M. Tech. (Computer Science) Dissertation Series

# Development of a Connectionist Model for Image Skeletonization

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

by

*Prem Shankar Patel*

under the supervision of

*Mr. J. Basak and Dr. N.R. Pal*

**INDIAN STATISTICAL INSTITUTE**
203, Barrackpore Trunk Road
Calcutta-700 035

submitted on July 22, 1994

# CERTIFICATE

This is to certify that the dissertation work entitled *Development of Connectionist Model for Image Skeletonization* completed and submitted by *Prem Shankar Patel*, as a partial fulfillment of the requirements for the degree of M. Tech.(Computer Science) is a bonafide record of work done under our guidance and supervision.

| | |
|---|---|
| Mr. J. Basak | Dr. N.R. Pal |
| Programmer, | Associate professor, |
| M.I.U., I.S.I., | M.I.U., I.S.I., |
| 203, B.T. Road, | 203, B.T. Road, |
| Calcutta-700 035. | Calcutta-700 035. |

# Acknowledgement

I take this opportunity to thank my supervisers Mr. J. Basak and Dr. N R Pal for their guidance throughout this study. I would also like to extend my thanks Dr. M.K. Kundu, Head of Machine Intelligence Unit(MIU), for his valuable suggestions and for providing computer facilities in MIU.

[illegible]

Prem Shankar Patel

ISI, CALCUTTA
LIBRARY
26/07/94.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# INTRODUCTION

## 1.1 Introduction

Image skeletonization, also popularly known as thinning, is a process in which a given image is transformed to a single pixel thick image preserving important image features such as skeletal points, image symmetry, connectivity etc. It helps in reducing the storage space requirement for storing input data by removing unnecessary details and is also useful in extraction of the important image features. It is an important preprocessing step for many image analysis tasks such as finger print identification, optical character recognition, remotely sensed imagery, automated inspection of PCBs etc.

In literature, there exists several thinning algorithms which can be broadly classified into two categories: sequential and parallel algorithms. A parallel algorithm uses only the result obtained from the previous iteration for deleting(or reducing graylevel of) a pixel in the current iteration [1, 3, 4, 5, 7, 11]. On the other hand, a sequential algorithm makes use of both the result obtained from the previous iteration as well as the results obtained so far in the current iteration to process the candidate pixel [7, 13, 14]. The sequential algorithms have advantage that they are faster to implement on the usual general purpose sequential computers than parallel algorithms [17].

In case of two-tone images, there are only two graylevels one for background pixels and other for object pixels. Using a set of rules, it is required to decide whether a pixel is to be deleted or not. If the pixel satisfies the deletion criteria then its graylevel is changed to background graylevel. On the other hand, in case of graytone images, the whole deletion procedure becomes rather more complicated due to the fact that the object as well as background both may have a range of graylevels instead of single graylevels. One approach for thinning a graytone image, is to first convert it into a two-tone image by suitable thresholding and then apply any two-tone thinning algorithm to get the desired result. A problem associated with this method is that the information other than the object outline is lost forever and further processing in the graytone domain is not possible. Also, the method is very sensitive to noise. All these shortcomings could possibly be eliminated to an extent if thinning is performed in the graytone domain itself. An additional advantage of using a graytone thinning algorithm is that the resulting image is enhanced both in contrast of the object and smoothing in the background[1, 8].

In this introductory chapter, some definitions associated with thinning has been given. A very brief survey of the existing parallel thinning algorithms has been made and some of the algorithms, which we will refer in subsequent discussions has been briefly described. Also, nature of the thinning problem and applicability of some of the existing artificial neural net(ANN) models for thinning has been discussed informally. Then, based on that a Multilayer Perceptron(MLP), also known as Backpropagation Neural Network(BPNN) has been selected for forming a basic building block, which will be finally used for development of the Connectionist Model for thinning. Scope of the thesis has been given in the last section.

## 1.2 Some Definitions Associated With Thinning

Let the image $\Sigma$ be a bounded regular grid of points in a two-dimensional plane. In $\Sigma$, It is assumed that the background pixels have lower graylevel than the object pixels. In case of two-tone images, the graylevel of a background pixel is considered to be 0 and that of an object pixel to be 1. Some standard definitions associated with two-tone images and two-tone thinning algorithms are as given below:

**border point** An object pixel is called a border point if it has one or more background pixel as its direct neighbors. It is also known as contour point.

