suppose, a node (Bh) is connected to three links, namely, link 1, 2 and 3. Let, in the initialization process, link 1 carry activation at some instance. It has been mentioned before that due to the selective attention mechanism the node will be able to receive signal only through one link at a time. The node gets activated by the signal carried through link 1. As soon as the node gets activated, it stimulates all deactivating links between links 1 & 2, 1 & 3, and 2 & 3. Since link 1 carries activation, links 2 and 3 are inhibited and they cannot carry signals any further. On the other hand, since links 2 and 3 are not carrying any signal at that instance, the link 1 is not inhibited. In the recognition process, the link 1 continues to carry signals while links 2 and 3 remains inhibited. It is to be noted that such kind of biased connection will occur only when the node becomes activated. If there exists some conflict in the node then the node will not be activated. As a consequence, the biasness of the connection will eventually occur only when ACN ensures that the node is receiving only one signal.

Selective Competition

In the settling process of the network, each hidden B-cell competes with other B-cells which are connected to the same input B-cell. This competition allows only one B-cell (winner-take-all) to be able to send differential activation to the output layer. This is, as if, the same type of B-cells compete among themselves, and the type of the cell is determined by the input signal or the input node. This can be realized if the type information can somehow be extracted from the links. The connections presented in Fig.7.9 show how this can be implemented. Suppose, two B-cells, namely, B_{h1} and B_{h2} have to compete among themselves. The inhibitory links connecting B_{h1} and B_{h2} are activated by the pi-connection from the links going to the cells B_{h1} and B_{h2} from the same input B-cell. The activation is performed using the modifier connection. Usually, all the internode links between the hidden B-cells cannot carry activation. When these links are stimulated by the modifiers from the input nodes then only they play active role in the competition between the B-cells.

Competition over a Local Neighborhood

The output B-cells have local competition. This is necessary because each output B-cell is connected to a group of hidden B-cells (purview of the output B-cell). The cells are arranged in such a way that two different groups of hidden B-cells may or may not have overlap between them. In case there is an overlap, confusion would arise regarding the location of objects. For example, let the hidden layer consists of two neighboring groups G_1 and G_2. In that case, some B-cells from G_1 and G_2 can be activated due to the presentation of the feature set corresponding to a single object. The output layer will collect activation values from the groups G_1 and G_2 (say output B-cells O_1 and O_2), and as a result, indicate that the object is present at two neighboring locations. If there were a group (say G_3) having overlap with both G_1 and G_2 such that most of the activated hidden B-cells fall into G_3 then the activation of the output B-cell (say O_3) collecting activations from G_3 would be higher than those of O_1 and O_2. If there exists competition between O_1, O_2, and O_3 then O_3 would indicate the actual position of the object and there would be no confusion. Moreover, if an object consists of a large number of features then the number of hidden cells required to represent the feature-object associations would be large. If there were no overlapping groups then output cells would be widely

separated and imprecision may arise.

The local competition can take place in several ways. Two examples are presented in Fig.7.10. In the first case, each group has overlap with eight other groups (figure shows only four spatially separated groups). Each output B-cell competes with eight neighboring output nodes in its local neighborhood. In the second example, each group of hidden B-cells has overlap with twenty six neighboring groups (only eight spatially separated groups are shown in figure). In this case, the output cell competes with the neighboring twenty six nodes. The overlapping between the groups of hidden B-cells can even be larger at the cost of larger number of output cells.

7.4.4 Dynamic Behavior of the System

Let us now describe the dynamic behavior of the system. In the initialization process, each activated coalition of input A and B cells activates a number of A and B cells in the hidden layer. Note that, the activation values of A cells in any layer represent the presence or absence of the corresponding entities, and therefore, the activation levels of A-cells are either 1 or 0. On the other hand, the activation values of the B-cells represent the confidence about the presence of the corresponding entities. In the consequent discussion, we will be considering the updating of the activation values of the B-cells only. The output layer B cells, in the initialization process, receive activation from the hidden B-cells over the corresponding purview. In the settling process, the output B-cells send their activation values back to the hidden layer B-cells. After receiving the top-down feedback, the hidden layer B-cells connected to the same input B-cell selectively compete with each other. After competition, for each input B-cell there exists one winner-take-all (WTA) hidden B-cell. Each WTA hidden B-cell computes the difference of the activation signal coming from the input layer and the top-down feedback received from the output layer. The difference of the activation values (differential support) is propagated to the output B-cell. Each output B-cell has negative self-feedback and updates its state depending on the differential activation value received from the hidden layer and the self-feedback. Before going to mathematical formulation of network dynamics, let us briefly discuss about the properties of the cells.

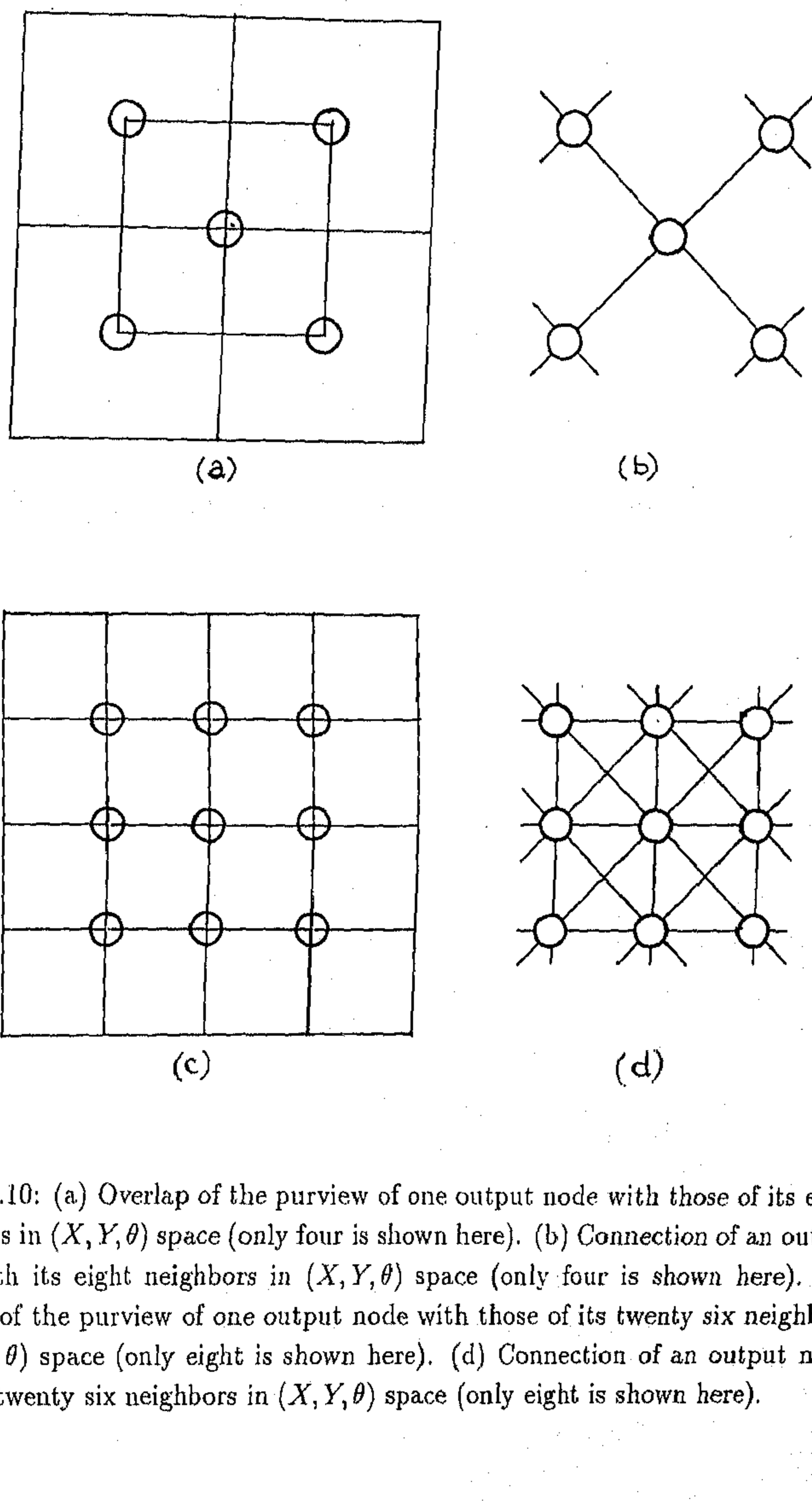


Figure 7.10: (a) Overlap of the purview of one output node with those of its eight neighbors in (X, Y, θ) space (only four is shown here). (b) Connection of an output node with its eight neighbors in (X, Y, θ) space (only four is shown here). (c) Overlap of the purview of one output node with those of its twenty six neighbors in (X, Y, θ) space (only eight is shown here). (d) Connection of an output node with its twenty six neighbors in (X, Y, θ) space (only eight is shown here).

Properties of the Cells

The input-output function of a hidden or output A-cell is formulated as

$$v = f\left(\max_i u_i - \varepsilon\right)$$

where v is the output of an A-cell, u_i is the i^{th} input, ε is a small threshold, and $f(.)$ is a step transfer function. Each input A-cell has a transfer function which produce maximum output for certain range of input. Mathematically,

$$f(x) = \begin{cases} 1 & \text{for } x_1 < x < x_2 \\ 0 & \text{otherwise} \end{cases}$$

The actual range of values (x_1, x_2) will be discussed in Section 7.6.

Each input B-cell has two parts. One of them holds the activation value and has a linear transfer function. The other part has a radial basis function [29] which produce maximum output for certain orientations. Each hidden layer B-cell has four parts. One of them detects if there exists any collision. One of them competes within a local neighborhood to represent the proper position of the hidden node activated by an input node as mentioned in Section 4.1. One part of each B-cell holds the activation value propagated from the input layer. The other part selectively competes with the other B-cells to determine which particular cell would support the output cells in the settling process. The response of collision detector part ($v1$) of the B-cells is already discussed. The portions which enable hidden B-cells to compete within local neighborhoods in the initialization process, have simple linear gain, i.e.,

$$v2 = u_i$$

where u_i is the activation received from the input coalition to which the hidden node is biased. The portions which compete selectively with other B-cells have also similar linear gain, i.e.,

$$v3 = fb$$

where fb is the feedback received from output layer. The portion which holds the activation received from input layer has exactly the same gain as $v2$.

In the output layer, each B-cell possesses two different parts, one of them represents the confidence about the presence of an entity at the corresponding location, which

can be mathematically expressed as,

$$vol = g\left(\sum_i u_i\right)$$

where $g(.)$ is an S-function, as described in Chapter 4 (in the output layer linear gain is not used). u is the input received from hidden layer over a purview. The second portion of each output B-cell locally competes with other cells which has exactly the same gain as vol.

The notations used here to represent output of different cells will not be used any further. In the subsequent discussion, we will be considering the updating of the activation values of that portions of output B-cells only which represent the confidence about the presence of an entity (i.e., vol). However, the notations used to represent the activation values would be different and clarified in due context.

STM Equations

Let us now mathematically describe the dynamical behavior of the network. Before going into the details of the dynamics, let us clarify some symbols used here. As mentioned before, $W1$ and $W2$ represent the weights of the type 1 and type 2 links between input and hidden layer. $w1$ and $w2$ denote the weights of the type 1 and type 2 bottom-up links respectively, and $z1$ and $z2$ denote the weights of the type 1 and type 2 top-down links respectively between hidden and output layer. The hidden layer A-cells are denoted by an ordered pair (i,k) where i and k denote the input and output A-cells to which it is connected. Let the $(i,k)^{th}$ hidden A-cell and $(j,l)^{th}$ hidden B-cell be conjunctively connected to the k^{th} output A-cell and l^{th} output B-cell, i.e., $(j,l)^{th}$ hidden B-cell is within the purview of the l^{th} output B-cell and connected to j^{th} input B-cell. The updating of the activation level of an output B-cell can be written as

$$\frac{dubo_l}{dt} = \sum w1_{ik}.w2_{jl}.e_{ijkl} - w_s.(vb_l)^2, \tag{7.4}$$

where ubo total activation received by an output B-cell. The summation is taken over the purview of the output cell (l) and only for those links which are selectively stimulated. vb_l is the output of l^{th} output B-cell (note that this is same as vol, mentioned before) which is given as

$$vb_l = g\left(ubo_l\right).$$

The differential support e_{ijkl} can be written as

$$e_{ijkl} = \begin{cases} (hb_{ijkl} - fb_{ijkl}) & \text{if } fb_{ijkl} > fb_{ijk'l'} \text{ for all } k' \neq k \wedge l' \neq l \\ 0 & \text{otherwise} \end{cases} \tag{7.5}$$

where fb is the feedback support given as

$$fb_{ijkl} = z1_{ki}.z2_{lj}.vb_l.va_k \tag{7.6}$$

hb_{ijkl} represents the activation value received by the hidden B-cell from the input layer in the transformed space due to the activation of i^{th} input A-cell and j^{th} input B-cell (Equn.7.2). Note that, equation 7.5 shows that differential support is computed only at that node which receives maximum feedback for a given i,j. The updating of the activation values of output B-cells is analogous to the dynamics presented in Chapter 4 (for X-tron). Due to the similar reason, as presented in Chapter 4, the system dynamics converges here also.

7.5 Learning Process

The weights of the type 1 links between input and hidden layer and the type 1 bottom-up and top-down links between hidden and output layer are learned under supervised mode. (Note that, the weights of type 2 links are fixed and are not updated). The updating of the weights takes place at two different levels. The weights from the input to the hidden layer represent the transformational offsets from the feature reference frame to the object reference frame, and the weights from the hidden layer to the output layer represent the likeliness of appearances of particular feature-object combinations. The two stages of learning are being discussed below.

The learning methodology can be structured as follows :

Step 1 : Present the features (i.e., activate A-cells) and their locations (i.e., activate B-cells) at the input layer. Present the corresponding object with its location at the output layer. It is to be noted here that during the learning process only one object can be present at a time.

Step 2 : Check if the required transformation values from the feature reference frame to the object reference frame already exist in the links from the input

layer to the hidden layer. If the transformation value for a feature-object pair exists then the corresponding hidden B-cell would be activated within the purview of the output B-cell. Otherwise there will be no such activated hidden B-cell for that pair.

If any activated hidden B-cell does not exist then

A : check if there exists any bottom-up and top-down link between that particular feature-object association (corresponding activated hidden A-cell) and the object (the desired activated output A-cell).

If it exists then

i : update the weights of the bottom-up and top-down type 1 links (i.e., update $w1$ and $z1$) considering that the feature-object association is absent at that instant, i.e., decrease the corresponding weights.

ii : activate a hidden B-cell within the purview of the output B-cell such that hidden B-cell is nearest to the center of the purview.

iii : learn transformation values in the links from the input to the hidden layer.

If the corresponding bottom-up and top-down links do not exist (i.e., either the object is a new one or the corresponding feature did not appear in the object in the previous trials) then

i : allocate a hidden A-cell for that feature-object association and activate a hidden B-cell within the purview of the output node such that the hidden B-cell is nearest to the center of the purview.

ii : create bottom-up and top-down type 1 links between the hidden A-cell and the output A-cell.

iii : initialize the weight of the bottom-up type 1 link to a certain small value and the weight of the top-down type 1 link to unity.

If the transformation exists then

A : Check if the activated hidden B-cell is at the center of the purview of the output B-cell, and adjust the weights of the type 1 links from input layer to the hidden layer (i.e., the transformational values) depending on the desired position of the hidden B-cell (i.e. center of the purview) and the actual position of the hidden B-cell.

B : update the weights of the bottom-up and top-down type 1 links between the hidden layer and the output layer.

Step 3 : copy the weights of the bottom-up and top-down type 1 links between the other type 1 links connected with the same hidden and output A-cells.

Step 4 : copy the weights of the type 1 links from input layer to the hidden layer between other type 1 links within the same group with same gating channel (gating channel is being discussed subsequently).

7.5.1 Weights from Input to Hidden Layer

The average transformational offset from the feature reference frame to the object reference frame are adaptively captured by learning the weights from input to the hidden layer. Note that, the type 2 links store information about physical offsets $(\epsilon_r, \epsilon_\alpha, \epsilon_\theta)$ as described in equation 7.1. An object may have multiple instances of the same feature located at different places. In other words, there should be provision to store multiple instances of transformational offsets between an input A-cell and a hidden A-cell. To encounter this problem, each hidden A-cell has more than one gating channel (Fig.7.11) and to each gating channel a group of type 1 links is connected. During learning, the transformational weights ($W1$) are copied between the type 1 links only if they are in the same group. The weights of the links in two different groups are not copied although they connect the same input and hidden A-cells. As a result, even if multiple instances of the same feature exist in a particular object, the different transformational offsets are stored in the different groups of the type 1 links and they would not affect each other.

Let i^{th} A-cell and j^{th} B-cell in the input layer are activated, correspondingly k^{th} A-cell and l^{th} B-cell should be activated in the output layer. In that case, $W1$ is updated in such a way that $(i,k)^{th}$ hidden A-cell and a B-cell at the same position of l^{th} output B-cell get activated. Let us denote the hidden B-cell with the same notation, i.e., l. In that case, updating of the weight of type 1 link from input to hidden layer is given as

$$\frac{dW1^*_{ik}}{dt} = \alpha_{ik}(\epsilon_* - W1^*_{ik}) \tag{7.7}$$

where $*$ stands for r, θ, or ϕ. α is agility factor whose value decreases with the number of presentations. The agility factors of input and output nodes have already been discussed in Section 4.4.2. Here, we consider that hidden nodes also have their agility factors. ϵ_* represents the physical offset between the position of input B-cell and the desired position of output B-cell. With this learning rule

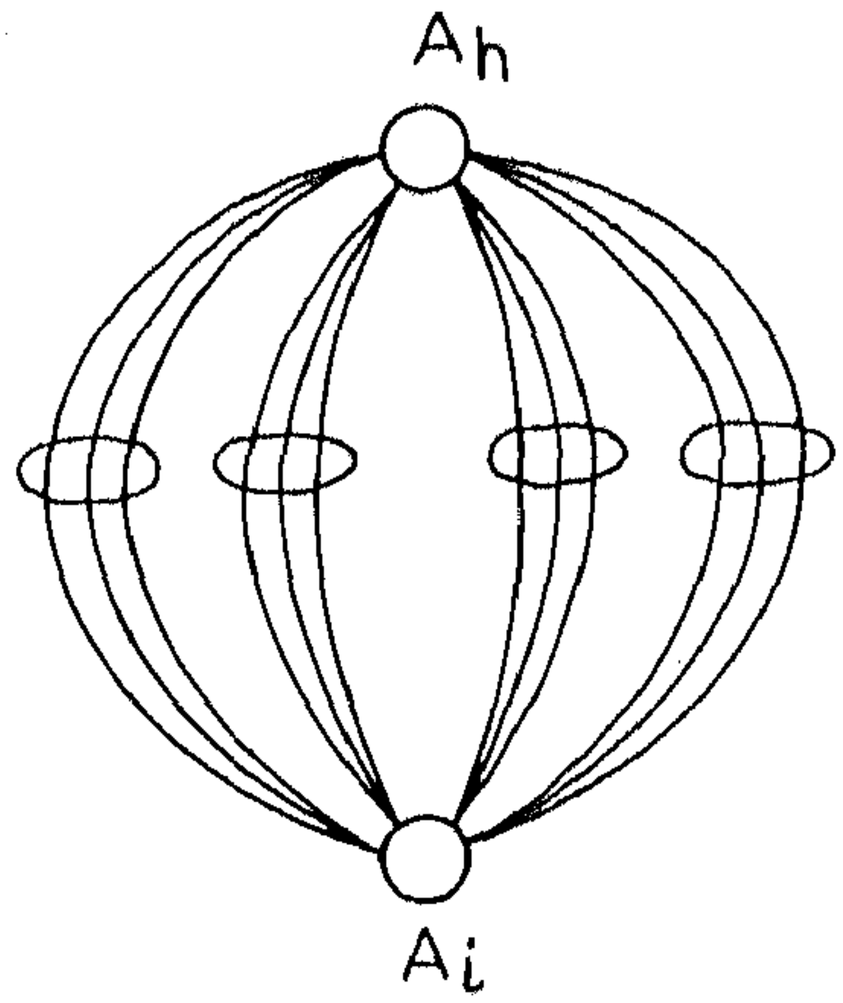


Figure 7.11: Diagram of gating channels of the node Ah. A number of links are connected between the nodes Ai and Ah. The links going to the same channel only have equal weights. The links within a loop represent the links going to the same channel.