**internal point** An object pixel is called an internal point if it has no background pixel as its direct neighbor.

**end point** A border point with exactly one object pixel as its direct neighbor is called an end point.

**single point** a border point with no object pixel as its direct neighbor is called a single point.

**connectivity** Any two points $P$ and $Q$ in $\Sigma$ are said to be connected if there exists a path from $P$ to $Q$ such that all intermediate pixels in the path are object pixels.

**simple point** A border point is called simple if the set of its 8 direct neighbors form exactly one connected component.

**connection points** Border points which are neither single points nor simple points are called connection points.

**skeletal points** Border points which are connection points, end points or single points are called skeletal points.

**thinning** [9, 17] Based on these definitions, the two-tone thinning can be defined as repeated replacing of border pixels by background pixels provided this does not disconnect any pair of object pixels in its local neighborhood. Basically, the final result of any thinning algorithm should have no simple point except that it is an end point.

In case of graytone images we can modify the definitions as follow:

**border point** A pixel of graylevel $l$ is called a border point of graylevel $l$ if it has one or more pixels with graylevel less than $l$ as its direct neighbors.

**internal point** A pixel of graylevel $l$ is called an internal point if it has no pixel with graylevel less than $l$ as its direct neighbor.

**end point** A border point of graylevel $l$ with exactly one pixel with graylevel $\geq l$ as its direct neighbor is called an end point.

**single point** A border point of graylevel $l$ with no pixel with graylevel $\geq l$ as its direct neighbor is called a single point.

**connectivity** [2] Let $\sigma(P)$ denotes the graylevel of a point $P \in \Sigma$. Any two points $P$ and $Q$ in $\Sigma$ are said to be connected if there exists a path $P = P_0, P_1, \ldots, P_n = Q$ such that for each $P_i$, $\sigma(P_i) \geq min(\sigma(P), \sigma(Q))$. In other words, two points are said to be connected if there exists a path joining them on which no point is lighter than either of them.

**simple point** As given above.

**connection points** As given above.

**skeletal points** As given above.

**thinning** [2, 9, 10] Based on these definitions, the graytone thinning can be defined as repeated replacing of each point's graylevel by the minimum of its neighbors, provided this does not disconnect any pair of points in its local neighborhood.

## 1.3 Some Existing Thinning Algorithms

Here, we provide a brief survey on parallel thinning algorithms. Thinning algorithms came into existence sometimes in late fifties. In 1966, Rutowitz [3] presented in proper form. This algorithm is computationally very expensive. So, later many researchers tried to improve this classical algorithm and in this way a large number of algorithms came into existence. Some thinning algorithms has been presented by Stefanelli et al.[16]. In 1975, Rosenfeld[15] gave a characterization of parallel thinning algorithms. Later Dyer and Rosenfeld[10] presented a thinning algorithm for graytone images and then a formal definition of graylevel connectivity was presented by Rosenfeld[2] which formed the mathematical basis of graylevel thinning. A significant improvement over Rutowitz algorithm was achieved by Zhang and Suen[4] in 1984, which was further modified by Lu et al.[5] in 1986. One can refer Smith[12] for a detailed survey of different types thinning algorithms existing till that time. Unfortunately, Zang-Suen thinning also suffered from a problem of two subiterations. Recently, in 1988, Zhang and Wang[11] presented a heuristically modified parallel thinning algorithm for two-tone images which not only eliminated the problem of two subiterations, but as far as my knowledge goes it is also the most efficient known parallel thinning algorithm till today.

| | | |
|---|---|---|
| $P_9$ $(i-1,j-1)$ | $P_2$ $(i-1,j)$ | $P_3$ $(i-1,j+1)$ |
| $P_8$ $(i,j-1)$ | $P_1$ $(i,j)$ | $P_4$ $(i,j+1)$ |
| $P_7$ $(i+1,j-1)$ | $P_6$ $(i+1,j)$ | $P_5$ $(i+1,j+1)$ |

Figure 1.1: $(3 \times 3)$ neighborhood of the candidate pixel $P_1$

In graytone domain a parallel graytone thinning algorithm has been presented by Kundu et al.[1] The PGTA is basically a generalization of the modified Zhang-Suen thinning algorithm for two-tone images. Rutowitz thinning algorithm, modified Zhang-Suen thinning algorithm, Zhang-Wang thinning algorithm and PGTA will be referred in our subsequent discussion and hence, these are briefly described in the following subsections.