(equation 7.7), basically an iterative averaging of the offsets between the locations of features and objects is performed under different presentations. However, in such kind of learning, question may arise regarding how to measure ϵ_r locally. In the supervised process of training, the corresponding offsets between the features and objects can be supplied to the network.

Another technique may be used to learn $W1$ values using gradient descent technique. The activation value received by a hidden layer B-cell is given by (Equn.7.2)

$$ubh_{jl} = (W1^r_{ik}.W2^r_{jl} + W1^\theta_{ik}.W2^\theta_{jl} + W1^\phi_{ik}.W2^\phi_{jl})..xa_i..xb_j,$$

and the output of hidden B-cell is same as the input received by it, i.e., $hb_{jl} = ubh_{jl}$ The error at the l^{th} cell is given by

$$E_l = \frac{1}{2}(t_{kl} - hb_{jl})^2$$

The change in $W1$ can be given as

$$\Delta W1^*_{ik} = \eta \frac{\partial E_l}{\partial W1^*_{ik}} \tag{7.8}$$

After algebraic computation this becomes

$$\Delta W1^*_{ik} = \eta(t_{kl} - hb_{jl})W2^*_{jl}xa_ixb_j \tag{7.9}$$

where ubh is the total input to the hidden layer B-cell and $*$ stands for either r, θ, or ϕ, and η is the rate of learning.

We have used the first technique, i.e., iterative averaging of the transformational offsets in our implementation. Once the transformation values are learned for a feature-object combination, they are copied over the other type 1 links (corresponding to the same feature-object combination) which are conjunctively connected to other type 2 links. This kind of copying the weights to other links has been introduced in [28].

7.5.2 Weights from Hidden to Output Layer

In the second part of training, the feature importance values with respect to the objects are learned using the same learning rules as presented in Section 4.4.2, i.e., the weights are iteratively updated depending on the nature of co-occurrence of the features and objects. The asymptotic measures for weights can be written as

$$z1_{ki} = p(f_i|o_k) \tag{7.10}$$

and

$$w1_{ik} = \frac{p(f_i|o_k)p(o_k|f_i)}{\gamma + \sum_i p(f_i|o_k)p(o_k|f_i)} \tag{7.11}$$

The constant γ is used to incorporate Weber's law [130].

The updating of $w1$ can be formulated as

$$\frac{dw1_{ik}}{dt} = \left(\alpha_i\delta_{kl}z1_{ki} + \alpha^o_k\left(\frac{w1_{ik}}{z1_{ki}}\right)\right)xa_ixb_jt_{kl} - (\alpha_ixa_ixb_j + \alpha^o_kt_{kl})\,w1_{ik} \tag{7.12}$$

In this equation it is considered that the $(i,k)^{th}$ hidden A-cell and $(j,l)^{th}$ hidden B-cell are conjunctively connected to the k^{th} A-cell and l^{th} B-cell in the output layer. The activation values originally received by the hidden B-cells are considered to be exactly the same as the input B-cells in the transformed space. Note that, in equation (7.12), no subscript j and l have been used in the left side which indicates

that the weights are copied over all positions. t_{kl} is the desired output of the k^{th} object at the l^{th} location in the B-cells. δ_{kl} is given as

$$\delta_{kl} = \frac{t_{kl} - va_k vb_l}{\gamma g'(ubo_l)} \tag{7.13}$$

Here the activation of the A-cell is not considered in the denominator since the activation of the A-cell is unity if the object is present. In the learning process, if it is found that the A-cell corresponding to k^{th} object is absent then it is allocated and necessary connections are made. Similarly, in the hidden layer, corresponding to the necessary feature-object pairs, A-cells are allocated. The weights from the hidden layer to the output layer are set to a small value. Therefore, after the first part of learning, hidden B-cells within the purview of the k^{th} object would receive some activations and therefore it will be always active.

The updating of the top-down links are given as

$$\frac{dz1_{ki}}{dt} = \alpha_k^o t_{kl}(xa_i xb_j - z1_{ki}) \tag{7.14}$$

α is the agility factor of the nodes which provides an approximate measure of how long the node has been active during the training phase. The updating of α_i is given as

$$\frac{d\alpha_i}{dt} = -(\alpha_i)^2 xa_i xb_j \tag{7.15}$$

Here, it is to be noted that agility factors are attributes of the A-cells only and it is updated by the activation of the corresponding B-cells. The B-cell activations are received by the A-cells through stimulated type 3 links. The updating of α_k^o is given as

$$\frac{d\alpha_k^o}{dt} = -(\alpha_k^o)^2 t_{kl} \tag{7.16}$$

In the second stage of learning also, the weights of the type 1 links are copied over other type 1 links representing the same feature-object combinations.

7.6 Implementation of the Network

The network has been simulated in *SUN* 3/60 workstations. The images of four different 2D objects (hammer, spanner, plier, and knife) have been presented to the network. The position of each object is specified by the co-ordinates of the centroid of the object and orientation is represented by the orientation of the principal axis of the object.

7.6.1 Representation of the Features

Different characteristic features like lines, edges, corners, holes etc. can be considered for object description. Here, only corners are considered as the characteristic features. The significant interrelations among the corners can also be taken into consideration (since the network learns the transformations from the feature reference frame to the objects reference frame, the interrelations are not used in the work).

The silhouette images (512×512) of the objects are considered here. The image has been segmented using graylevel thresholding. The threshold was selected to be 90 (maximum gray value in the image was 255). The image has been smoothed by applying growing and shrinking operations on it. The boundaries of the objects are detected by checking the 4-neighborhood. The corner or break points on the boundary have been detected by using the divide and conquer strategy as developed by Han et al. [246] (note that, the corners could have been detected by any other suitable algorithm).

Whenever a corner is detected, it is supposed to have a certain curvature or cornerity value and a direction. The curvature value depends on the angle of the corner. Depending on the cornerity value a corner is encoded into a particular feature. The cornerity value at a certain break point is measured as the angle between the two lines joining the two neighboring break points on either sides along the boundary. For example, let X be a break point where the cornerity value is to be measured. Let Y and Z be the breakpoints first encountered when the boundary is scanned clockwise and anticlockwise respectively starting from X. In that case the cornerity value at X is the angle between the lines XY and XZ, and the direction of the corner is the direction of the bisector of the angle YXZ.

7.6.2 Encoding of the Features

The input image (512×512) is spatially divided into 64×64 grids so that each grid contains 8×8 pixels. In each grid location at most a single feature is allowed to be present. The actual grid size depends on the nature of the image. In each grid location a number of input nodes is present. Each input node represents a particular encoded feature at that particular location. First, let us consider the

way of encoding the features. The entire range of cornerity values is divided into a number of slots. Each slot is considered to be a separate feature. The corners are divided into slots of 9 degrees so that there will be 40 different input features. In the present scheme all the corners within a slot will be treated as the same feature. The slots of corners are

$$(0-9) \quad (9-18) \quad \cdots \quad (342-351) \quad (351-360)$$

i.e., a feature can be written as

$$c_i = \begin{cases} 1 & \text{if } i = \lfloor angle/9 \rfloor \\ 0 & \text{otherwise,} \end{cases}$$

where *angle* denotes the angle of the corner. c_i represents if the corresponding feature is present. If c_i is unity then xa_i (activation level of i^{th} input A-cell) is unity (we consider that the i^{th} feature is mapped to i^{th} input A-cell).

7.6.3 Representation of Locations

The entire range of angles is quantized into 60 slots to represent the orientation of the features and the objects, and each grid alongwith a orientation slot is mapped onto an input B-cell. Therefore, each input B-cell has a tolerance of 8×8 pixels and 6^o in orientation. The hidden layer also has $64 \times 64 \times 60$ grids to represent the input-output association. The output layer is divided into $16 \times 16 \times 15$ grids. Each grid location contains one output node, and in the junction of six neighboring output nodes another node is placed. As a result, each output node has a tolerance of 32×32 pixels spatially and 24^o in orientation. Each output node is connected to a group of 64 hidden nodes. This indicates that the network is able to accommodate at most 64 feature-object associations for any output node. In other words, the network is able to learn and recognize those objects which have less than 64 features. The input, hidden and output grids of B-cells have been presented in Fig.7.12.

7.6.4 Training and Testing

Four different objects (as shown in Figs. 7.13, 7.14, 7.15, 7.16, 7.17) have been considered. During the training phase of the system, each object has been presented to the network in different positions and orientations.

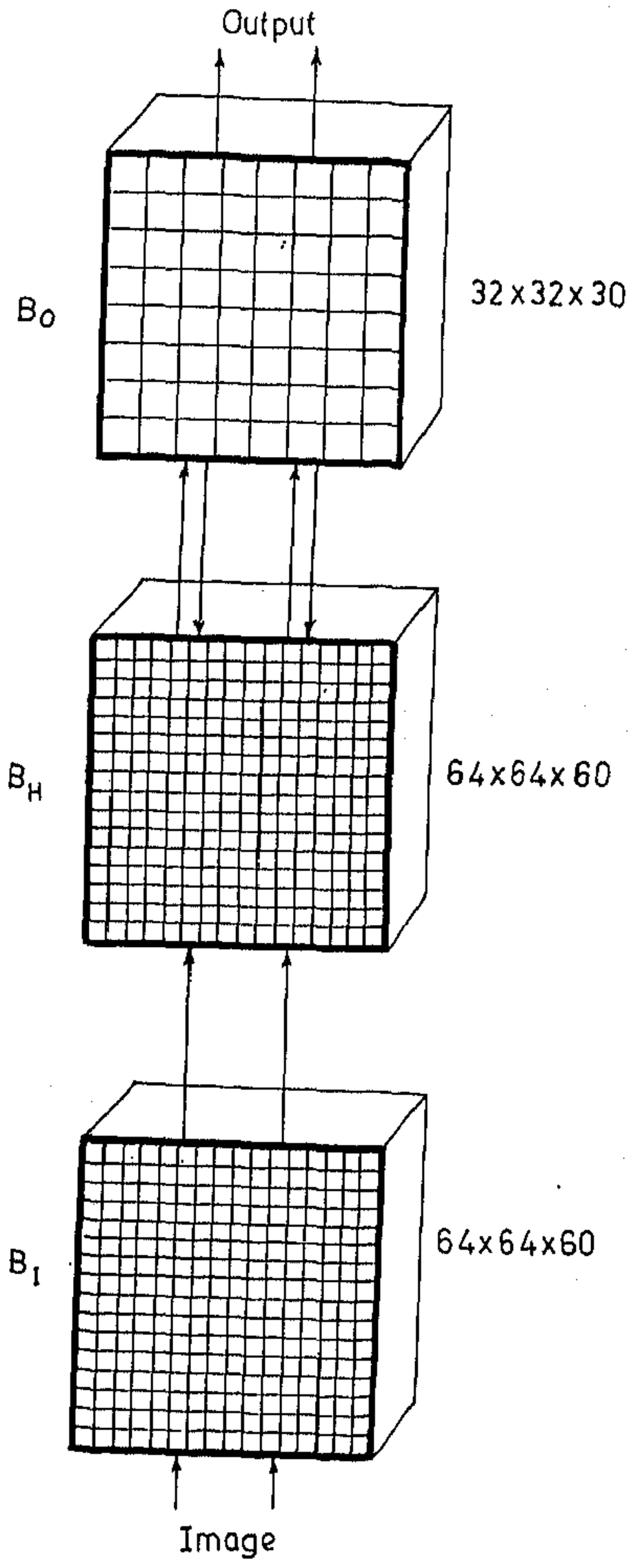
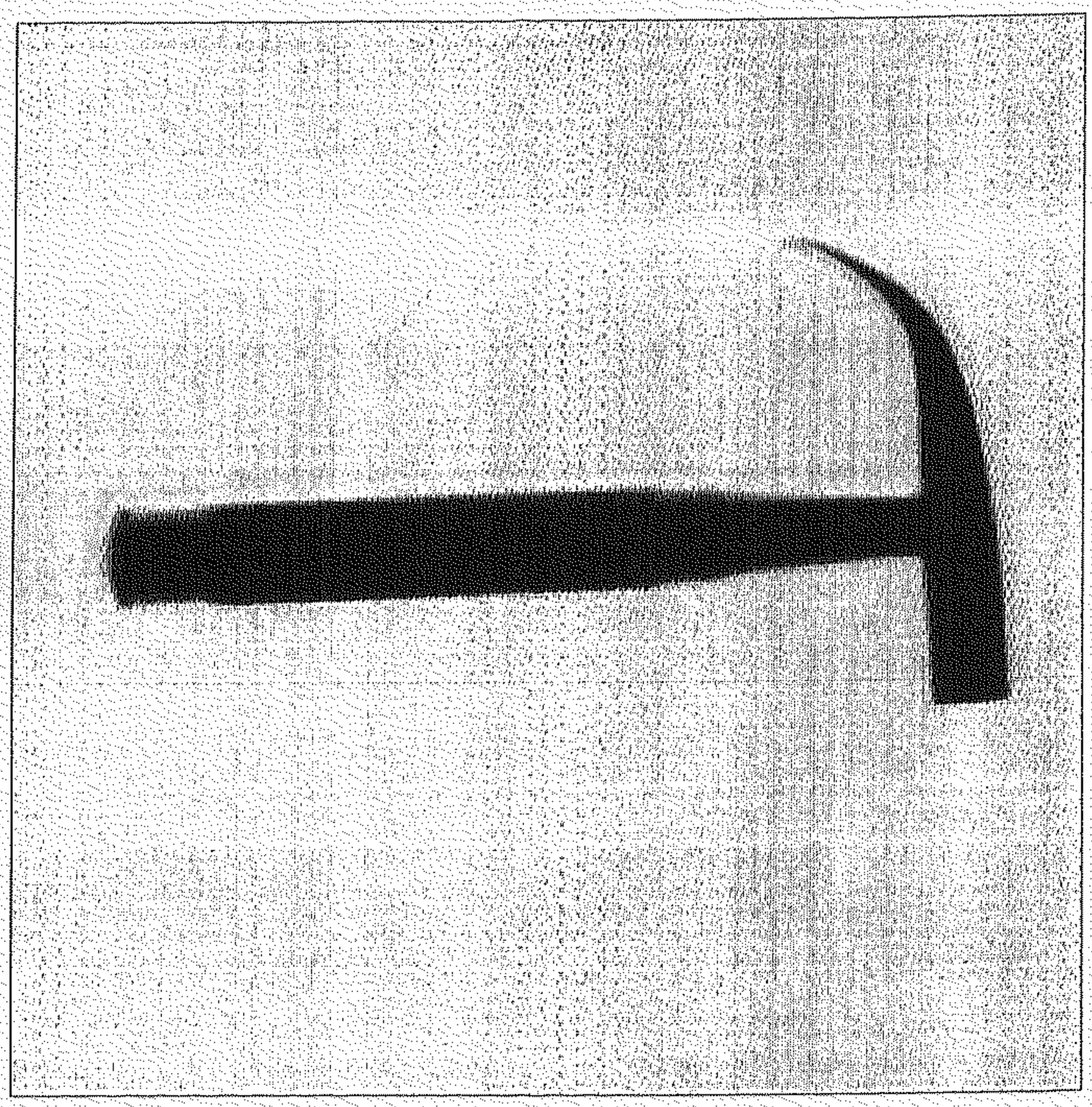
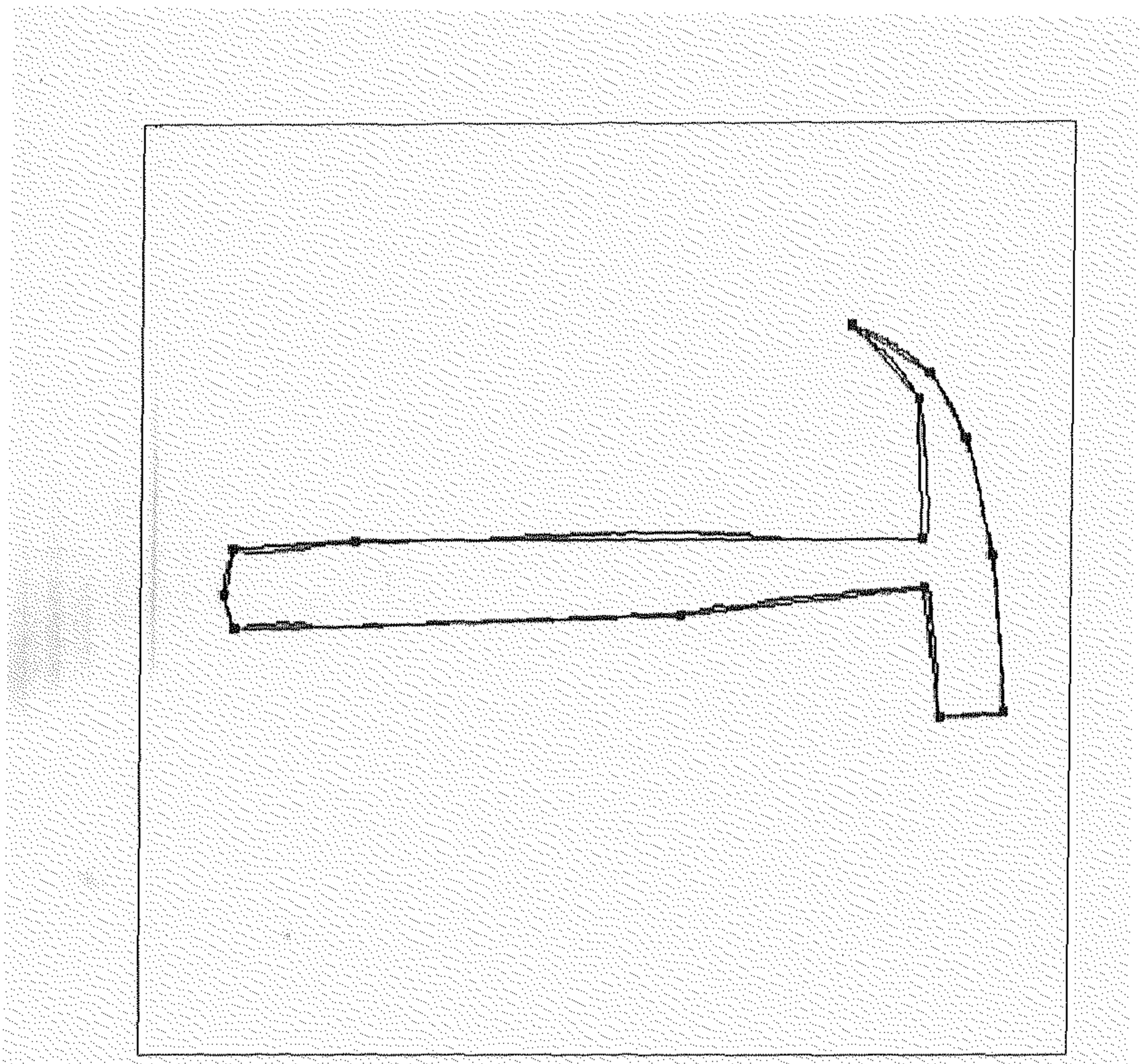


Figure 7.12: The actual gridsize for B-cells of the network. B_I, B_H, B_O are the input, hidden and output layers respectively.