### 1.3.1 Rutowitz Thinning Algorithm

In 1966, Rutowitz[3] gave a thinning algorithm for two-tone images. It is an iterative algorithm. In this algorithm, a $3 \times 3$ window as shown in figure 1.1 is used for each of the candidate pixels $P_1$ in each of the iterations. In a iteration, a candidate pixel $P_1$ gets deleted if all the conditions are satisfied:

1. $P_1 = 1$
2. $2 \leq B(P_1) \leq 6$
3. $A(P_1) = 1$
4. $P_2 * P_4 * P_8 = 0 \ or \ A(P_2) \neq 1$
5. $P_2 * P_4 * P_6 = 0 \ or \ A(P_4) \neq 1$

where, $A(P_1)$ = Number of 01 patterns in the ordered set $P_2, P_3, \ldots, P_9, P_2$.
$B(P_1)$ = Number of nonzero neighbors of $P_1$, and
all $*'s$ denote logical ANDs.

A new iteration starts only after the current iteration has been completely finished. The algorithm terminates when no pixels gets deleted in a iteration.

### 1.3.2 Modified Zhang-Suen Thinning Algorithm

In this algorithm[4, 5] also, the same $3 \times 3$ window as shown in figure 1.1 is used. Here, each iteration has been divided into two subiterations. In a subiteration, a candidate pixel $P_1$ gets deleted all the following conditions are satisfied:

| | | | |
|---|---|---|---|
| $P_{10}$ $(i-2,j-1)$ | $P_{11}$ $(i-2,j)$ | $P_{12}$ $(i-2,j+1)$ | $P_{13}$ $(i-2,j+2)$ |
| $P_9$ $(i-1,j-1)$ | $P_2$ $(i-1,j)$ | $P_3$ $(i-1,j+1)$ | $P_{14}$ $(i-1,j+2)$ |
| $P_8$ $(i,j-1)$ | $P_1$ $(i,j)$ | $P_4$ $(i,j+1)$ | $P_{15}$ $(i,j+2)$ |
| $P_7$ $(i+1,j-1)$ | $P_6$ $(i+1,j)$ | $P_5$ $(i+1,j+1)$ | $P_{16}$ $(i+1,j+2)$ |

Figure 1.2: The $4 \times 4$ window for Zhang-Wang thinning algorithm.

1. $P_1 = 1$
2. $2 \leq B(P_1) \leq 6$
3. $A(P_1) = 1$
4. $P_2 * P_4 * P_8 = 0$
5. $P_2 * P_4 * P_6 = 0$
   In the next subiteration, every thing except conditions (4) & (5) remains same. The conditions (4) & (5) change to (6) & (7) respectively as given below:
6. $P_2 * P_6 * P_8 = 0$
7. $P_4 * P_6 * P_8 = 0$

It can be noted that the first and third conditions of the Rutowitz and modified Zang-Suen algorithms are same. In the second condition, the lower limit of $B(P_1)$ has been increased from 2 to 3 and in last two conditions, the expensive calculation of $A(P_2)$ and $A(P_4)$ has been successfully avoided in the modified Zang-Suen algorithm.

### 1.3.3 Zhang-Wang Thinning Algorithm

In this algorithm, a $4 \times 4$ window as shown in figure 1.2 has been used for each candidate pixel $P_1$ and the 4*_th* and 5*_th* conditions of Rutowitz algorithm are heuristically modified to

4. $P_2 * P_4 * P_8 = 0 \ or \ P_{11} = 1$
5. $P_2 * P_4 * P_6 = 0 \ or \ P_{15} = 1$

This is the most efficient algorithm so far known which also eliminates the problem of two subiterations present in the modified Zang-Suen algorithm.

### 1.3.4 PGTA

This algorithm is a generalization of the modified Zang-Suen algorithm. Here, graylevels of the neighborhood pixels of a candidate pixel $P_1$ is mapped to a 2 levels compatible state depending on their spatial graylevel distribution. This two-tone mapping (thresholding) is done around a threshold value calculated locally.

The threshold value $T$ calculated over $N \times N$ window is given by

$$T = \frac{1}{N^2 - 1} \sum_{i=2}^{N^2} P_i \tag{1.1}$$

where $P_i$ is the graylevel of the $(i-1)\_th$ neighborhood pixel, while candidate pixel is not considered for computing $T$.