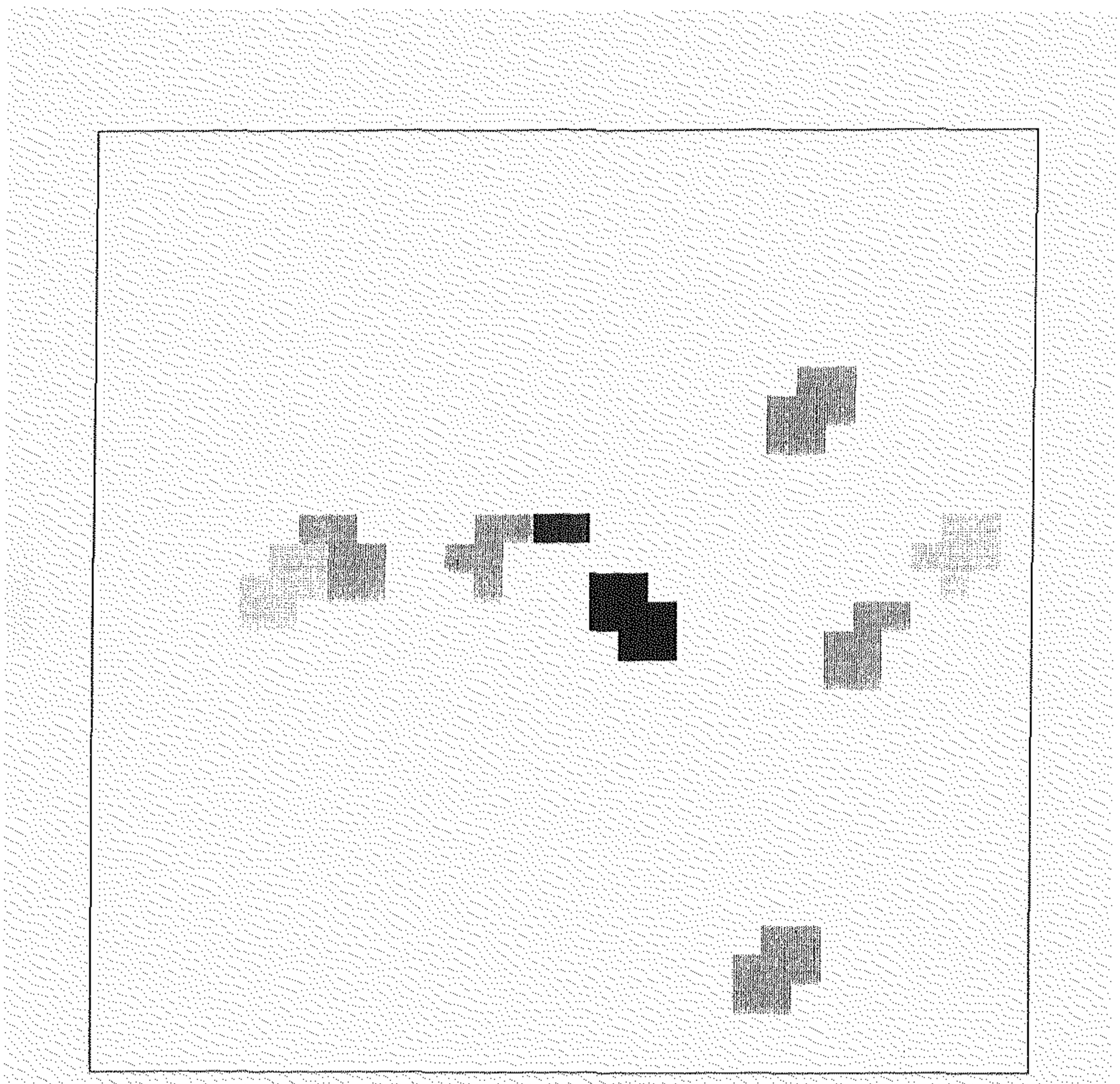
(a)

Figure 7.13 cont'd.



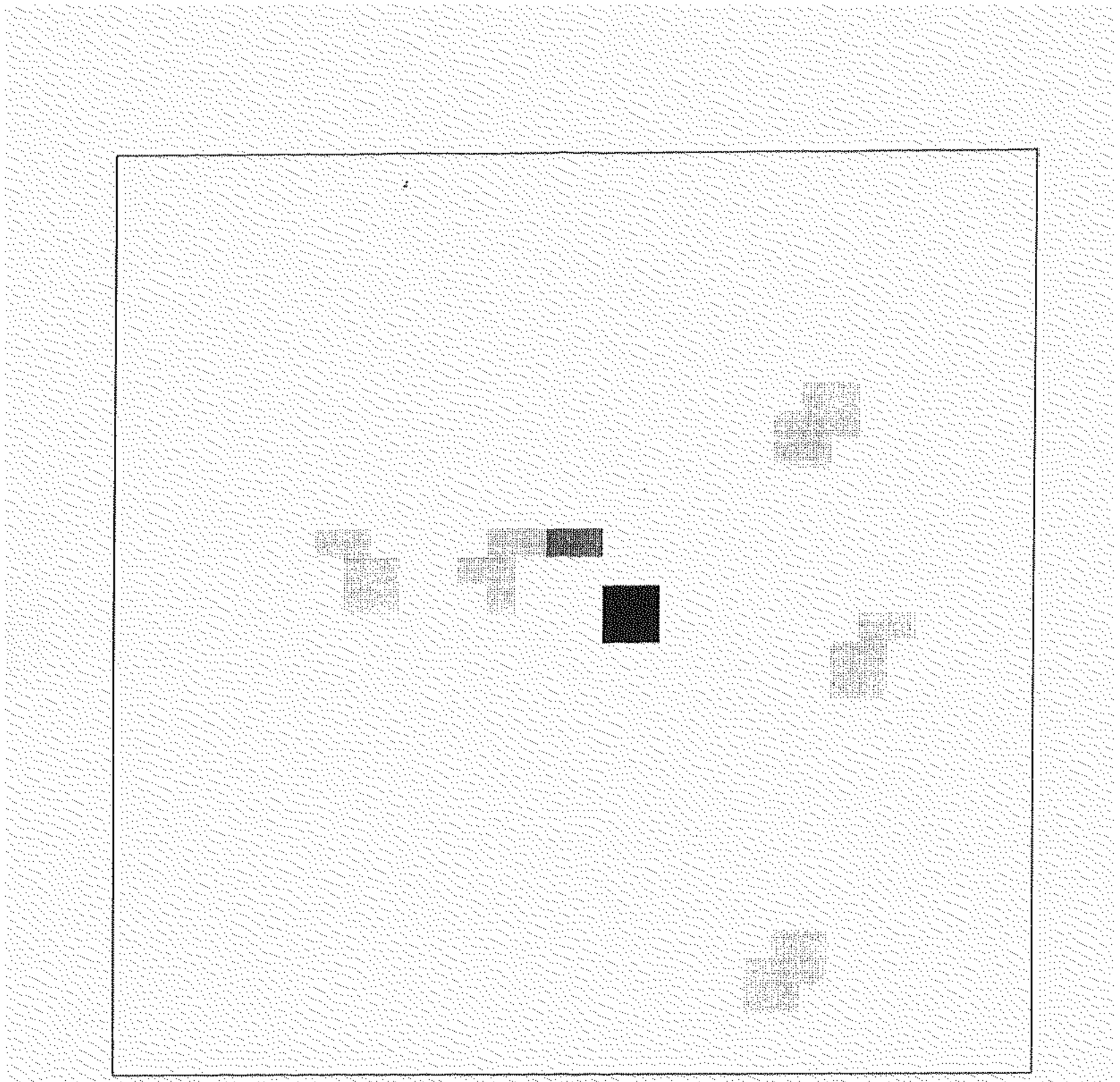
(b)

Figure 7.13 cont'd.



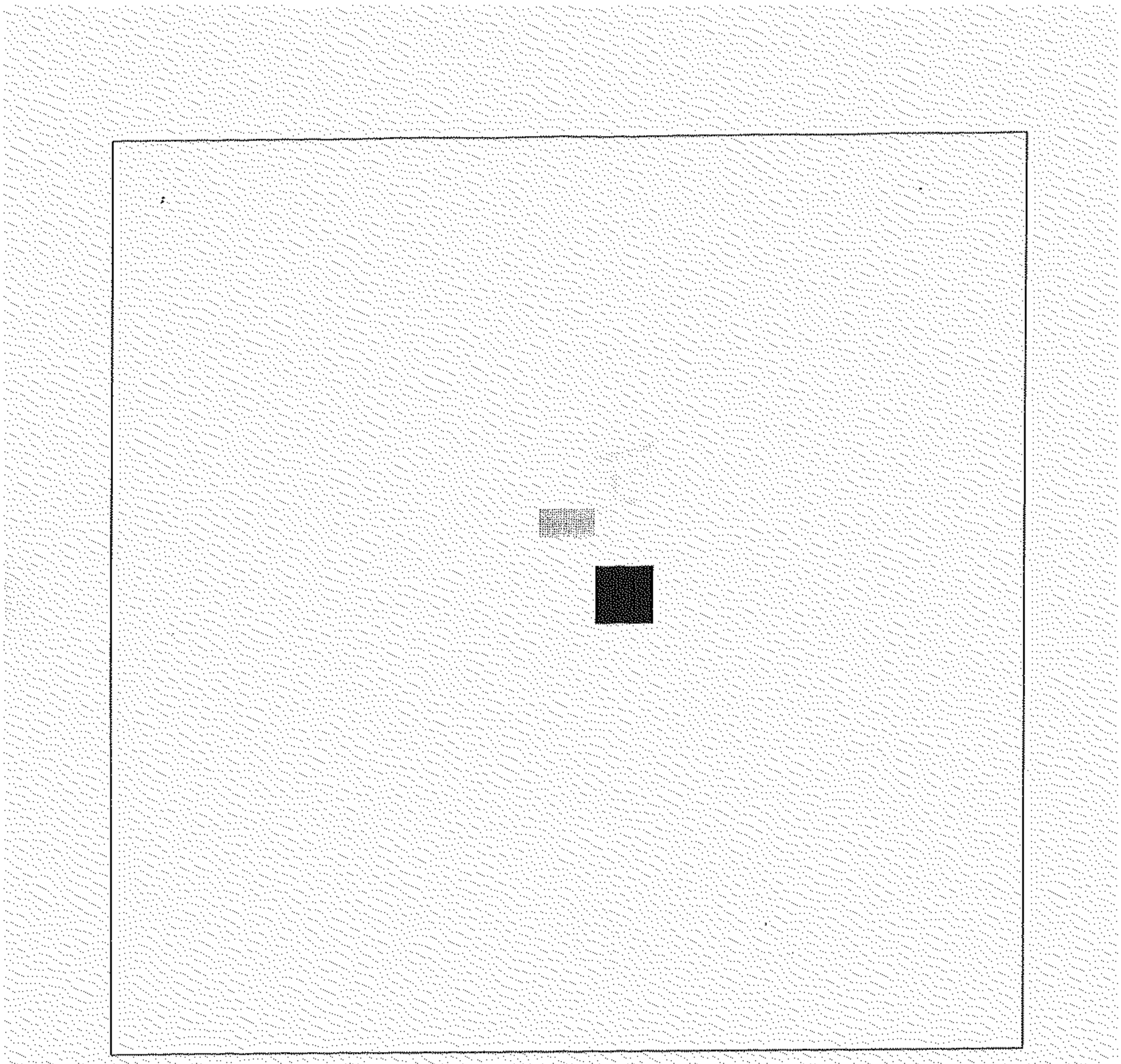
(c)

Figure 7.13 cont'd.



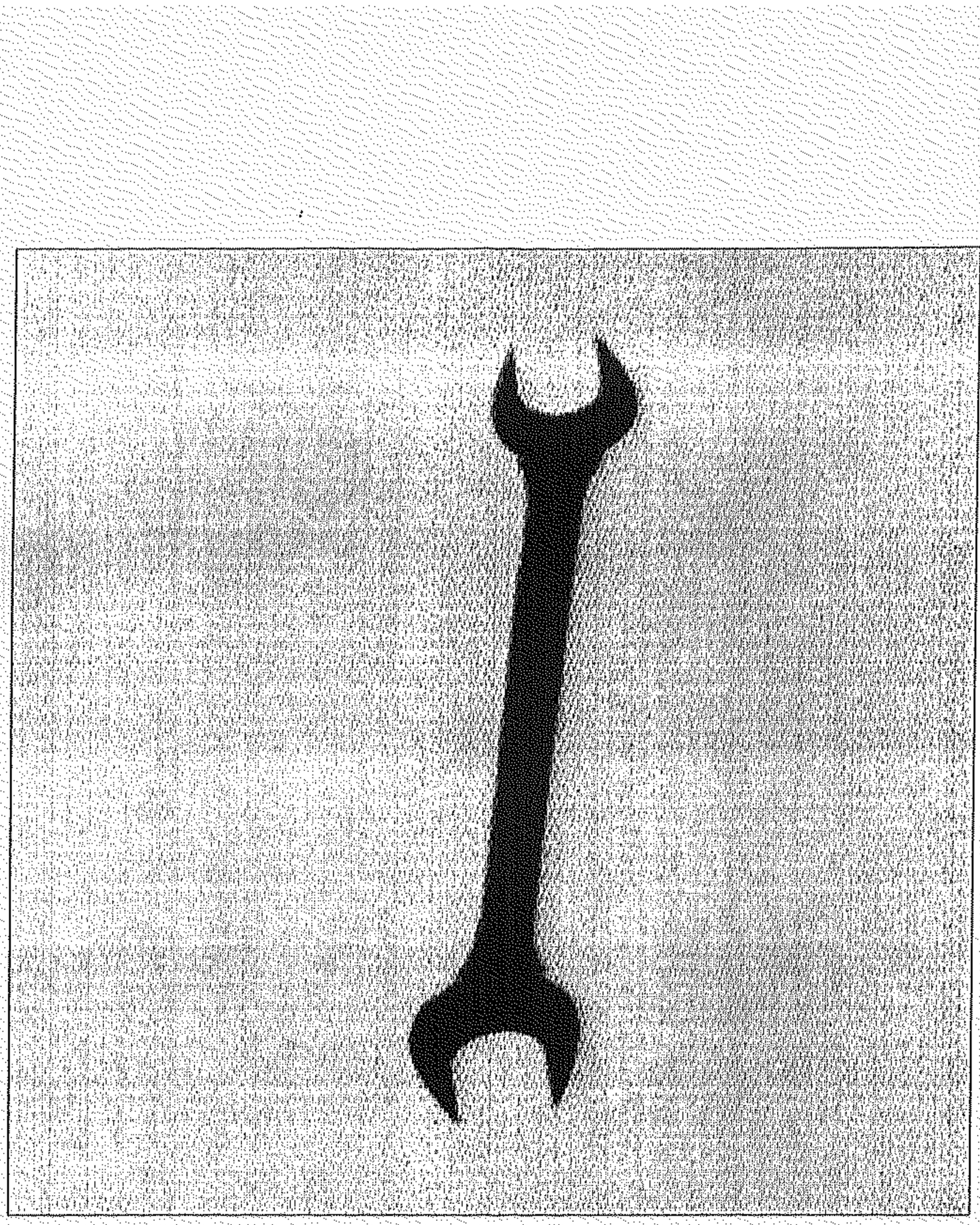
(d)

Figure 7.13 cont'd.



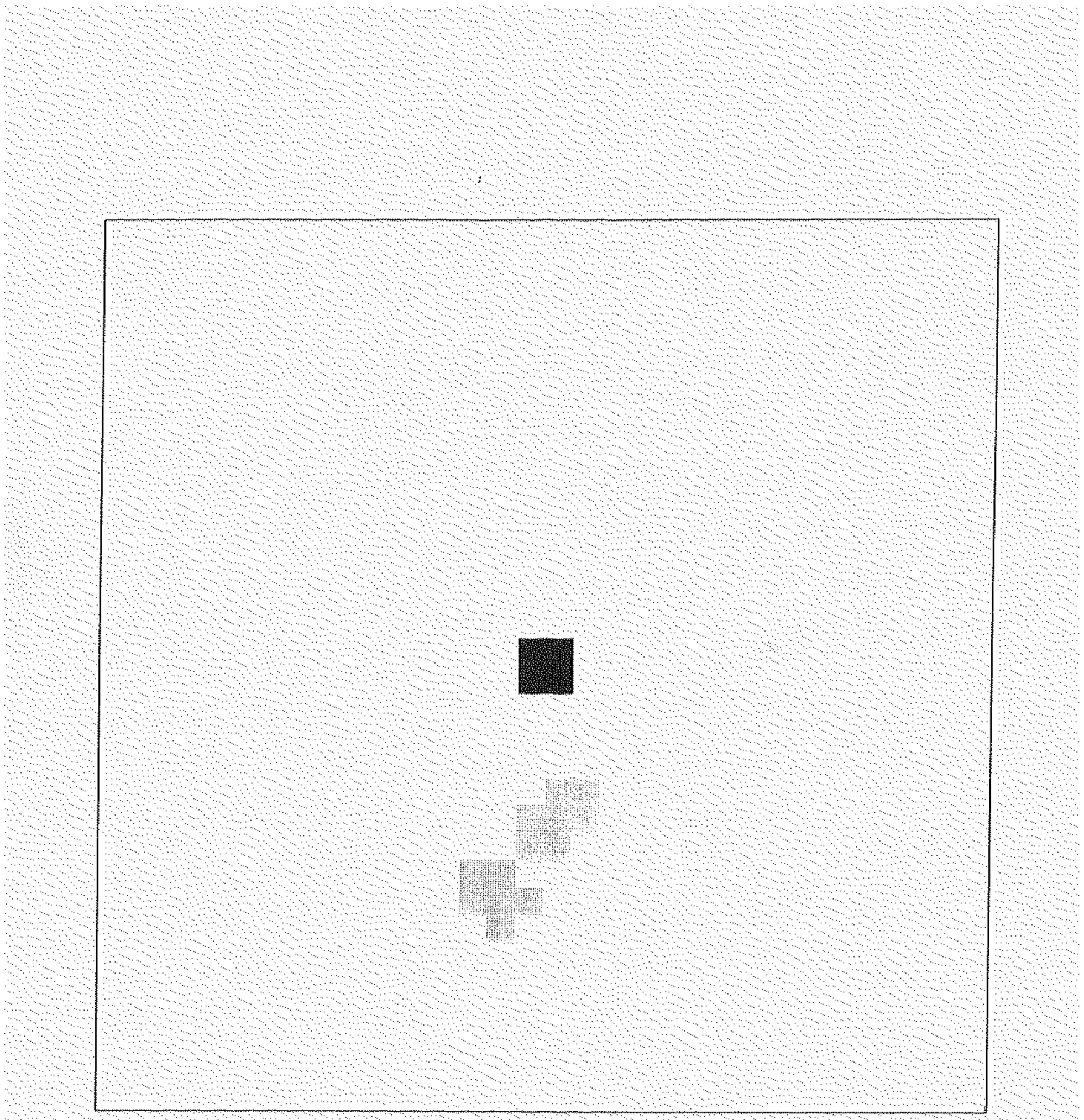
(e)

Figure 7.13: (a) Image of a hammer. (b) Feature map of the image. (c) Activations in the output layer (B-cells) after initialization process. (d) Activations in the output layer (B-cells) after 50 iterations. (e) Activations in the output layer (B-cells) after 300 iterations.



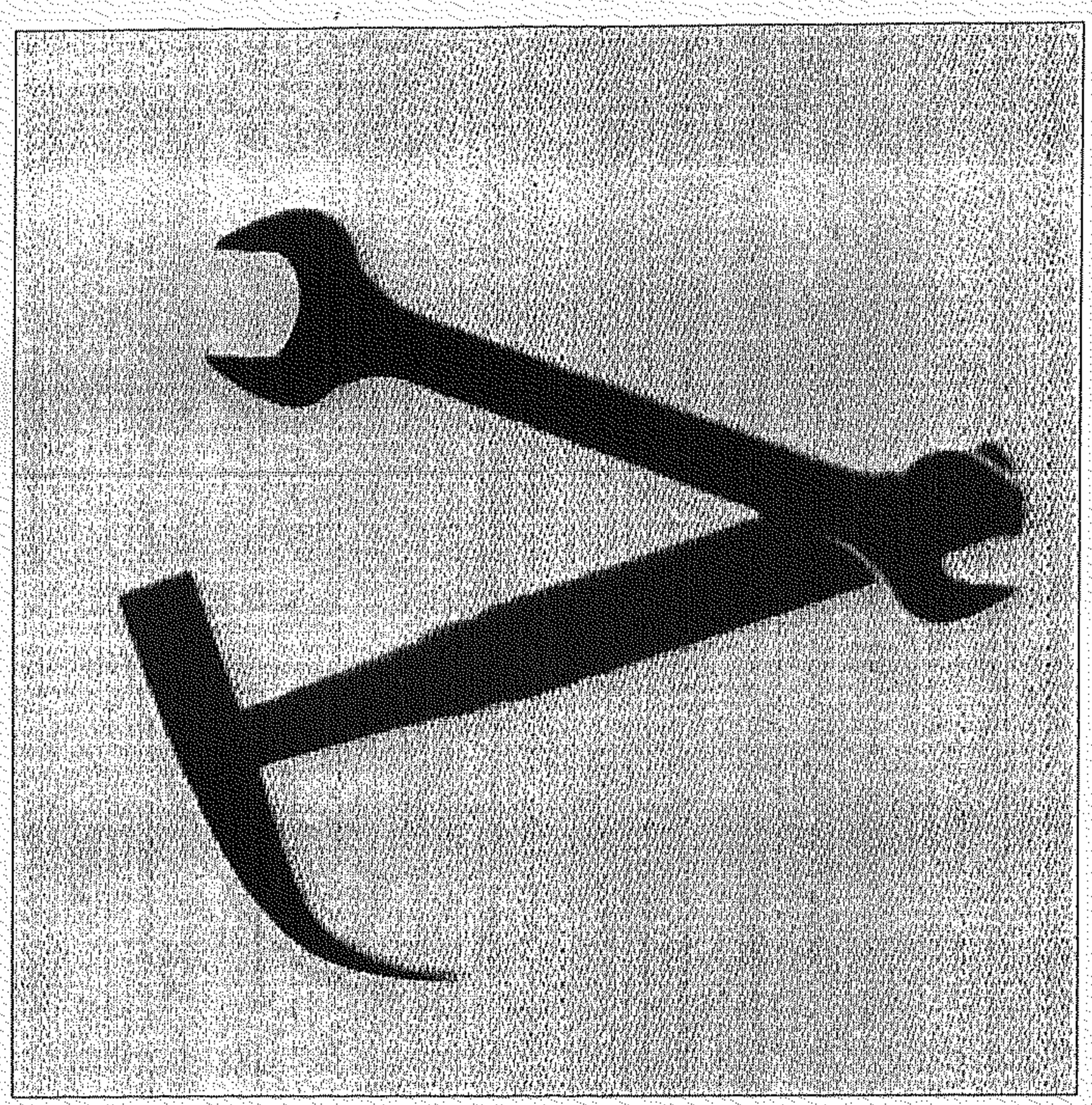
(a)

Figure 7.14 cont'd.



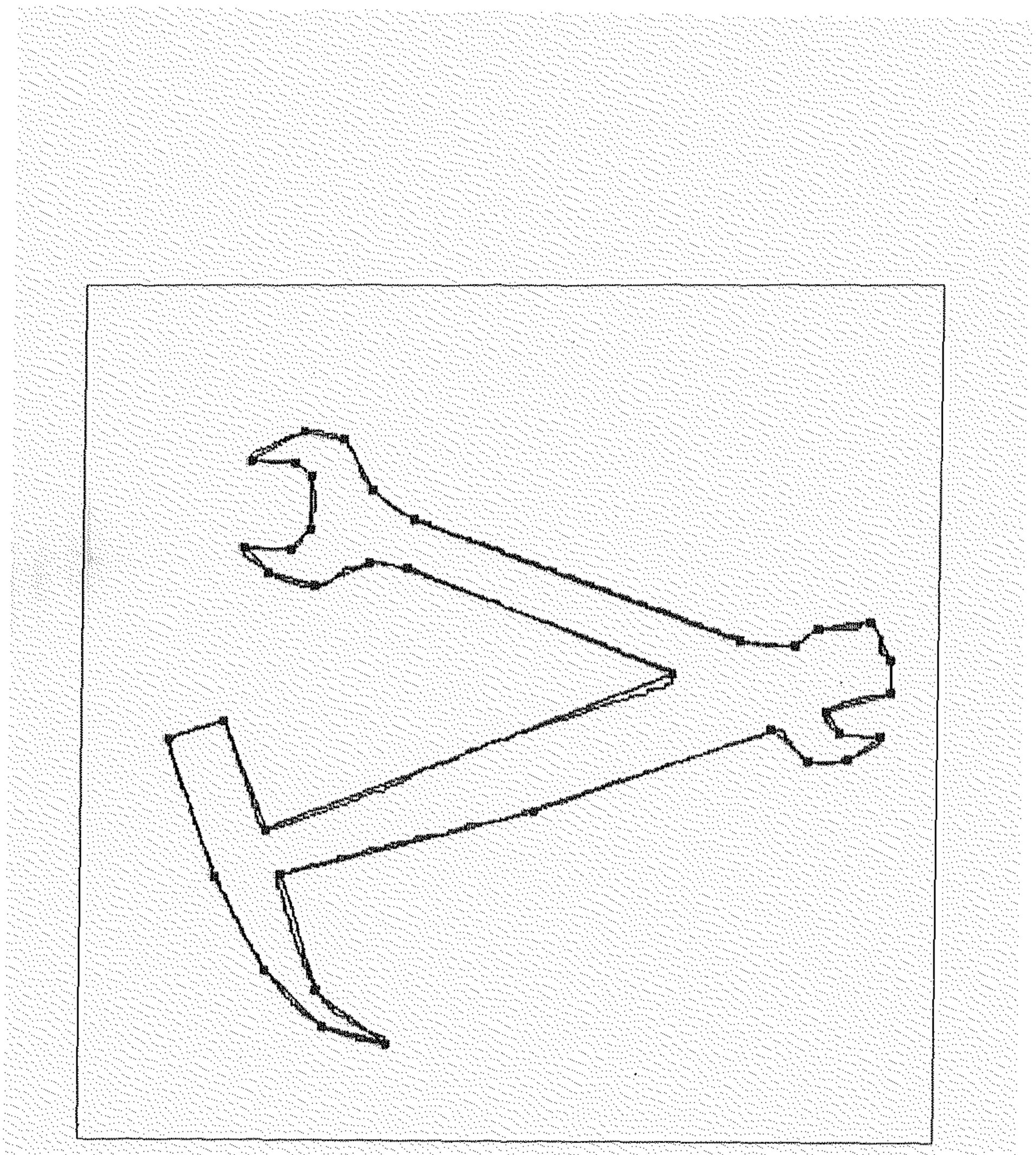
(b)

Figure 7.14: (a) Image of a spanner. (b) Activations in the output layer (B-cells) after 300 iterations.



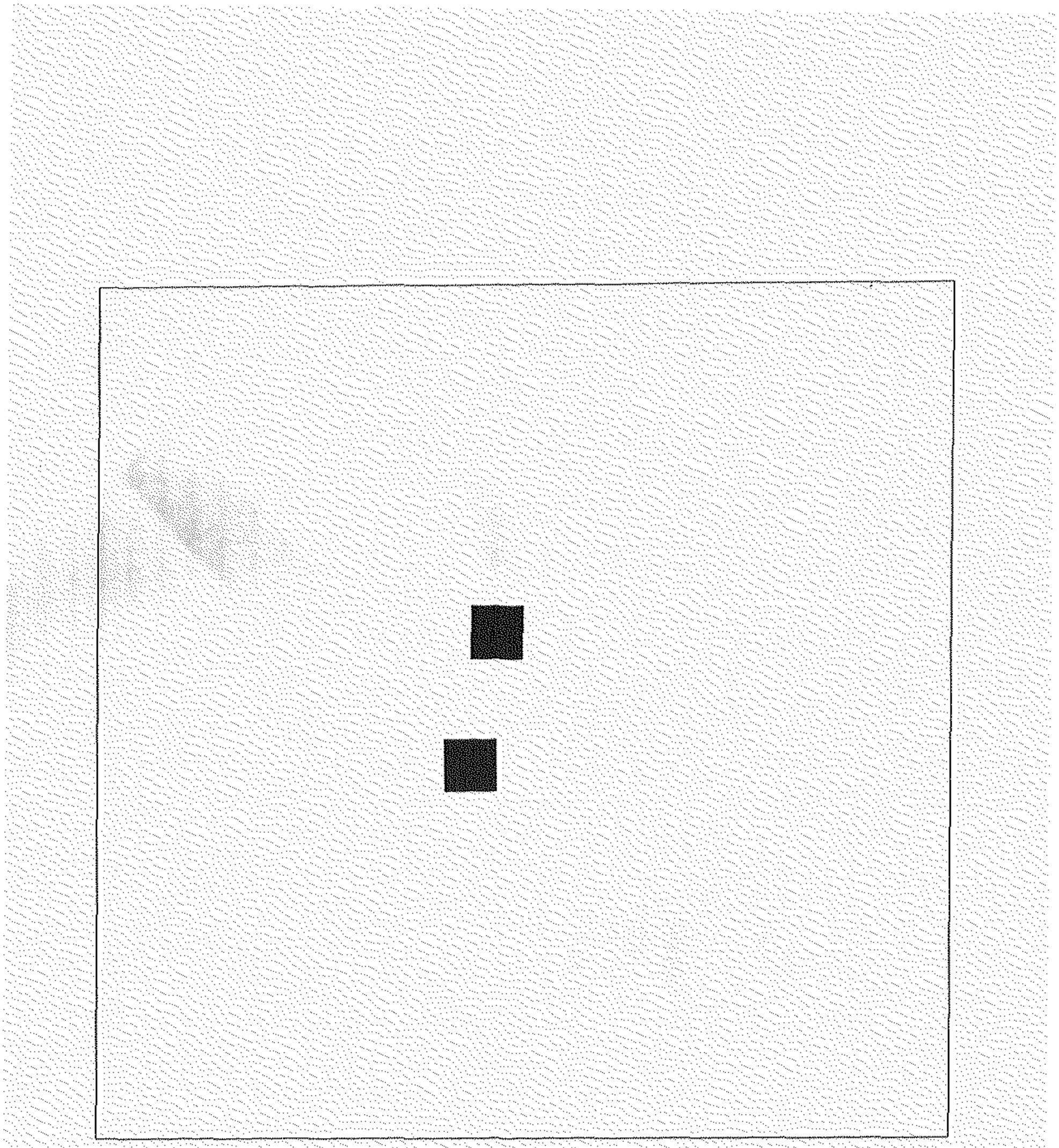
(a)

Figure 7.15 cont'd.



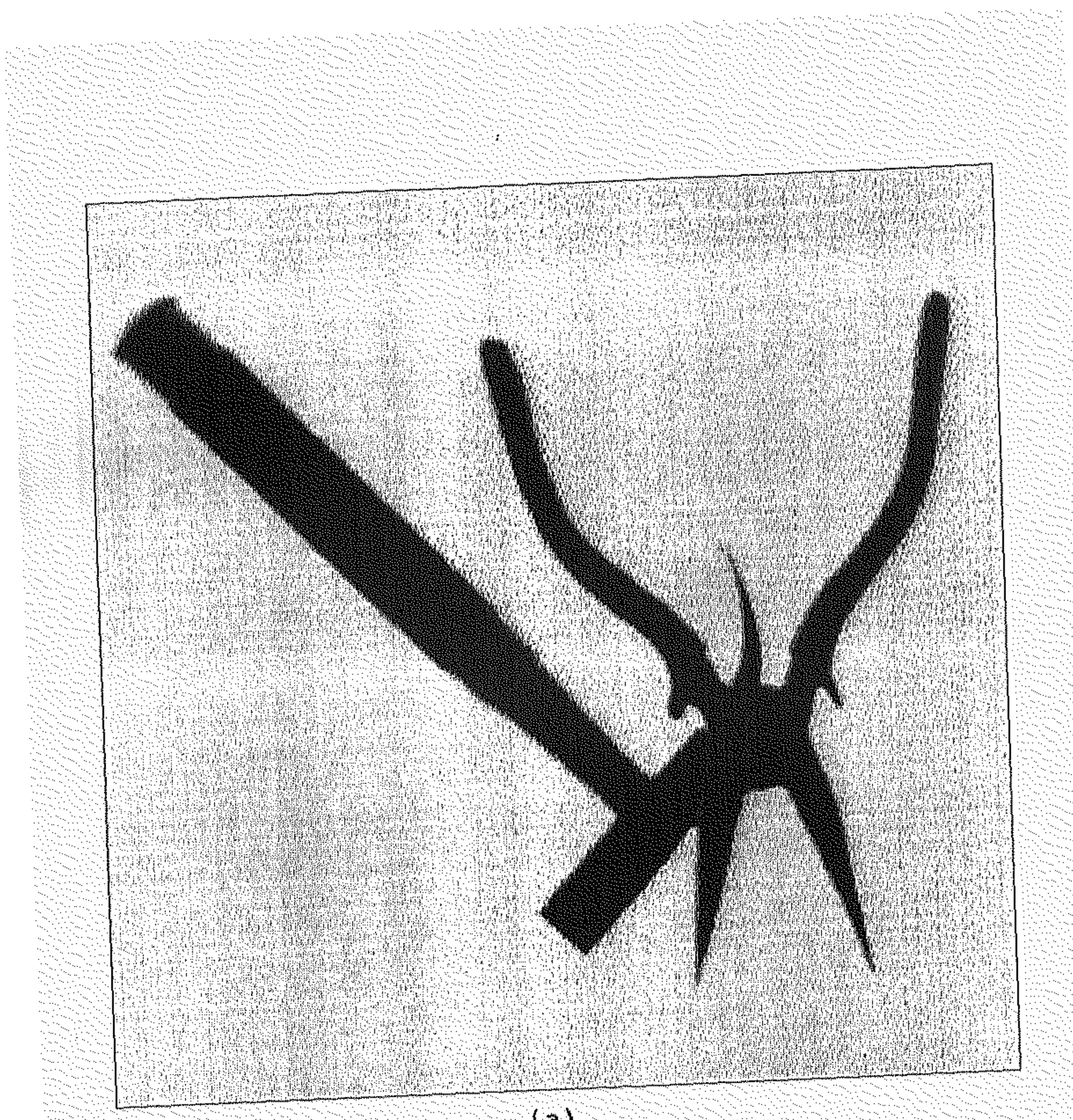
(b)

Figure 7.15 cont'd.



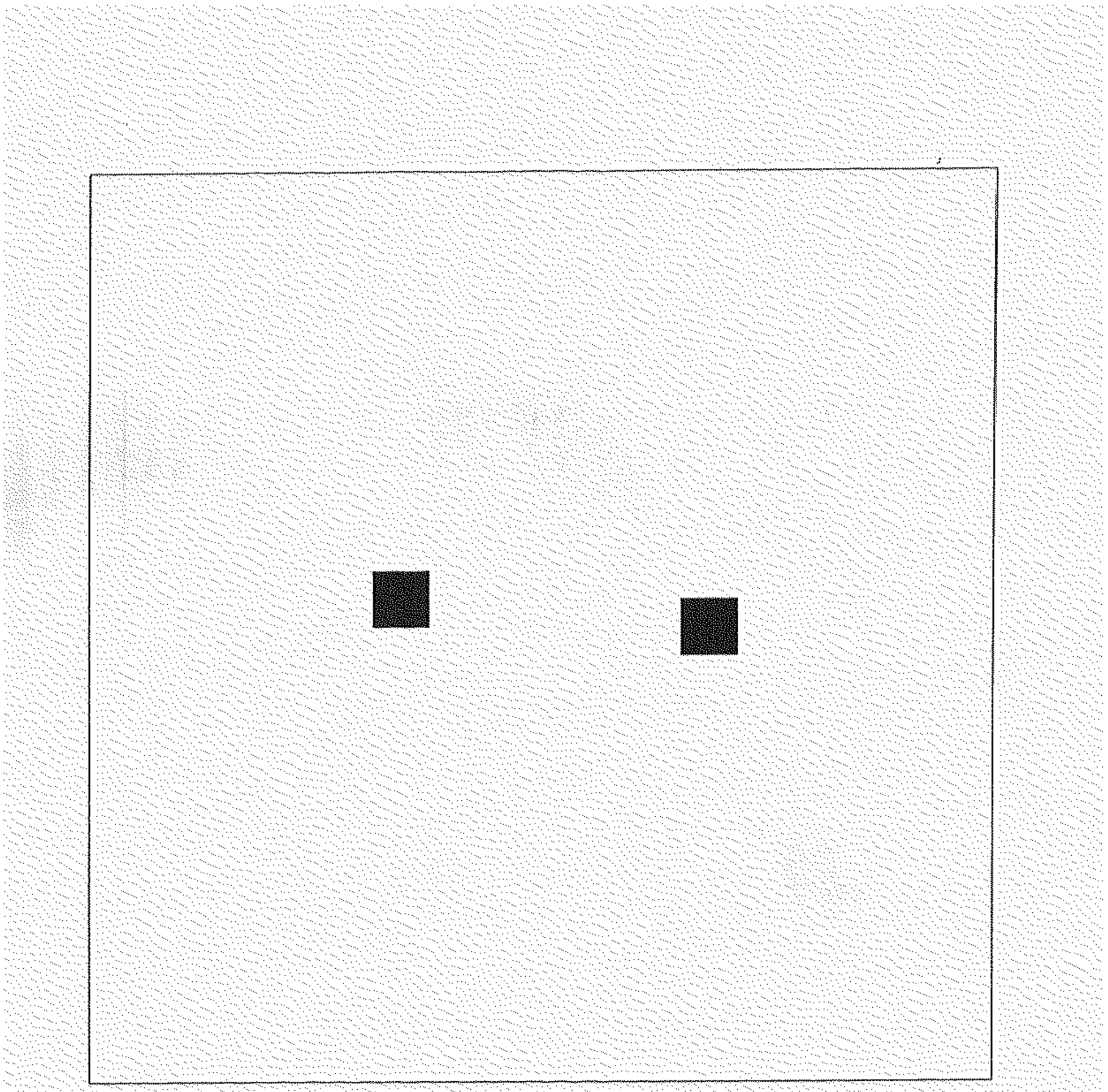
(c)

Figure 7.15: (a) Image of a hammer and a spanner overlapping each other. (b) Feature map of the image. (c) Activations in the output layer (B-cells) after 300 iterations. Note that the activation values have been thresholded. The threshold is selected as 0.3.