For a $3 \times 3$ window

$$T = \frac{1}{8} \sum_{i=2}^{9} P_i \tag{1.2}$$

Assuming that the graylevel of background is lower than the graylevel of object, the thresholded value of a neighborhood pixel $P_i$ is given by

$$P'_i = \begin{cases} 0 & \text{if } P_i \leq T \\ 1 & \text{otherwise.} \end{cases} \tag{1.3}$$

The condition (2) of the Modified Zhang-Suen algorithm has been slightly modified by reducing the upper limit of $B(P_1)$ from 6 to 5 and rest of the conditions (i.e.(3)-(6)) remained same as in the Modified Zhang-Suen algorithm. In a subiteration, graylevel of a candidate pixel $P_1$ is changed to

$$P'_1 = min(P_2, P_3, P_4, \ldots, P_9) \tag{1.4}$$

if all the following conditions are satisfied :

1. $P_1 > min(P_1, P_2, \ldots, P_9)$
2. $3 \leq B(P_1) \leq 5$
3. $A(P_1) = 1$
4. $P'_2 * P'_4 * P'_8 = 0$
5. $P'_2 * P'_4 * P'_6 = 0$
   In the next subiteration only condition (4) & (5) are changed to
6. $P'_2 * P'_6 * P'_8 = 0$
7. $P'_4 * P'_6 * P'_8 = 0$

Also, let $n_k$ denotes the number of pixels where the graylevel has changed from $(k-1)\_th$ to $k\_th$ iteration. If $n_k$ is less than a predefined number, the algorithm stops.

## 1.4 Applicability of an ANN Model to Thinning

Thinning is a process where change of graylevel of a pixel depends on its local neighborhood. Moreover, all pixels can be processed simultaneously. There are many classical parallel thinning algorithms already existing for this purpose.

Since, artificial neural network(ANN) models are mainly used for local neighborhood operations. Moreover, ANNs provide robust, massive parallel computational framework; ANNs are most suitable for this purpose. we will consider only the basic methodology adopted by the parallel thinning algorithms to form a basis for development of the Connectionist Model. ANN models are also popularly known as connectionist models, parallel distributed Processing models or neuromorphic system.

In literature, There exists many standard ANN models like Hopfield net, multilayer perceptron(MLP), self-organising nets, functional-link nets, cellular neural net(CNN) etc. Some of them require learning while others do not. Learning may be supervised or unsupervised. Here, neither it is possible to enumerate all properties of these nets nor we are intended to do so. Here, we want to select an ANN model, based on some desirable features and nature of the parallel thinning procedures, which may be most suitable for developing a basic building block for thinning. Then, using this basic building block, a complete model can be developed.

**Supervised learning can be performed** Since, in case of parallel thinning algorithms, we have given a candidate pixel along with certain numbers of its local neighbors in the form of a window and a set of rules specifying that when to delete(or reduce the graylevel of) that pixel. So, it is already known to us that what will be the possible input and what will be its corresponding output. So, a training data set can be generated and a supervised learning can be performed.

**Training data are not orthogonal** Since, a pixel in a given window can have any possible graylevel in the given range, if we want to perform learning then we have to learn all possible patterns exhaustively. If we want to store these patterns in some other form, e.g., as done in Hopfield net when used as associative memory, then also all patterns has to be stored exhaustively. So, these patterns to be learned or to be stored are not orthogonal to each other (orthogonality among only input data is of concern).

**The basic building block should have generalization capability** Since, a pixel in a given $N \times N$ window can take any of the possible graylevels in the given range; if there are m graylevels, then there will be total $m^{N^2}$ patterns to learn. e.g., for a $3 \times 3$ window with only two-tone input, there will be total 512 patterns to learn. In case of graytone images, it is not possible to learn all possible gray patterns due to its astronomical size. One way to deal with this problem is: learn only two-tone patterns and then generalize it to graytone images. So, the Model to be used for forming basic building block should have this generalization capability.

## Suitability of Hopfield Net

The Hopfield net is generally used as an associative memory or to solve optimization problems. This net has two major limitations when used as an associative memory. First, the number of patterns that can be stored and accurately recalled is severely limited. If too many patterns are stored, then the net may converge to a novel spurious pattern different from all stored patterns. Also, If the number of nodes in the Hopfield net is $N$, then the number of classes in which randomly generated exemplar patterns can be classified is generally kept well below $0.15N$. A second limitation associated with this net when used as associative memory is that it requires orthogonality among the patterns to be stored. If patterns are not orthogonal then they becomes unstable when stored. A number of orthogonalization procedures have been developed which eliminate this problem and improve performance of the net, but increased complexity of the net is unavoidable [20].