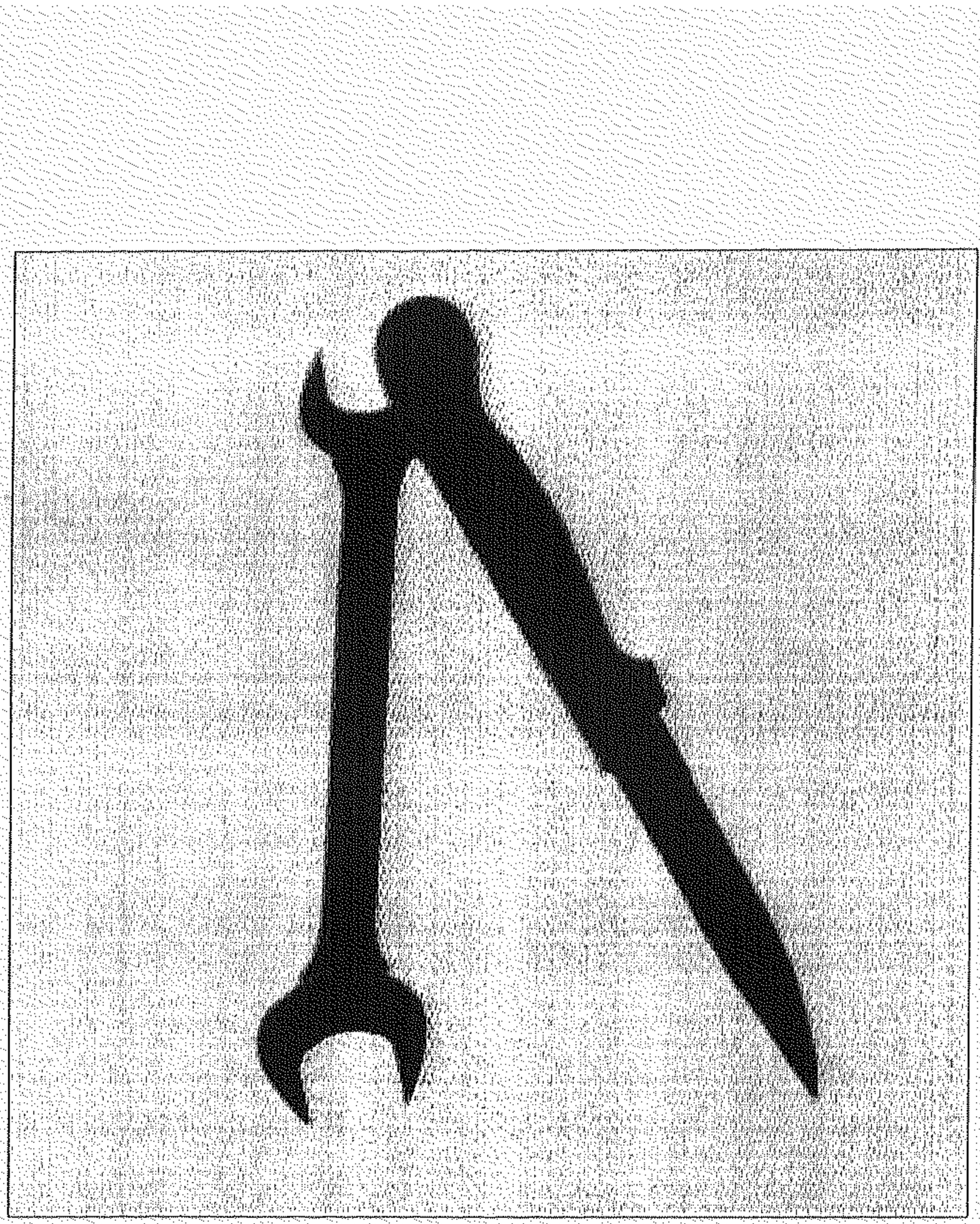
(a)

Figure 7.16 cont'd.



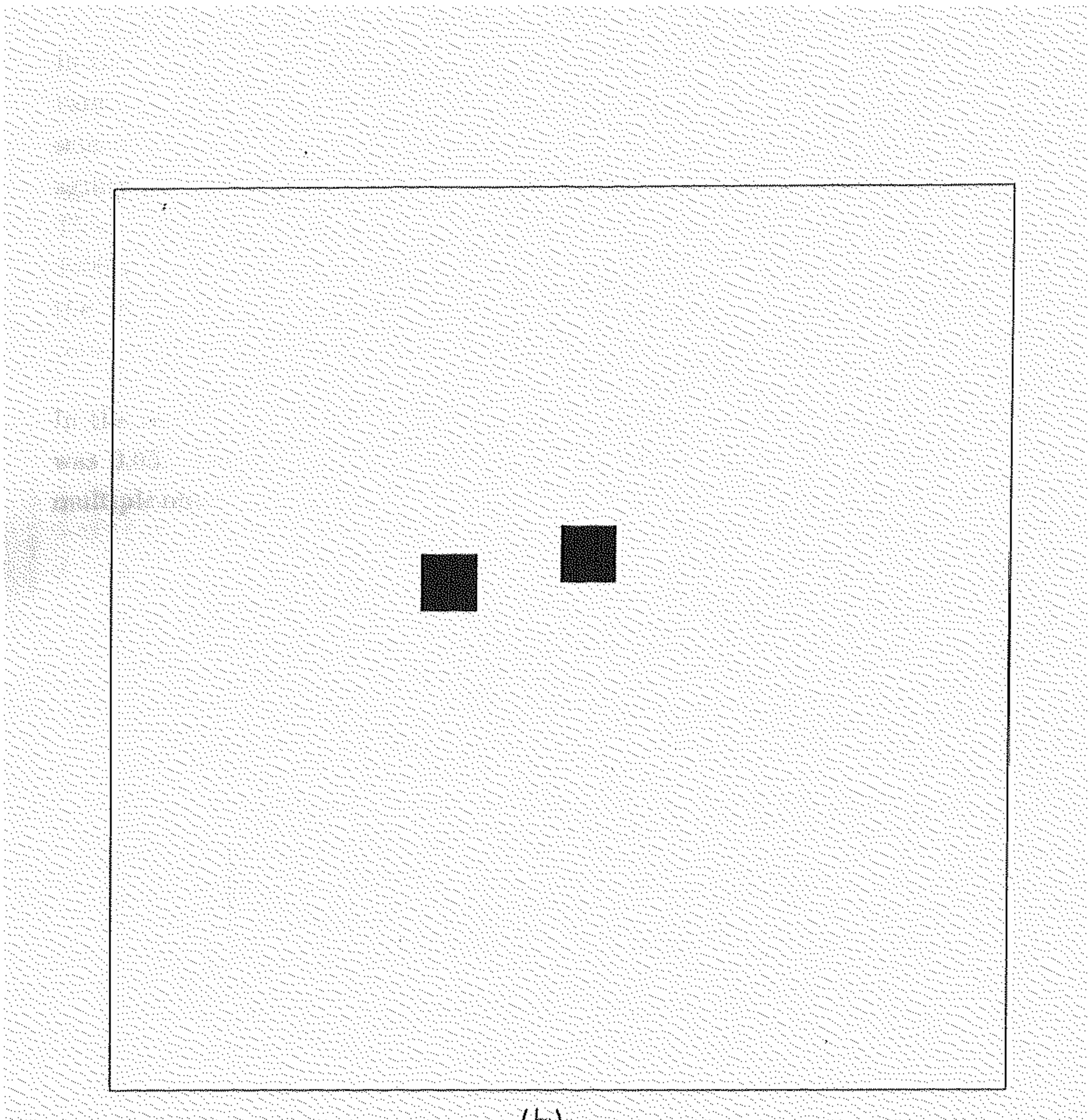
(b)

Figure 7.16: (a) Image of a hammer and a plier overlapping each other. (b) Activations in the output layer (B-cells) after 300 iterations. Note that the activation values have been thresholded. The threshold is selected as 0.3.



(a)

Figure 7.17 cont'd.



(b)

Figure 7.17: (a) Image of a spanner and a knife overlapping each other. (b) Activations in the output layer (B-cells) after 300 iterations. Note that the activation values have been thresholded. The threshold is selected as 0.3.

In the present case four different instances of each object have been presented. The value of γ has been taken as 0.15. The time step for each learning trial has been selected as 0.05. After every 200 iterations the nodes have been flushed, i.e., the agility factors (Section 7.5) of all the nodes were set to unity after every 200 trials. This causes the network to revive the learning capability for new situations. After 3000 trials the weights were found to change their value by less than 0.0005, and the training phase was terminated. After 3000 trials the network was found to consist of 288 type 1 links from the hidden layer to the output layer.

In the recognition process the time step was considered as 0.1. The self-feedback was 0.05. The Figs. 7.13, 7.14, 7.15, 7.16, 7.17 show the results for single and multiple objects for different number of iterations. The Table 7.1 shows the results

Table 7.1: outputs of the system after 300 iterations.

case	object	output	X	Y	θ
1 (Fig.7.13)	1	1.0	256.0	288.0	72.0
	2	0.0	-	-	-
	3	0.0	-	-	-
	4	0.0	-	-	-
2 (Fig.7.14)	1	0.0	-	-	-
	2	1.0	256.0	256.0	168.0
	3	0.0	-	-	-
	4	0.0	-	-	-
3 (Fig.7.15)	1	1.0	288.0	224.0	264.0
	2	1.0	208.0	240.0	60.0
	3	0.0	-	-	-
	4	0.0	-	-	-
4 (Fig.7.16)	1	1.0	240.0	176.0	36.0
	2	0.0	-	-	-
	3	1.0	256.0	352.0	24.0
	4	0.0	-	-	-
5 (Fig.7.17)	1	0.0	-	-	-
	2	1.0	224.0	192.0	336.0
	3	0.0	-	-	-
	4	1.0	208.0	272.0	12.0

of the five different instances of the objects (both single) and mixed or overlapped) after 300 iterations. Case 1 and 2 correspond to the single instances of objects presented in Figs. 7.13, 7.14. Cases 3, 4 and 5 correspond to the overlapped instances presented in Figs. 7.15, 7.16 and 7.17. In Table 7.1 (X, Y) represent the position, θ represents the orientation of the objects.

7.6.5 Number of Nodes

In the input layer 40 A-cells are used to represent the 40 slots of corners (or features). The output layer has 4 A-cells to represent the output objects. The number of A-cells in the hidden layer depends on the type of objects presented. If there exists sufficient overlap between the patterns of the objects then the number of hidden nodes will decrease. As discussed in Section 4.5, the number of hidden A-cells is approximately proportional to $m + n$ where m is the number of objects and n is the number of features, for sufficiently large number of objects (Equn.(4.37)). The input layer contains a B-cell in each grid location, i.e., $64 \times 64 \times 60$ B-cells or approx. 2^{18} B-cells. The hidden layer also contains a B-cell in each grid and the total number of B-cells in hidden layer is also 2^{18} approximately. The output layer uses the structure shown in Fig.7.10(a). The number of B-cells in the output layer is therefore $16 \times 16 \times 15 + 15 \times 15 \times 14$ i.e., 2^{13} approximately. The required number of cells is therefore 2^{19} approximately, i.e., 5×10^5 approximately.

7.7 Robustness of the Scheme

7.7.1 Noise Tolerance and Stability

The connectionist system presented here can take care of the noise present in the image level. Due to the presence of noise, a corner can change its curvature value from one slot to another, and as a result, the output of the corresponding object may degrade. It is intuitive that if a feature exhibits its variation during the training phase then the network should be able to capture the variations without much affecting the output in the recognition process.

To mathematically model the noise redundancy, a feature is considered to have a

distribution around its mean value. A feature (at any instant) can be represented as $\delta(c - c_i)$ in the analog domain, where c_i is the mean value of the i^{th} feature. The function $\delta(.)$ is the Dirac-delta function (note that, the noise redundancy should ideally be treated in discrete domain, but for the sake of simplicity, we have considered it in the analog domain). Here, the value of the feature should not be confused with the confidence value of the feature. The feature value indicates which particular slot has been fired. The features (e.g., corners in the present model) are encoded in such a way that if the angle of the corner changes the feature value will change. The shift is dependent on the amount of change in angle.

The convergence proof of the learning algorithms, as described in Section 4.4.3, indicates that the weights of the top-down links pick up the distribution of the features. The weights of the top-down link for the i^{th} feature[3] can be represented as

$$z1_i(c, x, y, \xi) = p_i(c) \tag{7.17}$$

The second subscript is omitted to represent any arbitrary object (the distribution for position and orientation is not written because we are not considering any variation in the position and orientation). Since according to the learning rules the weights of the bottom-up links are proportional to that of the top-down links (equation (7.11)), the distribution of the weights of the bottom-up links can be represented as

$$w1_i(c) = w_i p_i(c) \tag{7.18}$$

The output under stability is given as

$$w_s o = \sum_i \int_c w1_i(T_i C_i - z1_i o)dc \tag{7.19}$$

where, o is used as the output (instead of the symbol vb). vb is the actual output where each output B-cell has a nonlinear transfer function. For the sake of simplicity we have assumed it to be linear, and a different notation is used (output of the corresponding A-cell is considered to be unity). Here we consider that the activated hidden nodes are within the purview of the output cell (i.e., $w2$ and $z2$ are unity for all features). The transformational matrix is represented as T_i (obtained with the help of $W1$). C_i represents the i^{th} feature alongwith its coordinates (x, y, ξ)[4]. Since we are not considering the effect of positional and orientational

[3] $z1$ values pick up the feature value c, position (x, y), and orientation ξ

[4] The integration is performed to represent that the feature values are distributed in analog domain.

variance, T_iC_i can be represented as

$$C_i = \delta(c - c_i)$$

By algebraic manipulation, the output can be written as

$$o = \frac{\sum_i w_i \int_c \delta(c - c_i) p_i(c) dc}{w_s + \sum_i w_i \int_c p_i^2(c) dc} \tag{7.20}$$

Considering the nonoverlapping distribution for each feature and a Gaussian distribution for the weights of the top-down links corresponding to each feature, $p_i(c)$ can be written as

$$p_i(c) = \frac{1}{\sqrt{2\pi}\sigma_c} \exp\left(-\frac{(c - c_i)^2}{2\sigma_c^2}\right) \tag{7.21}$$

where σ_c is the variance of the feature value around the mean c_i for the i^{th} feature under noisy environment.

Under noiseless, ideal situation the feature values are expected to be the same as their mean values, i.e., the i^{th} feature for a particular object will have the feature value c_i. Therefore under noiseless condition the output of the desired object would be

$$o = \frac{\frac{\sum_i w_i}{\sqrt{2\pi}\sigma_c}}{w_s + \frac{\sum_i w_i}{2\sqrt{\pi}\sigma_c}} \tag{7.22}$$

If the value of w_s is small enough compared to the total weights of the bottom-up links coming to a particular object then the output becomes

$$o = \sqrt{2}$$

Since the transfer function of each neuron is such that the output saturates and cannot go beyond unity, the output under noiseless condition will ideally saturate to unity.

Let the object be such that the features do not coincide with the mean values, and let the i^{th} feature in the transformed space be given as

$$C_i = \delta(c - c_i - \Delta c) \tag{7.23}$$

In that case, from equn. (7.20) the output of the desired object would be

$$o = \sqrt{2}\left(1 - \frac{w_i}{\sum_i w_i}\left(1 - \exp(-\frac{\Delta c^2}{2\sigma_c^2})\right)\right) \tag{7.24}$$

Here, the effect of w_s has been neglected. Since the output saturates at unity, the effect of the shift will be perceived at the output depending on the shift in c. The mathematical treatment basically reveals the fact that the noise degradation depends on the distribution of the feature during the learning process. If the feature suffers wide variation in the learning process (i.e., large σ_c), then the variation of that feature in the recognition process (i.e., Δc) does not cause much degradation in the output. In other words, no single instance of the feature is given great importance in the recognition process. On the other hand, if the feature does not suffer much deviation (i.e., small σ_c) in the learning trials, then in the recognition process if the feature suffers deviation (i.e., large Δc) then the output will be deteriorated.

However, in this mathematical treatment the confidence of a feature at any point has been modeled using Dirac-delta function which is not true in real life. Therefore, the noise degradation may not be so smooth as presented here.

7.7.2 Crosstalk and False Alarming

It was mentioned before that the links are stimulated by the modifiers attached to them. The modifiers will stimulate or deactivate certain links only when the signal coming through the modifier link is sufficiently high i.e. greater than certain threshold (ϵ). Let a particular combination of A and B cells be active in the input layer (say Ai1 and Bi1) and the type 12 link from the coalition of Ai1 and Bi1 stimulate the links connected to the coalition of Ah1, Bh1 in the hidden layer. Since the signals carried by the modifier links are sufficiently high over a neighborhood centered around Bh1, the links emanating from the coalitions of Ah1 and the neighborhood cells of Bh1 would also be stimulated. Therefore, even if the competition takes place over a neighborhood of Bh1, and only Bh1 wins, the links from the neighborhood of Bh1 remain stimulated. If the feature is such that it instantiates another feature-object pair in the neighborhood of Bh1 then a false coalition may be formed.

To prevent such crosstalk, the links are provided with a special attenuable LTM (ALTM). Initially all the links are not capable of carrying activations. If the links are modified by some stimulating modifier and the cells connected to the links are active then only the links retain their signal carrying capability. If the links are

not modified or do not have any active cells connected to them then they lose their capability of carrying the activations. Mathematically, the decaying property is given as,

$$\frac{dw}{dt} = -\frac{w}{\tau_w} \tag{7.25}$$

The time constant τ_w can be selected depending on the rate of change of the zone of attention controlled by the ACN.

7.8 Conclusions and Discussion

A scheme for polygonal object recognition using the principles of connectionist computation has been presented here. Several concepts motivated by the psychological findings have been used. The main contribution of this model lies in the introduction of the concept of separating the networks for identification and pose estimation (locating) of objects. This very fact helps in reducing the total number of neurons to a great extent as compared to the other models [188], [189]. Nonetheless, there is not much gain as far as the number of links is concerned.

The model also uses the theory of selective attention for the initial hypotheses formation. The attention mechanism used here is *early selection* process. This pseudo-parallel mechanism, used in the initial hypotheses formation, basically helps in the simultaneous recognition of multiple objects (both single and overlapped instances).

The learning rules are able to quantify the relative importance values of the features with respect to objects. The system is also designed to learn the transformation values from the features to the objects and is proved to be tolerant of variations in feature values.

The analysis of robustness of the model shows a graceful degradation in its performance when a feature suffers noise in its value, position or orientation. The experimental results also support the fact that even under occlusion or variation of the features in their position, orientation or value, the system is able to recognize the objects present in the scene.

Chapter 8

Conclusions and Scope of Further Research

8.1 Conclusions

In this thesis, we have presented some results of investigation demonstrating the effectiveness of connectionist approaches to deal with the tasks of edge/line linking and feature interpretation. Several new application-specific models are developed for edge/line linking, mixed category perception (both supervised and unsupervised), simultaneous extraction of multiple primitives from an image, learning and recognition (identification and localization) of objects. Two basic models (Hopfield and multilayer perceptron) are also used while developing the methods for structural learning and recognition. These methods follow the basic principles of connectionist computation, although the concepts behind them have their root in classical techniques and their implementation sometimes consider the findings in psychology and neurobiology. The effectiveness of these methods has been demonstrated on both synthetic and real-life data.

The task of edge/line linking has been dealt within connectionist framework by employing cooperative and competitive processes between the neighboring neurons (Chapter 2). In the model, developed for line linking, the neurons interact over eight neighborhood while the model for edge linking employs interactions between the neurons over larger (circular) neighborhoods considering the process of edge

induction. Two different state updating rules have been used for line linking in graylevel images. From the point of linking line segments, the first updating rule (Equn.(2.4)) performs better than the second updating rule (*Equn.(2.10)*), while the noise reduction capability of the second rule is better than the first one.

Note that, the method for line linking can also be used for linking edge segments considering the gradient images (Section 2.5). However, as far as the linking of edge segments is concerned, the method developed on the basis of edge induction (Section 2.4) performs better than the method of line linking (Section 2.3), although the effect of smoothing and dislocation of edge segments is more in the edge induction process. It is also to be noted that lengthening of open ended edge segments and reduction of strengths at the junction points may occur (as mentioned in Section 2.6) in the edge induction process.

Although the methods developed for both edge and line linking are conceptually analogous to the existing relaxation labeling techniques or edge diffusion techniques [67], [68], [71], they provide a new direction of implementing the cooperative processes onto neural networks. Moreover, the methods demonstrate a way how the geometric constraints for edge/line linking can be incorporated into neural network models and the boundary filling-in process [153], [155] can be realized in a better way within a connectionist framework. In other words, the investigation is a step towards bridging the gap between classical image processing techniques and the findings in cognitive psychology.

The study on feature set interpretation has been conducted in two parts. In the first part, the characteristics of Hopfield model as a constraint optimization problem solver, has been exploited for formulating a method for matching relational descriptions of candidate structures with prototype structures (Chapter 3). Although, Hopfield network consists of only symmetric interconnection links, the asymmetric relational constraints (required for more general descriptions) are mapped onto the network with the help of a suitable graph-theoretic transformation. As a result, the method can be applied for general structural description matching problems involving realistic asymmetric spatial relations like *right*, *left*, *top*, *bottom* etc., unlike the other relevant techniques [161], [162], [164]. The system is able to identify the best match for a distorted candidate structure from a set of prototype structures when descriptions of handtools and characters are considered as input.

In the second part of the study on feature set interpretation, several application-

specific models are developed (Chapters 4–7) for the purpose of mixed category perception, simultaneous extraction of multiple primitives, and identification and localization of multiple objects. X-tron (Chapters 4 and 5), a three-layered model, employs an iterative verification process to interpret overlapped set of features corresponding to the mixed categories (under both supervised and unsupervised modes). Special learning rules are also designed for X-tron, based on some predetermined probabilistic measures. Categorization principle of X-tron is developed on the basis of some ambiguity measure. X-tron is able to categorize both single and mixed patterns when presented in the form of numerical feature vectors. The functioning of X-tron is somewhat similar to ART [130], [132], [133], although no order search mechanism, as performed in ART, is necessary in X-tron for categorization. Moreover, unlike ART, X-tron is able to categorize in the presence of overlapped patterns. It is also different from the existing methods [142], [144], [145], [146], [149], [150], [152], [147], [148] for mixed category perception. X-tron is able to learn in the unsupervised mode unlike the model proposed by Cho and Reggia [142]. It is able to form stable category codes unlike the model proposed by Cohen and Grossberg [147], [148]. It performs the mixed category perception by dissociating the competition process between the categories, from the output layer unlike the model developed by Marshall [144], [145], [146] or SONNET developed by Nigrin [149], [150], [152]. It is to be noted here that X-tron is not able to categorize embedded patterns which is possible in SONNET, but X-tron is able to deal with a high degree of overlap between the mixed patterns which may not be possible in [144], [145], [146], [149], [150], [152].

The learning capability of X-tron is also comparable with the existing models. ART [130] employs both fast and slow learning paradigms while in X-tron, the speed of learning depends on the agility factors of the nodes which are automatically adjusted in the process of learning. X-tron can also automatically adjust the number of nodes during the categorization process. Its categorization performance gracefully degrades with the level of noise (additive, subtractive or mixed form).

Although the underlying principle of X-tron provides a basis for mixed category perception, the model itself is not able to deal with the real-life images. However, the principle can be employed with suitable modifications for both primitive extraction and object recognition from real-life images. This is demonstrated in Chapters 6 and 7.

In Chapter 6, the principles of X-tron and line linking (Chapter 2) are integrated

so that multiple primitives (linear segments) can be simultaneously extracted from an image. This integration provides an efficient technique for simultaneous extraction of multiple peaks in Hough space without going through any explicit threshold selection scheme. An MLP model with properly chosen architecture, in conjunction with this integrated model, is found to learn and recognize distorted versions of handwritten Bengali characters of similar shapes. This provides an indication about the robustness of the extracted (structural) primitives and also the ability of MLP based system in discriminating between the structures of nearly similar shapes. Moreover, the MLP model, in this method, is able to learn and recognize the structures independent of their size; only the parameters of the layer corresponding to the Hough space are to be properly adjusted when the sizes of the input structures are changed.

Principle of X-tron has been finally employed for the development of PsyCOP (Chapter 7), a system for learning, and orientation and shift invariant recognition of multiple objects. PsyCOP is able to learn and recognize flat, rigid objects like hammer, spanner, plier etc. Unlike the other existing connectionist models (as mentioned in Chapter 1) for object recognition, PsyCOP employs two separate channels for identification and localization of objects which is also supported by one of the major psychological finding that – identification and localization occur in two different zones of visual cortex. It also uses selective attention mechanism (similar to MORSEL [187], [186]) for initial instantiation of the nodes.

PsyCOP is able to recognize multiple objects simultaneously which is not possible in other models including neocognitron [171], [175], [177] (except MORSEL). However, in MORSEL, orientation invariant object recognition is not possible. Also, unlike PsyCOP, the structural relationships between the features and objects were not considered in MORSEL or neocognitron, which may be essential for the recognition of some general kind of structures (apart from the characters) and for the incorporation of orientation invariance. Although, the structural relationships between the features and the objects were considered in TRAFFIC [188], [189], the problem of simultaneous recognition of multiple objects was not considered. Moreover, in TRAFFIC, the localization problem was dealt with by keeping nodes for each object at each location. This would require larger number of nodes as compared to PsyCOP for the recognition of the same set of objects. Note that, PsyCOP accepts the corner features as input, and cannot accept the pixel level information directly (unlike neocognitron [171], [175], [177] or MORSEL [187], [186]).

It has been theoretically investigated that the performance of PsyCOP gracefully degrades with the level of noise.

8.2 Scope of Further Research

The model for linking line segments operates over eight neighborhood, while the model for linking edge segments operates over extended circular neighborhoods. The functioning of the model for edge linking may be enhanced by considering elliptical neighborhoods. The major axis of the ellipse should lie in a direction along the edge (i.e. orthogonal to the direction of the edge segment). This would enhance the cooperative process over a larger region and reduce the competitive process over a smaller region. It would also help the model to link distant edge segments with similar alignments and at the same time protect a segment to some extent from getting adversely affected by other segments with different alignments.

At the junction points of two different edge segments (for example, a corner of a box in a chess board), the edge directions may be totally opposite and therefore, the edges may get disconnected at the junctions. A scheme may be devised to protect the junction points in the linking process based on the a priori knowledge of the junction points. In other words, the linking process can be executed in conjunction with a separate featural level process. This may also be incorporated with a feedback from higher level processes, in a manner described in Chapter 6. The same cooperative and competitive processes can also be used to extract higher level entities by grouping the edge/line segments. Such connectionist models would then provide a better realization of the region filling-in and the perceptual grouping processes as described by Grossberg and Mingolla [153], [155].

The linguistic shape descriptions (e.g., small, medium, large, linear, circular etc.) which have been used for structure matching (Chapter 2), can be quantified using fuzzy set theory [238], [82] which have better efficacy in dealing with uncertainty arising from descriptions. The relational descriptions like left, right, top, bottom etc. can also be quantified/represented within this framework.

X-tron (described in Chapters 4 and 5) may be extended to a hierarchical model for better representation of the feature-object relationships. The intermediate nodes in that case, would represent the macrofeatures of subparts present in the objects.

One way of forming the hierarchical model is to search for the group of features which occur, as a whole, in different instances of the objects and then treating these group features as macrofeatures, intermediate nodes can be allocated for them. The process of formation of such a hierarchical model may also be substantiated by the concept of part-whole hierarchy as described by Hinton [185].

X-tron is able to categorize patterns even when they appear in mixed form. However, sometimes it may be necessary to form stable categories for certain patterns which are embedded in some other patterns. Development of a model based on the principle of X-tron for categorizing embedded patterns therefore constitutes a part of further investigation.

The principle of X-tron has been integrated with that of line linking for the extraction of line primitives. It may also be extended to extract other form of primitives (e.g., arc) of known parametric form. Moreover, with the help of feedback links of X-tron and employing cascaded modules, complex primitives including the junction points may be extracted out.

PsyCOP identifies and localizes the objects with orientation and shift invariance. However, the scale invariance may also be effectively incorporated by associating an approximate scale value depending on the distances of the other features from that particular feature. An input node corresponding to a feature would then activate a group of hidden nodes over a certain range of scales, and a cluster of activations would form in the hidden layer for some particular scale value. The scale invariance may also be efficiently achieved if a hierarchical model is used where the input features instantiate subparts and the subparts activate the objects in turn.

PsyCOP uses only the supervised mode of learning. Using a principle similar to that of X-tron, an unsupervised learning scheme may be devised to categorize the real-life objects even when they appear in mixed form. However, in the measure of certainty factor, the occluded features, feature locations etc. are also to be taken care of.

PsyCOP can also be extended to a hierarchical model. The nodes in the hierarchy, in that case, would represent the subparts of the objects. This may enable the system to recognize flexible objects, i.e., objects with movable subparts (like scissor). Here, the subparts can be found by extracting out the common subset of features in different instances of the same object. A separate strategy may be

designed to automatically build up the hierarchy depending on the nature of the objects. Although only corner features are encoded in the investigation, several other structural features (e.g., holes) may also be used for the recognition of objects which are even more complex in nature.

Note further that, edge/line linking and feature set interpretation require efficient search algorithms or optimization procedures. Genetic algorithms (GAs) [247], which are highly parallel, robust, and adaptive search techniques, may be employed for finding broken edge/line segments for proper linking, and for determining the required translation, orientation and scaling of the model templates in order to get the accurate match between candidate and prototype objects. In this connection, we mention the recent investigation [248] which combine the characteristics of GAs with that of stochastic gradient descent technique in order to deal jointly (synergistically) with the problems in low-, intermediate-, and high-level vision in presence of noisy images.

Appendix A

Proofs on Edge Point Induction

Here we are going to present the mathematical proofs of some of the properties of edge point induction (Chapter 2) [204], [205].

A.1 Induction due to a single edge vector

Let us consider $0 < \alpha \leq \pi/2$, and $0 < \phi \leq \pi/2$, where ϕ is the angle subtended by PQ with X-axis. P is the point whose effect on Q (due to edge induction) is currently being considered (Fig.2.5). The edge vector induced at Q is considered to be perpendicular to the direction of PQ (this is governed by the nature of the proposed edge point induction). Therefore, the induced edge $\mathbf{e}^{'}$ due to P will subtend an angle $\pi/2 + \phi$ with the X-axis. The induced edge strength is given as

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

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

$$e^{'} = e_y\cos\phi - e_x\sin\phi \tag{A.1}$$

The horizontal and vertical components of $\mathbf{e}^{'}$ are

$$e_x^{'} = e^{'}\cos(\frac{\pi}{2} + \phi)$$

$$= -e^{'} \sin\phi$$

and

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

respectively. Substituting the value of $e^{'}$ (from Equn.(A.1)) we get

$$\begin{aligned} e_x^{'} &= e_x \sin^2\phi - e_y \cos\phi \sin\phi \\ e_y^{'} &= e_y \cos^2\phi - e_x \cos\phi \sin\phi \end{aligned}$$

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

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

Therefore,

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

and

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

Substituting the value of $e^{'}$, the above expressions become

$$\begin{aligned} e_x^{'} &= e_x \sin^2\phi - e_y \sin\phi \cos\phi \\ e_y^{'} &= e_y \cos^2\phi - e_x \sin\phi \cos\phi \end{aligned}$$

The equations for $e_x^{'}$ and $e_y^{'}$ can be proved to be true in all quadrants for all possible values of α and ϕ.

A.2 Induction due to a straight edge

Consider any two edge points M, N on the edge AB, such that $MP = PN = x$. PP' is perpendicular to AB (Fig.2.6). In this case, the edges at P' induced by edge

points M and N subtend equal and opposite angles with PP', and as a result, an edge vector along PP' is induced (the edge strengths at M and N are considered to be equal). The strength of the edge induced at P due to edge point M is $e_1^{'} = e\cos\theta$ where θ is the angle of the induced edge with PP'. The edge induced by the edge point N has the same strength and subtends an angle $-\theta$ with PP'. Therefore, the resultant induced edge strength (due to M and N) is $e_{12}^{'} = 2e\cos^2\theta$.

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

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

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

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

A.3 Induction over an interval

From Appendix II it can be said that the increase in edge response at any point P' (Fig.2.6) in time dt is

$$de^{'} = 2ew\left(\sqrt{R^2-D^2} - D\tan^{-1}\left(\sqrt{(R/D)^2-1}\right)\right)dt \qquad \text{(A.2)}$$

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

$$e^{'} = \frac{2ew}{\gamma}\int_D^{R_0}\left(\sqrt{R^2-D^2} - D\tan^{-1}\left(\sqrt{(R/D)^2-1}\right)\right)dR \qquad \text{(A.3)}$$

After simplification the expression reduces to

$$e^{'} = \frac{2ew}{\gamma}\left[\frac{1}{2}R_0\sqrt{R_0^2 - D^2} + \frac{D^2}{2}\log(\frac{R_0}{D} + \sqrt{(R_0/D)^2 - 1}\right]$$
$$-R_0 D\tan^{-1}\left(\sqrt{(R_0/D)^2 - 1}\right)$$

The expression has been derived without considering the induction from other edges in the neighborhood and the self-inhibition. From the expression (Equn.)of *induced edge strength it is clear that* the induced edge strength will be more if P' is nearer to Ab and will be less if P' is further from AB. To avoid saturation of induced edge strength at any point in the neighborhood of an edge segment like AB, we must have $e^{'}$ to be less than unity for any position of P'. Since $e^{'}$ increases as D decreases, we must consider $R_0 >> D$. Then we have

$$e^{'} = \frac{2ew}{\gamma}\left[\frac{1}{2}R_0^2 + \frac{1}{2}D^2\log(2R_0/D) - R_0 D\tan^{-1}(R_0/D)\right] \qquad \text{(A.4)}$$

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

$$e^{'} = \frac{ewR_0^2}{\gamma} \qquad \text{(A.5)}$$

Since we require $e^{'}$ to be less than unity, we must have

$$\gamma > ewR_0^2 \qquad \text{(A.6)}$$

Bibliography

[1] P. Suetens, P. Fua, and A. Hanson, "Computational strategies for object recognition," *ACM Computing Surveys*, vol. 24, pp. 5–61, 1992.

[2] P. Besl and R. Jain, "Three-dimensional object recognition," *ACM Computing Surveys*, vol. 17, pp. 75–145, 1985.

[3] R. T. Chin and C. R. Dyer, "Model based recognition in robot vision," *ACM Computing Surveys*, vol. 18, pp. 69–108, 1986.

[4] A. Wallace, "A comparison of approaches to high-level image interpretation," *Pattern Recognition*, vol. 21, pp. 241–259, 1988.

[5] F. Zhao, "Machine recognition as representation and search – a survey," *Int. J. Patt. Recogn. Artificial Intell.*, vol. 5, pp. 715–747, 1991.

[6] M. M. Trivedi and A. Rosenfeld, "On making computers "see"," *IEEE Trans. Syst., Man, and Cybern.*, vol. 19, pp. 1333–1335, 1989.

[7] D. Marr, *Vision*. San Francisco, CA: W. H. Freeman, 1982.

[8] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Royal Society London B*, vol. 207, pp. 187–287, 1980.

[9] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. New York: Academic Press, 1982.

[10] R. C. Gonzalez and P. Wintz, *Digital Image Processing*. Massachusetts: Addison-Wesley, 1987.

[11] N. R. Pal and S. K. Pal, "A review of segmentation techniques," *Pattern Recognition*, vol. 26, pp. 1277–1294, 1993.

[12] D. Waltz, "Understanding line drawings of scenes with shadows," in *The Psychology of Computer Vision*, (P. H. Winston, ed.), pp. 19–91, New York: McGraw-Hill, 1975.

[13] M. A. Fichsler, J. M. Tenenbaum, and H. C. Wolf, "Detection of roads and linear structures in low resolution aerial imagery using a multisource knowledge integration technique," *Computer Vision, Graphics and Image Process.*, vol. 15, pp. 201–233, 1981.

[14] S. Chaudhury, *Development of methodologies for recognition of partially obscured shapes.* PhD thesis, Dept. of Computer Science and Engg., Indian Institute of Technology, Kharagpur, India., 1989.

[15] L. Shapiro and R. Haralick, "Organization of relational models for scene analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, pp. 595–602, 1982.

[16] L. Shapiro and R. Haralick, "A metric for comparing relational descriptions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, pp. 90–94, 1985.

[17] R. Mohan, *Perceptual Organization for Computer Vision.* PhD thesis, University of Southern California, 1989.

[18] D. Lowe and T. Binford, "Segmentation and aggregation : an approach to figure-ground phenomena," in *Proc. ARPA Image Understanding Workshop*, 1982.

[19] D. Lowe, *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers, 1985.

[20] S. Palmer, "The psychology of perceptual organization : a transformational approach," in *Human and Machine Vision*, (J. Beck, B. Hope, and A. Rosenfeld, eds.), N.Y.: Academic Press, 1983.

[21] B. L. Bullock, "The necessity for a theory of specialized vision," in *Computer Vision Systems*, (A. R. Hanson and E. M. Riseman, eds.), New York: Academic Press, 1978.

[22] J. Skryzpek, "Neural specification of a general purpose vision system," Tech. Rep. CSD-890072, Computer Science Department, University of California, Los Angeles, USA, 1989.

[23] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 4, pp. 4–22, 1987.

[24] B. M. Forrest, D. Roweth, N. Stroud, D. Wallace, and G. Wilson, "Neural network models," *Parallel Computing*, vol. 8, pp. 71–83, 1988.

[25] S. E. Fahlmann and G. E. Hinton, "Connectionist architecture for artificial intelligence," *IEEE Computer*, vol. 20, pp. 100–109, 1987.

[26] J. A. Feldman, "Dynamic connections in neural networks," *Biological Cybernetics*, vol. 46, pp. 27–39, 1982.

[27] J. A. Feldmann and D. H. Ballard, "Connectionist models and their properties," *Cognitive Science*, vol. 6, pp. 205–254, 1982.

[28] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing : Explorations in Microstructures of Cognition (Ed.), Vol I.* Cambridge MA: Bradford Books/MIT Press, 1986.

[29] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation.* California: Addison-Wesley, 1991.

[30] D. R. Hush and B. G. Horne, "Progress in supervised neural networks," *IEEE Signal Process. Magazine*, vol. , pp. 8–39, 1993.

[31] H. E. Weschler, *Neural networks for perception.* San Diego, CA: Academic Press Inc., 1992.

[32] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141–152, 1985.

[33] J. J. Hopfield and D. W. Tank, "Computing with neural circuits : a model," *Science*, vol. 233, pp. 625–633, 1986.

[34] G. V. Wilson and G. S. Pavley, "On the stability of the traveling salesman problem algorithm of hopfield and tank," *Biological Cybernetics*, vol. 58, pp. 63–70, 1988.

[35] W. E. Lillo, M. H. Loh, S. Hui, and S. H. Żak, "On solving constrained optimization problems with neural networks : a penalty method approach," *IEEE Trans. Neural Networks*, vol. 4, pp. 931–940, 1993.

[36] G. L. Bilbro, M. White, and W. Synder, "Image segmentation with neurocomputers," in *Neural Computers*, (R. Eckmiller and C. V. D. Malsberg, eds.), : Springer-Verlag, 1988.

[37] W. E. Blanz and S. L. Gish, "A connectionist classifier architecture applied to image segmentation," *Int. J. Pattern Recog. Artificial Intell.*, vol. 5, pp. 603–617, 1991.