Also, the hopfield net is normally used with binary inputs. So, if we store only binary patterns in the net then it may not be possible to generalize the net for graytone images [20].

Due to these limitations, the Hopfield net when used as associative memory is not a good choice for forming the basic building block. Note that, it does not mean Hopfield net is not at all suitable. Some other technique may be employed.

## Thinning using CNN

Recently, a Cellular Neural Network(CNN) model for thinning has already been developed by Matsumoto et. al. [25]. It uses a CNN with eight planes, each one defined by a set of *peeling templates*, and a set of *stopping templates*. It is an analog and parallel processor, where thinning has been accomplished by implementing the following two tasks:

1. peeling the thick pixels off. Peeling templates has been employed for this purpose.
2. stopping the peeling process when the pixel size reduces to exactly one. Stopping templates has been used for this purpose.

## Suitability of MLP

As pointed out earlier, a set of training data can be generated and a supervised learning can be performed; a multilayer perceptron(MLP) can possibly be used for forming the basic building block. Moreover, MLP does not demand orthogonality among training data set. The decision regions formed by perceptrons are similar to those formed by maximum likelihood Gaussian classifiers which assumes input are uncorrelated and distributions for different classes differ only in mean values [20]. Due to this sort of classification property an MLP learned with only two-tone patterns can be able to generalize itself for use in thinning graytone images. All these inherent properties of MLP make it an excellent choice for use as a basic building block.

There are many other nets whose suitability can be check. Here, we have selected MLP for forming the basic basic building block.

## 1.5 Scope of the Thesis

In a recent paper, a thinning algorithm which is characterized as "A parallel graytone thinning algorithm (PGTA)" is presented by Kundu et al.[1] There are some concerns regarding this algorithm which bring into question its utility as a parallel graytone thinning algorithm. In this dissertation, It has been shown that the PGTA is even not able to ensure protection of skeletal points and as a consequence it is not able to preserve the graylevel connectivity in the image. Also, there is no proper stopping criteria for the iteration. All these problems arises mainly because of improper selection of threshold.

In this work, we have proved that the candidate pixel's graylevel is the unique value which can be used for thresholding its window; and, if used it will be able to eliminate all the above mentioned problems, effectively. In fact, there is no need to generate a thresholded window explicitly, in this case. Also, using Zhang-Wang algorithm[11] instead of modified Zhang-Suen algorithm[4, 5] a more efficient(almost twice as efficient as PGTA) and generalized version of parallel graytone thinning algorithm has been presented. We will refer this algorithm as Modified PGTA, henceforth. Also, a comparative study has been done to check capabilities of PGTA and Modified PGTA in case of noisy images.

Since, thinning requires massive parallel processing on its pixels in their local neighborhood, an artificial neural network(ANN) model is ideally suitable for this purpose. Here, an informal study to check applicability of an ANN model for development of a basic building block has been carried out in section 1.4 and based on that a multilayer perceptron(MLP), also known as backpropagation neural network(BPNN) has been chosen for forming the basic building block. Then, using this basic building block a new Connectionist Model for thinning has been proposed.

Since, it is not possible to learn all possible gray patterns due to its astronomical size; the MLP has been learned only for binary patterns and then these learned weights are used in the proposed connectionist model. Several attempts has been made to use this model for graytone thinning. The methods used can be classified into two categories : *normalization* which can be *global*, *local* or *dynamic* and *thresholding* which can also be *global*, *local* or *dynamic*.

# Chapter 2

# Modification of PGTA

## 2.1 Introduction

In a recent paper, a thinning algorithm which is characterized as "A parallel graytone thinning algorithm (PGTA)" is presented by Kundu et. al.[1] There are some concerns regarding this algorithm which bring into question its utility as a parallel graytone thinning algorithm. In this chapter, it is shown that the PGTA is even not able to ensure protection of skeletal points and as a consequence it is not able to preserve the graylevel connectivity in the image. Also, there is no proper stopping criteria for the iteration. All these problems arises only because of improper selection of threshold.

In this chapter, we have proved that the candidate pixel's graylevel is the unique value which can be used for thresholding its window; and, if used it will be able to eliminate all the above mentioned problems, effectively. In fact, there is no need to generate a thresholded window explicitly, in this case. Also, using Zhang-Wang algorithm[11] instead of Modified Zhang-Suen algorithm[4, 5] a more efficient(almost twice as efficient as PGTA) and generalized version of parallel graytone thinning algorithm named as Modified PGTA has been presented. Also, a comparative study has been done to check capabilities of PGTA and Modified PGTA in case of noisy images.