[38] A. Ghosh, N. R. Pal, and S. K. Pal, "Self-organization for object-extraction using multilayer neural networks and fuzziness measure," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 54–68, 1993.

[39] S. Lu and A. Szeto, "Hierarchical artificial neural networks for edge enhancement," *Pattern Recognition*, vol. 26, pp. 1149–1163, 1993.

[40] L. Bedini and A. Tonazzini, "Neural network use in maximum entropy image restoration," *Image and Vision Computing*, vol. 8, pp. 108–114, 1990.

[41] T. A. Jamison and R. J. Schalkoff, "Image labeling : a neural network approach," *Image and Vision Computing*, vol. 6, pp. 203–213, 1988.

[42] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time delay neural networks," *IEEE Trans. Acoustics, Speech and Signal Process.*, vol. 37, pp. 328–340, 1989.

[43] E. Levin, R. Pieraccini, and E. Bocchieri, "Time-warping network : a neural approach to hidden markov model based speech recognition," *Int. J. Pattern Recog. Artificial Intell.*, vol. 7, pp. 783–799, 1993.

[44] A. N. Jain and A. H. Waibel, "Incremental parsing by modular recurrent connections networks," in *Advances in Neural Information Processing Systems*, (D. S. Touretzky, ed.), USA: Morgan Kaufmann, 1990.

[45] L. G. Miller and A. L. Gorin, "Strucutred networks for adaptive language acquisition," *Int. J. Pattern Recog. Artificial Intell.*, vol. 7, pp. 873–898, 1993.

[46] J. Marshall, "Self-organizing neural networks for perception of visual motion," *Neural Networks*, vol. 3, pp. 45–74, 1990.

[47] T. R. Tsao and V. Chen, "A neural scheme for optical flow computation based on gabor filters and generalized gradient method," *Neurocomputing*, vol. 6, pp. 305–325, 1994.

[48] S. I. Gallant, "Connectionist expert systems," *Communications Assoc. Comput. Mach.*, vol. 31, pp. 152–169, 1988.

[49] Y. Hayashi, "A neural expert system using fuzzy teaching input," in *Proc. 1st IEEE Int. Conf. on Fuzzy Syst., San Diego*, pp. 485–491, 1992.

[50] S. Mitra and S. K. Pal, "Logical operation based fuzzy MLP for classification and rule generation," *Neural Networks*, vol. 7, pp. 353–373, 1994.

[51] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, pp. 679–698, 1986.

[52] A. Rosenfeld, "Digital straight line segments," *IEEE Trans. on Computer*, vol. 23, pp. 1264–1269, 1974.

[53] W. Frei and C. C. Chen, "Fast boundary detection : a generalization and a new algorithm," *IEEE Trans. on Computer*, vol. 26, pp. 988–998, 1977.

[54] G. J. Vanderbrug, "Semilinear line detectors," *Computer Vision, Graphics and Image Processing*, vol. 4, pp. 287–293, 1975.

[55] G. J. Vanderbrug and A. Rosenfeld, "Linear feature mapping," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 8, pp. 768–774, 1978.

[56] L. T. Watson, K. Arvind, R. W. Enrich, and R. M. Haralick, "Extraction of lines and regions from gray tone line drawing images," *Pattern Recognition*, vol. 17, pp. 493–507, 1984.

[57] R. M. Haralick, "Ridge and valley on digital images," *Computer Vision, Graphics and Image Process.*, vol. 22, pp. 28–38, 1983.

[58] R. O. Duda and P. E. Hart, "Use of hough transform to detect lines and curves in pictures," *Communications ACM*, vol. 15, pp. 11–15, 1972.

[59] J. Sklansky, "On the hough technique for curve detection," *IEEE Trans. Computers*, vol. C-27, pp. 923–926, 1978.

[60] S. D. Shapiro, "Feature space transforms for curve detection," *Pattern Recognition*, vol. 10, pp. 129–143, 1978.

[61] C. Kimme, D. H. Ballard, and J. Sklansky, "Finding circles by an array of accumulators," *Communications Assoc. Comput. Mach.*, vol. 18, pp. 120–122, 1975.

[62] H. Wechsler and J. Sklansky, "Automatic detection of ribs in chest radiographs," *Pattern Recognition*, vol. 9, pp. 21–30, 1977.

[63] J. Illingworth and J. Kittler, "A survey of the hough transform," *Computer Vision, Graphics, and Image Processing*, vol. 44, pp. 87–116, 1988.

[64] D. Ballard and C. Brown, *Computer Vision.* Englewood Cliffs, New Jersy: Prentice-Hall Inc., 1982.

[65] A. Mansouri, M. A. S., and M. D. Levine, "Line detection in digital pictures : a hypothesis prediction/verification paradigm," *Computer Vision, Graphics and Image Processing*, vol. 40, pp. 95–114, 1987.

[66] S. Vasudevan, R. L. Cannon, J. C. Bezdek, and W. L. Cameron, "Heuristics for intermediate level road finding algorithm," *Computer Vision, Graphics and Image Processing*, vol. 44, pp. 175–190, 1988.

[67] S. W. Zucker, R. A. Hummel, and A. Rosenfeld, "An application of relaxation labelling to line and curve enhancement," *IEEE Trans. on Computer*, vol. 26, pp. 394–403, 1977.

[68] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labelling by relaxation operations," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 6, pp. 420–433, 1976.

[69] E. R. Hancock and J. Kittler, "Edge-labeling using dictionary based relaxation," *IEEE Trans. Patt. Analysis and Mach. Intell.*, vol. PAMI-12, pp. 165–181, 1990.

[70] E. R. Hancock and J. Kittler, "Discrete relaxation," *Pattern Recognition*, vol. 23, pp. 711–733, 1990.

[71] C. David and S. W. Zucker, "Potentials, valleys, and dynamic global coverings," Tech. Rep. TR-CIM-89-1, Computer Vision and Robotics Laboratory, McGill University, Canada, 1989.

[72] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: John Wiley & Sons, 1978.

[73] K. S. Fu, *Syntactic Pattern Recognition and Applications.* Englewood Cliffs, New Jersy: Prentice-Hall Inc., 1982.

[74] A. Pathak, S. K. Pal, and R. A. King, "Syntactic recognition of skeletal maturity," *Patt. Recogn. Lett.*, vol. 2, pp. 193–197, 1984.

[75] T. Vamos, "Industrial objects and machine parts recognition," in *Applications of Syntactic Pattern Recognition*, (K. S. Fu, ed.), New York: Springer-Verlag, 1977.

[76] R. Jakubowski, "Syntactic characterization of machine part shapes," *Cybernetics and Systems*, vol. 13, pp. 1–24, 1982.

[77] T. Pavlidis and F. Ali, "A hierarchical syntactic shape analyzer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, pp. 2–9, 1979.

[78] M. C. Bjorklund and T. Pavlidis, "Global shape analysis by k-syntactic similarity," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-3, pp. 144–155, 1981.

[79] K. C. You and K. S. Fu, "A syntactic approach to shape recognition using attributed grammars," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-9, pp. 334–345, 1979.

[80] K. C. You and K. S. Fu, "Distorted shape recognition using attributed grammars and error-correcting techniques," *Computer Graphics Image Process.*, vol. 12, pp. 1–16, 1980.

[81] W. H. Tsai and K. S. Fu, "Attributed grammar - a tool for combining syntactic and statistical approaches to pattern recognition," *IEEE Trans. Syst. Man, Cybern.*, vol. SMC-10, pp. 873–885, 1980.

[82] S. K. Pal and D. Dutta Majumder, *Fuzzy Mathematical Approach to Pattern Recognition.* New York: Wiley, 1986.

[83] M. Yachida and S. Tsuji, "A versatile machine vision system for complex industrial parts," *IEEE Trans. Computers*, vol. C-26, pp. 882–894, 1977.

[84] N. Y. Chen, J. R. Birk, and R. B. Kelley, "Estimating workpiece pose using the feature points method," *IEEE Trans. Auto. Control*, vol. AC-25, pp. 1027–1041, 1980.

[85] R. C. Bolles and R. A. Cain, "Recognising and locating partially visible objects," in *Robot Vision*, (A. Pugh, ed.), Berlin: Springer, 1983.

[86] R. Kashyap and M. Koch, "Computer vision algorithms used in the recognition of objects," in *Proc. IEEE Conf. on Robotics and Automation, St. Louis*, pp. 150–155, 1985.

[87] W. Wen and A. Lozzi, "Recognition and inspection of two-dimensional industrial parts using subpolygons," *Pattern Recognition*, vol. 25, pp. 1427–1434, 1992.

[88] M. Han and D. Jang, "The use of maximum curvature points for the recognition of partially occluded objects," *Pattern Recognition*, vol. 23, pp. 21–33, 1990.

[89] M. Eshera and K. Fu, "A similarity measure between attributed relational graphs for image analysis," in *Proc. 7th Int. Conf. on Pattern Recognition*, pp. 75–77, 1984.

[90] W. Grimson and T. Losano-Perez, "Recognition and localization of overlapping parts from sparse data in two and three dimensions," in *Proc. IEEE Conf. on Robotics and Automation, St. Louis*, pp. 61–66, 1985.

[91] A. Wallace, "Feature determination and inexact matching of images of industrial components," *IEEE Proc.*, vol. 132E, pp. 309–315, 1985.

[92] A. Wallace, "An informed strategy for matching models to images of segmented scenes," *Pattern Recognition*, vol. 20, pp. 349–363, 1987.

[93] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, pp. 111–122, 1981.

[94] J. L. Turney, T. Mudge, and R. A. Volz, "Recognizing partially occluded parts," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-7, pp. 410–421, 1985.

[95] G. Stockmann and A. Agarwala, "Equivalence of hough curve detection to matched filtering," *Graphics, Image Process.*, vol. 20, pp. 820–822, 1977.

[96] J. Segen, "Locating randomly oriented objects from partial views," in *Proc. 3rd Int. Conf. on Robot Vision and Sensory Controls, Cambridge, MA*, pp. 676–683, 1983.

[97] A. Arbuschi, V. Cantoni, and G. Musso, "Recognition and localization of parts using the hough transform technique," in *Digital Image Processing*, (Levialdi, ed.), California: Pitman, 1984.

[98] B. Bhanu and J. Ming, "Clustering based recognition of occluded objects," in *Proc. 8th Int. Conf. on Pattern Recognition, Paris*, pp. 732–734, 1986.

[99] J. R. Ullmann, "Edge replacement in the recognition of occluded objects," *Pattern Recognition*, vol. 26, pp. 1771–1784, 1993.

[100] T. Knoll and R. Jain, "Recognizing partially visible objects using feature indexed hypotheses," *IEEE journal of Robotics and Automation*, vol. RA-2, pp. 3–13, 1986.

[101] N. Ayache and O. Faugeras, "Hyper : a new approach for the recognition and positioning of two-dimensional objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, pp. 44–54, 1986.

[102] H. Tropf, "Analysis by synthesis search for semantic segmentation applied to workpiece recognition," in *Proc. 5th Int. Conf. on Pattern Recognition, Miami*, pp. 241–244, 1980.

[103] P. Rummel and W. Beutel, "Workpiece recognition and inspection by a model-based scene analysis system," *Pattern Recognition*, vol. 17, pp. 141–148, 1984.

[104] S. Chaudhury, A. Acharya, S. Subramanian, and G. Parthasarathy, "Recognition of occluded objects with heuristic search," *Pattern Recognition*, vol. 23, pp. 617–635, 1990.

[105] B. Bhanu and O. Faugeras, "Shape matching of two dimensional objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, pp. 137–155, 1984.

[106] T. Henderson and A. Samal, "Multiconstraint shape analysis," *Image Vision Computing*, vol. 4, pp. 84–96, 1986.

[107] T. Henderson and A. Samal, "2-d scene analysis using split-level relaxation," in *Proc. 8th Int. Conf. on Pattern Recognition, Paris*, pp. 195–197, 1986.

[108] G. Medioni and R. Nevatia, "Matching images using linear features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, pp. 675–685, 1984.

[109] W. E. L. Grimson, "On the recognition of curved objects," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-11, pp. 632–643, 1989.

[110] S. Umeyama, "Parameterized point pattern matching and its application to recognition of object families," *IEEE Trans. Patt., Anal. Mach. Intell.*, vol. 15, pp. 136–144, 1993.

[111] K. E. Price, "Matching closed contours," in *Proc. Seventh Int. Conf. Patt. Recogn.*, (Montreal, Canada), pp. 990–992, 1984.

[112] J. W. Gorman, O. R. Mitchell, and F. P. Kuhl, "Partial shape recognition using dynamic programming," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-10, pp. 257–266, 1988.

[113] N. Ansari and E. J. Delp, "Partial shape recognition : a landmark-based approach," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-12, pp. 470–483, 1990.

[114] F. Rosenblatt, "The perceptron : a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–408, 1958.

[115] M. L. Minsky and S. A. Papert, *Perceptrons.* Cambridge: MIT Press, 1969.

[116] P. J. Werbos, *Beyond Regression : New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Dept. of Applied Mathematics, Harvard University, 1974.

[117] M. Hoehfeld and S. Fahlman, "Learning with limited numerical precision using the cascade-correlation algorithm," *IEEE Trans. Neural Networks*, vol. 3, pp. 602–611, 1992.

[118] R. A. Jacobs, "Incremental rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, pp. 295–307, 1988.

[119] N. K. Bose and A. K. Garga, "Neural network design using voronoi diagrams," *IEEE Trans. Neural Networks*, vol. 4, pp. 778–787, 1993.

[120] B. Yoon, D. J. Holmes, and G. Langholz, "Efficient genetic algorithms for training layered feedforward neural networks," *Information Sciences*, vol. 76, pp. 67–85, 1994.

[121] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281–294, 1989.

[122] F. J. Pineda, "Recurrent back-propagation and dynamical approach to adaptive neural computation," *Neural Computation*, vol. 1, pp. 161–172, 1989.

[123] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. National Academy of Sciences, USA, pp 2554-2558*, 1982.

[124] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two state neurons," in *Proc. National Academy of Sciences, USA, pp 3088-3092*, 1984.

[125] B. Kosko, *Neural Networks and Fuzzy Systems : A Dynamical Approach to Machine Intelligence.* Englewood Cliffs, N.J.: Prentice Hall, 1991.

[126] B. Kosko, "Bidirectional associative memories," *IEEE Trans. Syst., Man, and Cybern.*, vol. 18, pp. 49–60, 1988.

[127] P. Simpson, "Higher ordered and intraconnected bidirectional associative memories," *IEEE Trans. Syst., Man, and Cybern.*, vol. 20, pp. 637–653, 1990.

[128] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for bolzmann machines," *Cognitive Science*, vol. 9, pp. 147–169, 1985.

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

[130] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics and Image Processing*, vol. 37, pp. 34–115, 1987.

[131] G. Carpenter and S. Grossberg, "Self-organization of stable category recognition codes for analog input patterns," *Applied Optics*, vol. 26, pp. 4919–4930, 1987.

[132] G. Carpenter and S. Grossberg, "Art3 : hierarchical search using chemical transmitters in self-organizing pattern recognition architectures," *Neural Networks*, vol. 3, pp. 129–152, 1990.

[133] G. Carpenter, S. Grossberg, and D. Rosen, "Fuzzy art : fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 493–504, 1991.

[134] G. Carpenter, S. Grossberg, and J. Reynolds, "Artmap : supervised real-time learning and classification of nonstationary data by a self-organizing neural network," *Neural Networks*, vol. 4, pp. 565–588, 1991.

[135] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds, and D. Rosen, "Fuzzy artmap : a neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Networks*, vol. 3, pp. 698–713, 1992.

[136] S. Amari, "Learning patterns and pattern sequences by self-organizing nets of threshold elements," *IEEE Trans. Computer*, vol. C-21, pp. 1197–1206, 1972.

[137] S. Amari, "Neural theory of association and concept formation," *Biological Cybernetics*, vol. 26, pp. 175–185, 1977.

[138] S. Amari and K. Maginu, "Statistical neurodynamics of associative memory," *Neural Networks*, vol. 1, pp. 63–74, 1988.

[139] J. A. Anderson, J. W. Silverstein, S. R. Ritz, and R. S. Jones, "Distinctive features, categorical perception, and probability learning," *Psychological Review*, vol. 84, pp. 413–451, 1977.

[140] L. O. Chua and L. Yang, "Cellular neural networks : applications," *IEEE Trans. on Circuits and Systems*, vol. 35, pp. 1273–1290, 1988.

[141] Y. Peng and J. Reggia, "A connectionist model for diagnostic problem solving," *IEEE Systems, Man, and Cybern.*, vol. 19, pp. 285–298, 1989.

[142] S. Cho and J. Reggia, "Learning competition and cooperation," *Neural Computation*, vol. 5, pp. 242–259, 1993.

[143] J. Reggia, C. DÁutrechy, G. Sutton III, and M. Weinrich, "A competitive distribution theory of neocortical dynamics," *Neural Computation*, vol. 4, pp. 287–317, 1992.

[144] J. Marshall, "A self-organizing scale-sensitive network," in *Proc. Int. Joint Conf. Neural Networks, San Diego, CA*, pp. 649–654, 1990.

[145] J. Marshall, "Representation of uncertainty in self-organizing neural networks," in *Proc. Int. Neural Networks Conf., Paris, France*, pp. 809–812, 1990.

[146] J. Marshall, "Development of perceptual context-sensitivity in unsupervised neural networks : parsing, grouping and segmentation," in *Proc. Int. Joint Conf. Neural Networks, Baltimore, MD*, pp. 315–320, 1992.

[147] M. Cohen and S. Grossberg, "Neural dynamics of speech and language coding : developmental programs, perceptual grouping, and competition for short-term memory," *Human Neurobiology*, vol. 5, pp. 1–22, 1986.

[148] M. Cohen and S. Grossberg, "Masking fields : a massively parallel neural architecture for learning, recognizing, and predicting multiple groupings of data," *Applied Optics*, vol. 26, pp. 1866–1891, 1987.

[149] A. Nigrin, "The stable classification of temporal sequences with an adaptive resonance circuit," in *Proc. Int. Joint Conf. Neural Networks, Washington DC*, pp. 525–528, 1990.

[150] A. Nigrin, "Sonnet : a self-organizing neural network that classifies multiple patterns simultaneously," in *Int. Joint Conf. Neural Networks, San Diego, CA*, pp. 313–318, 1990.

[151] A. Nigrin, *The stable learning of temporal patterns with an adaptive resonance circuit.* PhD thesis, Duke University, 1990.

[152] A. Nigrin, "A new architecture for achieving translational invariant recognition of objects," in *Proc. Int. Joint Conf. Neural Networks, Baltimore, MD*, pp. 683–688, 1992.

[153] S. Grossberg and E. Mingolla, "Neural dynamics of form perception : boundary completion, illusory figures, and neon color spreading," *Psychological Review*, vol. 92, pp. 173 – 211, 1985.

[154] M. Cohen and S. Grossberg, "Neural dynamics of brightness perception : features, boundaries, diffusion and resonance," *Perception and Psychophysics*, vol. 36, pp. 428–456, 1984.

[155] S. Grossberg and E. Mingolla, "Neural dynamics of perceptual grouping : textures, boundaries, and emergent segmentations," *Perception and Psychophysics*, vol. 38, pp. 141–171, 1985.

[156] R. Linsker, "From basic network principles to neural architecture : emergence of spatial opponent cells," in *Proc. National Academy of Sciences, USA, Vol 83, pp 7508-7512*, 1986.

[157] R. Linsker, "From basic network principles to neural architecture : emergence of orientation-selective cells," in *Proc. National Academy of Sciences, USA, Vol 83, pp 8390-8394*, 1986.