## 2.2 Problems associated with PGTA

### Deletion of Skeletal Points

If we take mean of the 8 direct neighbors graylevels as threshold $T$, then the skeletal points cannot be protected. Consider, some examples as shown in figures 2.1-2.3. Note that in figure 2.1, the candidate pixel with graylevel 9 is a single point gets deleted. Similarly, in figure 2.2, the candidate pixel with graylevel 9 is a connection point(for pixels $P_5$ and $P_7$) as well as an end point(for graylevel 9) gets deleted. It can also be noted that in both these cases $T < P_1$. Let us consider another example as shown in figure 2.3. In this case where $T > P_1$ then also the candidate pixel $P_1$ with

| | | |
|---|---|---|
| 5 | 5 | 5 |
| 0 | 9 | 0 |
| 1 | 0 | 1 |

(i)

| | | |
|---|---|---|
| 5 | 5 | 5 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

(ii)

Figure 2.1: (i) Given a $3 \times 3$ image window(with background graylevel = 0), (ii) Output of PGTA: The candidate pixel which is a single point, gets deleted. Here $T < P_1$.

| | | |
|---|---|---|
| 5 | 9 | 5 |
| 0 | 9 | 0 |
| 1 | 0 | 1 |

(i)

| | | |
|---|---|---|
| 5 | 9 | 5 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

(ii)

Figure 2.2: (i) Given a $3 \times 3$ image window(with background graylevel = 0), (ii) Output of PGTA: The candidate pixel which is a connection point as well as well as an end point, gets deleted. Here, $T < P_1$.

graylevel 2 which is a skeletal point gets deleted. In each of all the three cases, it can also be seen that the deletion of the candidate pixel divides the given window into 3 components which was originally an 8-connected single component. Thus, PGTA cannot preserve connectivity in all these cases. It has been suggested to reduce the upper limit of $B(P_1)$ from 6 to 5; but it can also not serve the purpose.

Also, due to this threshold problem, the basic image features which we want to protect, may completely destroy after certain number of iterations. e.g., see figure 2.4 where two branches of skeleton with higher graylevel degenerates to lower graylevelled branches. Thus, the mean of direct neighbors is not at all suitable for use as a threshold value.

## Improper Terminating Condition

In PGTA it has been suggested to terminate processing when the number of graylevel changes from $(k-1)\_th$ to $k\_th$ iteration is less than a predetermined value $n$. Un-

| | | |
|---|---|---|
| 7 | 7 | 7 |
| 0 | 2 | 0 |
| 2 | 0 | 2 |

(i)

| | | |
|---|---|---|
| 7 | 7 | 7 |
| 0 | 0 | 0 |
| 2 | 0 | 2 |

(ii)

Figure 2.3: (i) Given a $3 \times 3$ image window(with background graylevel = 4), (ii) Output of PGTA: The candidate pixel which is a connection point gets deleted. Here, $T > P_1$.