[158] R. Linsker, "From basic principles to neural architectures : emergence of orientation columns," in *Proc. national Academy of Sciences, USA, Vol 83, pp 8779-8783*, 1986.

[159] M. Bengtsson, "A neural system as a dynamical model for early vision," *Neural Networks*, vol. 6, pp. 313–325, 1993.

[160] G. K. Knopf and M. M. Gupta, "A multipurpose neural processor for machine vision systems," *IEEE Trans. Neural Networks*, vol. 4, pp. 762–777, 1993.

[161] N. Nasrabadi and W. Li, "Object recognition by a hopfield neural network," *IEEE Trans. Syst., Man, and Cybern.*, vol. 21, 1991.

[162] W. Li and M. Nasrabadi, "Object recognition based on graph matching implemented by a hopfield style network," in *Int. J. Conf. Neural Networks,II, Washington, DC*, 1989.

[163] N. Ansari and K. Li, "Landmark-based shape recognition by a modified hopfield neural network," *Pattern Recognition*, vol. 26, pp. 531–542, 1993.

[164] W. Lin, F. Liao, and T. Lingutla, "A hierarchical multiple-view approach to three-dimensional object recognition," *IEEE Trans. on Neural Networks*, vol. 2, pp. 84–92, 1991.

[165] P. W. M. Tsang, P. C. Yuen, and F. K. Lam, "Recognition of occluded objects," *Pattern Recognition*, vol. 25, pp. 1107–1117, 1992.

[166] P. W. M. Tsang and P. C. Yuen, "Recognition of partially occluded objects," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 23, 1993.

[167] G. N. Bebis and G. M. Papadourakis, "Object recognition using invariant object boundary representations and neural network models," *Pattern Recognition*, vol. 25, pp. 25–44, 1992.

[168] N. Srinivasa and M. Jouaneh, "An invariant pattern recognition machine using a modified art architecture," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 23, pp. 1432–1437, 1993.

[169] K. Fukushima, "Cognitron : a self-organizing multilayered neural network," *Biological Cybernetics*, vol. 20, pp. 121–136, 1975.

[170] K. Fukushima, "Neocognitron : a self organizing neural net model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[171] K. Fukushima and S. Miyake, "Neocognitron : a new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, vol. 15, pp. 455–469, 1982.

[172] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron : a neural network model for a mechanism of visual pattern recognition," *IEEE Transactions of Systems, Man, and Cybernetics*, vol. 13, pp. 826–834, 1983.

[173] K. Fukushima, "A hierarchical neural net model for associative memory," *Biological Cybernetics*, vol. 50, pp. 105–113, 1984.

[174] K. Fukushima, "Neural net model for selective attention in visual pattern recognition," *Biological Cybernetics*, vol. 55, pp. 5–15, 1986.

[175] K. Fukushima, "Neural network model for selective attention in visual pattern recognition and asociative recall," *Applied Optics*, vol. 26, pp. 4985–4992, 1987.

[176] K. Fukushima, "Neocognitron : A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, pp. 119–130, 1988.

[177] K. Fukushima, "A neural network for visual pattern recognition," *IEEE Computer*, pp. 65–75, March 1988.

[178] K. Fukushima, T. Imagawa, and E. Ashida, "Character recognition with selective attention," in *Proc. Int. Conf. Neural Networks*, pp. I–593 – I–598, 1991.

[179] K. Fukushima, "Character recognition with neural network," *Neurocomputing*, vol. 4, pp. 221–233, 1992.

[180] K. Fukushima and T. Imagawa, "Recognition and segmentation of connected characters with selective attention," *Neural Networks*, vol. 6, pp. 33–41, 1993.

[181] G. Himes and R. Iñigo, "Automatic target recognition using a neocognitron," *IEEE Trans. on Knowledge and Data Engineering*, vol. 4, pp. 167–172, 1992.

[182] J. Minnix, E. McVey, and R. Iníigo, "A multilayered self-organizing artificial neural network for invariant pattern recognition," *IEEE Trans. Knowledge and Data Engg.*, vol. 4, pp. 162–167, 1992.

[183] G. Hinton, "Shape representation in parallel systems," in *Proc. IJCAI-81, International Joint Committe for Artificial Intelligence*, 1981.

[184] G. Hinton, "A parallel computationm that assigns cannnonical object-based frames of reference," in *Proc. IJCAI-81, International Joint Committee for Artificial Intelligence*, 1981.

[185] G. E. Hinton, "Mapping part-whole hierarchies into connectionist networks," *Artificial Intelligence*, vol. 46, pp. 47–75, 1990.

[186] M. Mozer, *The perception of multiple objects : A connectionist approach.* Cambridge, MA: MIT Press, 1991.

[187] M. Mozer and M. Behrmann, "On the interaction of selective attention and lexical knowledge : a connectionist account of neglect dyslexia," Tech. Rep. CU-CS-441-89, Dept. of Computer Science, University of Colorado, Boulder, USA, 1989.

[188] R. S. Zemel, M. C. Mozer, and G. E. Hinton, "Traffic : a model of object recognition based on transformation of feature instances," Tech. Rep. CRG-TR-7, University of Toronto, Toronto, 1988.

[189] R. Zemel, "Traffic : a connectionist model of object recognition," Tech. Rep. CRG-TR-89-2, Dept. of Computer Science, University of Toronto, Canada, 1989.

[190] T. Poggio and S. Edelman, "A network that learns to recognize three-dimensional objects," *Nature*, vol. 343, pp. 263–266, 1990.

[191] S. Edelman and D. Weinshall, "A self-organizing multiple-view representation of 3-d objects," *Biological Cybernetics*, vol. 64, pp. 209–219, 1991.

[192] T. Poggio, S. Edelman, and M. Fahle, "Learning of visual modules from examples : a framework for understanding adaptive visual performance," *CVGIP : Image Understanding*, vol. 56, pp. 22–30, 1992.

[193] S. Edelman, "A network model of object recognition in human vision," in *Neural Networks for Perception*, (H. Wechsler, ed.), San Diego, CA: Academic Press Inc., 1992.

[194] H. Wechsler and G. L. Zimmerman, "2-d invariant object recognition using distributed associative memories," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. PAMI-10, pp. 811–821, 1988.

[195] G. L. Zimmerman, "Two-dimensional maps and biological vision : representing three-dimensional space," in *Neural Networks for Perception*, (H. Wechsler, ed.), San Diego, CA: Academic Press Inc., 1992.

[196] Y. Hirai and Y. Tsukui, "Position independent pattern matching by neural network," *IEEE Trans. Syst., Man, and Cybern.*, vol. 20, pp. 816–825, 1990.

[197] L. Chan, "Neural networks for collective translational invariant object recognition," *Int. J. Patt. Rec. Artificial Intell.*, vol. 6, pp. 143–156, 1992.

[198] W. Li and N. M. Nasrabadi, "Invariant object recognition based on a neural network of cascaded rce nets," *Int. J. Patter Recog. Artificial Intell.*, vol. 7, pp. 815–829, 1993.

[199] L. Spirkovska and M. B. Reid, "Coarse-coded higher-order neural networks for psri object recognition," *IEEE Trans. Neural Networks*, vol. 4, pp. 276–283, 1993.

[200] J. Feldman, "Four frames suffice : a provisional model of vision and space," *The Behavioral and Brain Sciences*, vol. 8, pp. 265–289, 1985.

[201] D. Sabbah, "Computing with connections in visual recognition of origami objects," *Cognitive Science*, vol. 9, pp. 25–50, 1985.

[202] S. M. Kosslyn, "Information representation in visual images," *Cognitive Psychology*, vol. 7, pp. 341–370, 1975.

[203] S. Kosslyn, J. Holtzman, M. Farah, and M. Gazzaniga, "A computational analysis of mental image generation : evidence from functional dissociations in split-brain patients," *Journal of Experimental Psychology : General*, vol. 114, pp. 311–341, 1985.

[204] J. Basak, B. Chanda, and D. Dutta Majumder, "An approach to line linking based on neural network model," in *Proc. Neuro Nimes*, (France), pp. 491–506, 1991.

[205] J. Basak, B. Chanda, and D. Dutta Majumder, "On edge and line linking in graylevel images with connectionist models," *IEEE Trans. Systems, Man, and Cybern.*, vol. 24, pp. 413–428, 1994.

[206] J. Basak, S. Chaudhury, S. K. Pal, and D. Dutta Majumder, "Matching of structural shape descriptions with hopfield net," *Int. J. Patt. Recogn. Artificial Intell.*, vol. 7, pp. 377–404, 1993.

[207] J. Basak, C. A. Murthy, S. Chaudhury, and D. Dutta Majumder, "A connectionist network for simultaneous perception of multiple categories," in *Proc. Eleventh IAPR Int. Conf. on Pattern Recognition*, (Hague, Netherlands), 1992.

[208] J. Basak, C. A. Murthy, S. Chaudhury, and D. Dutta Majumder, "A connectionist model for category perception : theory and implementation," *IEEE Trans. Neural Networks*, vol. 4, pp. 257–269, 1993.

[209] J. Basak, C. A. Murthy, and S. K. Pal, "A self-organizing connectionist model for mixed category perception," *Neurocomputing*, (Accepted), 1994.

[210] J. Basak and S. K. Pal, "X-tron : an incremental connectionist model for category perception," *IEEE Trans. Neural Networks*, (Accepted), 1994.

[211] J. Basak and S. K. Pal, "A self organising model for mixed category perception," in *Proc. INSA-CSI Seminar on Pattern Recognition, Artificial Intelligence, and Neural Networks*, (Dehradun, India), pp. 5–7, 1993.

[212] J. Basak, N. R. Pal, and S. K. Pal, "A connectionist system for learning and recognition of structures," *Neural Networks*, (Accepted), 1994.

[213] J. Basak, N. R. Pal, and S. K. Pal, "From linear features to peaks in hough space : a neural network approach," *Int. J. Patt. Recog. Art. Intell.*, (Communicated), 1994.

[214] J. Basak, N. R. Pal, and S. K. Pal, "A connectionist implementation of hough transform," in *Proc. ICAPRDT*, (Calcutta, India), pp. 242–249, 1993.

[215] J. Basak, N. R. Pal, and S. K. Pal, "A connectionist system for handwritten character recognition," in *Proc. 3rd Symp. Intelligent Syst.*, (Bangalore, India), pp. 21–27, 1993.

[216] J. Basak, N. R. Pal, and S. K. Pal, "A novel connectionist approach for automatic peak selection in hough space," in *Proc. Indo-US workshop*, (Pune, India), 1993.

[217] J. Basak and S. K. Pal, "Psycop : a psychologically motivated connectionist system for object perception," *IEEE Trans. Neural Networks*, (Accepted), 1994.

[218] J. Basak and S. K. Pal, "A new connectionist framework for determining 'what' and 'where' for mixed object recognition," *Int. J. Patt. Recogn. Artificial Intell.*, (Communicated), 1994.

[219] R. B. Ash, *Real Analysis and Probability.* London: Academic Press, 1972.

[220] L. G. Shapiro, J. D. Moriarty, R. M. Haralick, and P. G. Mulgaonkar, "Matching three-dimensional objects using a relational approach," *Pattern Recognition*, vol. 17, pp. 385–405, 1984.

[221] K. L. Boyer, A. J. Vayda, and A. C. Kak, "Robotic manipulation experiments using structural stereopsis for 3d vision," *IEEE Expert*, vol. , pp. 73–94, 1986.

[222] A. Eshera and K. S. Fu, "Recognising and locating partially visible objects," *IEEE trans. on Systems, Man and Cybernetics*, vol. SMC-14, pp. 398–412, 1984.

[223] E. Mjolness, G. Gindi, and P. Anandan, "Optimisation in model matching and perceptual organisation," *Neural Computation*, vol. 1, pp. 218–229, 1989.

[224] V. Govindan and A. P. Shivaprasad, "Character recognition - a review," *Pattern Recognition*, vol. 23, pp. 671–683, 1990.

[225] C. Stanfill and D. Waltz, "Toward memory-based reasoning," *Communications of the ACM*, vol. 29, pp. 1213–1228, 1986.

[226] D. O. Hebb, *The Organization of Behavior.* New York: Wiley, 1949.

[227] D. P. Mandal, C. A. Murthy, and S. K. Pal, "Formulation of a multivalued recognition system," *IEEE Trans. Syst., Man, and Cyberns.*, vol. SMC-20, pp. 607–620, 1992.

[228] K. Lua and K. Gan, "Recognizing chinese characters through interactive activation and competition," *Pattern Recognition*, vol. 23, pp. 1311–1321, 1990.

[229] K. Gan and K. Lua, "Chinese character classification using an adaptive resonance network," *Pattern Recognition*, vol. 25, pp. 877–882, 1992.

[230] L. Schomaker, "Using stroke- or character-based self-organizing maps in the recognition of on-line, connected cursive script," *Pattern Recognition*, vol. 26, pp. 443–450, 1993.

[231] Y. Yong, "Chinese character recognition via neural networks," *Pattern Recognition Letters*, vol. 7, pp. 19–25, 1988.

[232] B. Takács, O. Johnson, and G. Pieroni, "Two-stage learning of handwritten characters by clf networks," *Neural Network World*, vol. 1, pp. 41–51, 1993.

[233] Y. Le Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," in *Proc. Neural Information Processing Systems, (Ed. D. Touretzky)*, 1991.

[234] Y. Le Cun, "Generalization and network design strategies," Tech. Rep. CRG-TR-89-4, Dept. of Computer Sc., Univ. of Toronto, Canada, 1989.

[235] J. Denker, W. Gardner, H. Graf, D. Henderson, R. Howard, W. Hubbard, L. Jackel, H. Baird, and I. Guyon, "Neural network recognizer for handwritten zip code digits," in *Proc. Neural Information Processing Systems, (Ed. D. Touretzky)*, 1991.

[236] V. V. Vinod, S. Chaudhury, S. Ghose, and J. Mukherjee, "A connectionist approach for peak detection in hough space," *Pattern Recognition*, vol. 25, pp. 1253–1264, 1992.

[237] F. Cheng, W. Hsu, and M. Chen, "Recognition of handwritten chinese characters by modified hough transform techniques," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-11, pp. 429–439, 1989.

[238] L. Zadeh, K. Fu, K. Tanaka, and M. E. Shimura, *Fuzzy Sets and Their Applications to Cognitive and Decision Process.* London: Academic Press, 1975.

[239] L. Ungerleider and M. Mishkin, "Two cortical visual systems," in *Analysis of visual behavior*, (D. Ingle, M. Goodale, and R. Mansfield, eds.), Cambridge, MA: MIT Press, 1982.

[240] J. Duncan and G. Humphreys, "Visual search and stimulus similarity," *Psychological Review*, vol. 96, pp. 433–458, 1989.

[241] D. LaBerge and V. Brown, "Theory of attentional operations in shape identification," *Psychological Review*, vol. 96, pp. 101–124, 1989.

[242] R. Phaf, A. Heijden, and P. Hudson, "Slam : a connectionist model for attention in visual selection tasks," *Cognitive Psychology*, vol. 22, pp. 273–341, 1990.

[243] R. Shiffrin and W. Schneider, "Controlled and automatic human information processing : ii, perceptual learning, automatic attending, and a general theory," *Psychological Review*, vol. 84, pp. 127–190, 1977.

[244] C. Bundsen, "A theory of visual attention," *Psychological Review*, vol. 97, pp. 523–547, 1990.

[245] D. Ballard, "Parameter nets," *Artificial Intelligence*, vol. 22, pp. 235–267, 1984.

[246] 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.

[247] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Mass: Addison-Wesley, 1989.

[248] A. K. Bhattacharjya and B. Roysam, "Joint solution of low-, intermediate-, and high-level vision tasks by evolutionary optimization : application to computer vision at low snr," *IEEE Trans. Neural Networks*, vol. 5, pp. 83–95, 1994.

List of Publications of the Author

1. J. Basak, C. A. Murthy, S. Chaudhury, and D. Dutta Majumder, "A connectionist model for category perception : theory and implementation," *IEEE Trans. Neural Networks*, vol. 4, pp. 257–269, 1993.

2. J. Basak, S. Chaudhury, S. K. Pal, and D. Dutta Majumder, "Matching of structural shape descriptions with hopfield net," *Int. J. Patt. Recogn. Artificial Intell.*, vol. 7, pp. 377–404, 1993.

3. J. Basak, B. Chanda, and D. Dutta Majumder, "On edge and line linking in graylevel images with connectionist models," *IEEE Trans. Systems, Man, and Cybern.*, vol. 24, pp. 413–428, 1994.

4. J. Basak, C. A. Murthy, and S. K. Pal, "A self-organizing connectionist model for mixed category perception," *Neurocomputing*, (Accepted), 1994.

5. J. Basak and S. K. Pal, "X-tron : an incremental connectionist model for category perception," *IEEE Trans. Neural Networks*, (Accepted), 1994.

6. J. Basak and S. K. Pal, "Psycop : a psychologically motivated connectionist system for object perception," *IEEE Trans. Neural Networks*, (Accepted), 1994.

7. J. Basak, N. R. Pal, and S. K. Pal, "A connectionist system for learning and recognition of structures," *Neural Networks*, (Accepted), 1994.

8. J. Basak, B. Chanda, and D. Dutta Majumder, "An approach to line linking based on neural network model," in *Proc. Neuro Nimes*, (France), pp. 491–506, 1991.

9. J. Basak, C. A. Murthy, S. Chaudhury, and D. Dutta Majumder, "A connectionist network for simultaneous perception of multiple categories," in *Proc. Eleventh IAPR Int. Conf. on Pattern Recognition*, (Hague, Netherlands), 1992.

10. J. Basak and S. K. Pal, "A self organising model for mixed category perception," in *Proc. INSA-CSI Seminar on Pattern Recognition, Artificial Intelligence, and Neural Networks*, (Dehradun, India), pp. 5–7, 1993.

11. J. Basak, N. R. Pal, and S. K. Pal, "A connectionist implementation of hough transform," in *Proc. ICAPRDT*, (Calcutta, India), pp. 242–249, 1993.

12. J. Basak, N. R. Pal, and S. K. Pal, "A connectionist system for handwritten character recognition," in *Proc. 3rd Symp. Intelligent Syst.*, (Bangalore, India), pp. 21–27, 1993.

13. J. Basak, N. R. Pal, and S. K. Pal, "A novel connectionist approach for automatic peak selection in hough space," in *Proc. Indo-US workshop*, (Pune, India), 1993.

14. J. Basak and S. K. Pal, "A new connectionist framework for determining 'what' and 'where' for mixed object recognition," *Int. J. Patt. Recogn. Artificial Intell.*, (Communicated), 1994.

15. J. Basak, N. R. Pal, and S. K. Pal, "From linear features to peaks in hough space : a neural network approach," *Int. J. Patt. Recog. Art. Intell.*, (Communicated), 1994.

16. D. P. Mukherjee, J. Basak, and D. Dutta Majumder, "A corner detection algorithm and its parallel implementation," *Proc. of ICARCV92*, Singapore, 1992, pp cv-1.6.1 - cv-1.6.5.

17. J. Basak, B. Chanda, and D. Dutta Majumder, "CALF : A connectionist architecture for line finding," *Proc. PARCOM-90 Conf.*, Pune, India, Dec., 1990.

18. J. Basak, B. Chanda, and D. Dutta Majumder, "A connectionist model for detecting lines in graylevel images," *Tech. Report, No. TR/KBCS/4/91*, NCKBC, Indian Statistical Institute, Calcutta, 1991.

19. J. Basak, B. Chanda, and D. Dutta Majumder, "A connectionist model for edge detection," *Tech. Report, No. TR/KBCS/5/91*, NCKBC, Indian Statistical Institute, Calcutta, 1991.