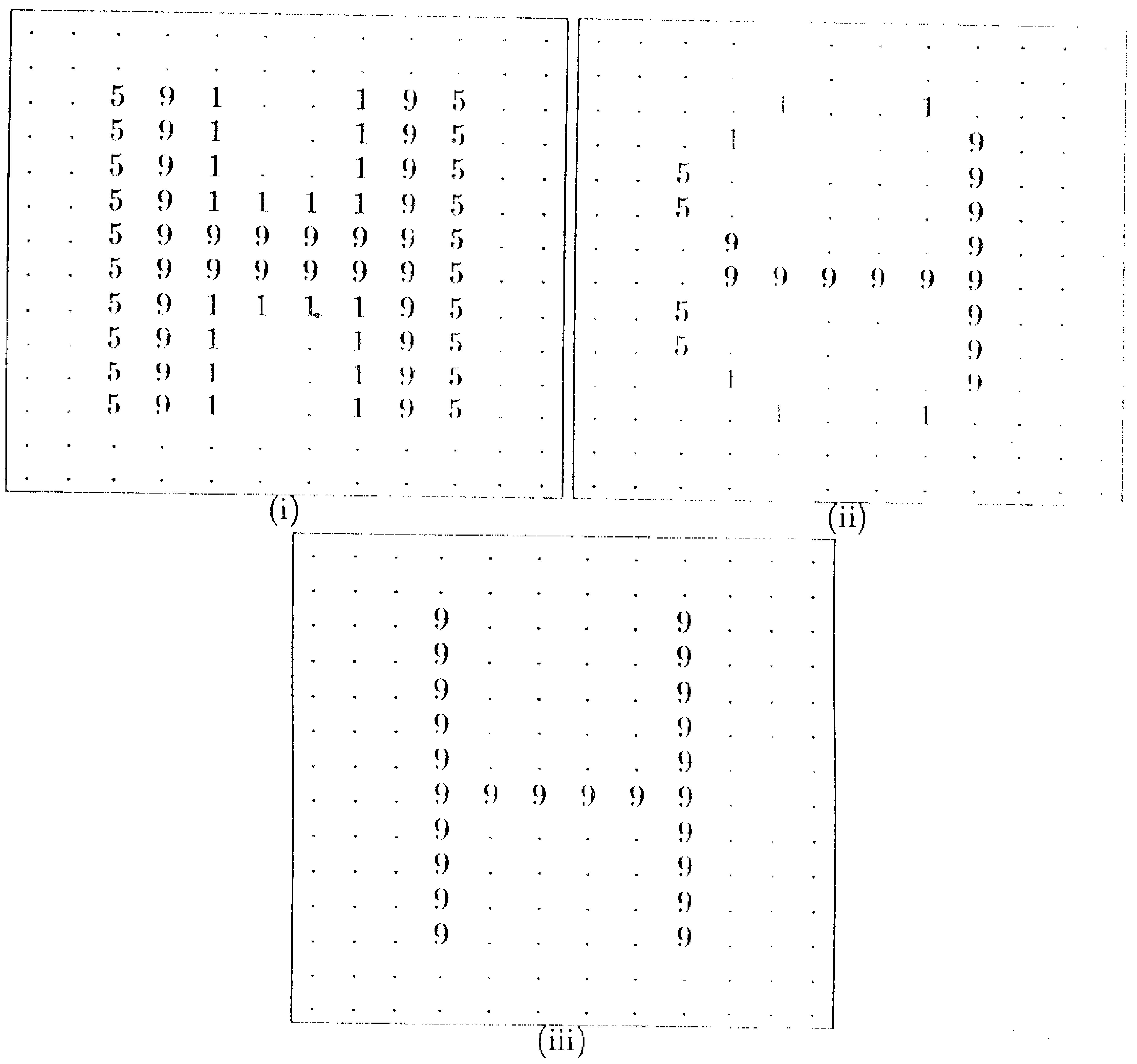


Figure 2.4: (i) Given an input image of character 'H' where all dots indicate background pixels with graylevel 0. (ii) Output of PGTA. (iii) Output of modified PGTA.

fortunately, $n$ depends not only on size but also on the texture of the image, and one cannot find an optimal value of $n$ for an arbitrary image in reasonable time, whatever heuristics he or she will use. In PGTA paper it is not mentioned why we should take such $n$; why not $n = 0$ always. One probable reason we feel may be the destruction of basic image features as pointed out earlier in this section.

## 2.3 Proposed Modifications

### Modification of Threshold

As pointed out in section 2.2, the mean of direct neighbors is not at all suitable for use as a threshold value. From the examples given in this section 2.2, it can also be observed that when $T \neq P_1$ the skeletal points cannot be preserved. On the other hand, if we take graylevel of $P_1$ as threshold and threshold all pixels in the given window using equation

$$P_i' = \begin{cases} 0 & \text{if } P_i < P_1 \\ 1 & \text{otherwise.} \end{cases} \tag{2.1}$$

then these skeletal points can be preserved. Based on these observations we have claimed that the graylevel of the candidate pixel is the unique value which can be used as threshold. The claim and its formal proof are as given below:

**Claim 1** *The graylevel of the candidate pixel in a given window is the unique value which can be used as threshold for thresholding the window preserving connectivity between any pair of points in the local neighborhood of the candidate pixel. In other words, if we take $T = P_1$ then even if graylevel of $P_1$ will get replaced by $min(P_2, P_3, \ldots, P_9)$, it will not disconnect any pair of points which were connected initially, within its window.*

***Proof:*** The most fundamental requirement of any thinning algorithm is that it must preserve connectivity. Also, it is necessary as well as sufficient to protect skeletal points of the image from getting deleted to preserve connectivity.

Let us consider 3 situations: $T < P_1$; $T = P_1$; and $T > P_1$.

When $T < P_1$ there will be more number of paths connecting $P_1$ to other points in the thresholded window than what is actually in the original gray image window. i.e. in a given window if there exists a $P_i$ such that $T < P_i < P_1$ then there can be no path from $P_1$ to a pixel $P_j \geq P_1$, for some $j$, via $P_i$. But, in thresholded window such pseudo paths can exist. This overestimation of connectivity is undesirable. If $P_1$ is a skeletal point in the original image window, it may become a simple point in its thresholded counterpart and if all the deletion conditions of PGTA get satisfied then this skeletal point's graylevel will be changed to the minimum of its 8 direct neighbors. Thus, $T < P_1$ does not ensure protection of skeletal points (see figures 2.1 and 2.2).

Also, when $T > P_1$ with similar arguments we can say that there will be less number of paths than what actually it should have. This underestimation of connectivity is also undesirable; because, it can also transform a skeletal point of the original image window into a simple point in it's thresholded counterpart. Finally, if all the deletion conditions get satisfied then this skeletal point's graylevel will be changed to minimum of its 8 direct neighbors. Since, thinning permits only deletion of simple points, this is against the definition of thinning (see figure 2.3).

Now, if we take $T = P_1$ then there will be exactly same number of connection paths from $P_1$ to any point $P_i$ in the thresholded window as in the original image window of the candidate pixel $P_1$. All pixels within the window with graylevel $\geq P_1$ will be thresholded to 1 and with graylevel $< P_1$ will be thresholded to 0. In this case, if $P_1$ is a skeletal, simple or internal point, it will remain skeletal, simple, or internal point respectively, in its thresholded window.

Thus, the candidate pixel $P_1$ is the only threshold which can protect the skeletal points and hence, preserves the connectivity. $Q.E.D.$

Now, if we take graylevel of $P_1$ as threshold and threshold all pixels in the window using equation 2.1, then for $P_1 = min(P_1, P_2, \ldots, P_9)$, the candidate pixel will become an internal point in the thresholded window and all conditions of PGTA will become $FALSE$. So, there is no need to explicitly check whether $P_1$ is minimum or not. Also, since we need not to calculate $T$, there is no need to explicitly generate a thresholded window. We can directly use the image window itself

### Modification of Terminating Condition

If we take $T = P_1$, there is no need of taking some predetermined value n for terminating condition. We can always take $n = 0$. Also, there is no need of counting number of changes in graylevel; instead of that, a *flag* can be used to indicate change in graylevel.

### Elimination of Two Subiterations

The PGTA is a generalization of modified Zhang-Suen algorithm and hence each iteration consists of two subiterations. Now, an algorithm which does not contain any subiteration inside a iteration and is almost twice as efficient as the modified Zang-Suen algorithm has already been developed by Zhang and Wang, for two tone images. So, generalization of this algorithm can be used as PGTA.

Thus, with these modifications, we have presented the proposed algorithm, named as modified PGTA, in the next section.

## 2.4 The Modified PGTA

For each candidate pixel $P_1$, consider a $4 \times 4$ window as shown in figure 1.2. The graylevel of a candidate pixel $P_1$ will be changed to $min(P_2, P_3, \ldots, P_9)$, if all the following conditions are satisfied :

1. $3 \leq B(P_1) \leq 6$
2. $A(P_1) = 1$
3. $(P_2 < P_1) * (P_4 < P_1) * (P_8 < P_1)$ $or$ $(P_{11} \geq P_1)$

4. $(P_2 < P_1) * (P_4 < P_1) * (P_6 < P_1)$ or $(P_{15} \geq P_1)$

**where,** $A(P_1)$ = Number of $(P_i, P_{i+1})$ patterns in the ordered set $P_2, P_3, \ldots, P_9, P_2$ such that $P_i < P_1$ and $P_{i+1} \geq P_1$.
$B(P_1)$ = Number of direct neighbors of $P_1$ having graylevel $\geq P_1$, and
All $*$'s denote logical ANDs.

Also, a complete algorithm for simulating the Modified PGTA in a serial computer is given in Appendix A.

## 2.5 RESULTS

Both PGTA and modified PGTA have been simulated on SUN SPARC station as well as on VAX8650 under VAX/VMS operating system environment.

Both these algorithms have been tested on many real life images. Some of these images and their corresponding outputs from PGTA and modified PGTA have been shown in figures 2.5-2.8. The finger prints as shown in figure 2.5 and figure 2.6 are of size $(256 \times 256)$, the noisy cclamp as shown in figure 2.7 and the noise free cclamp as shown in figure 2.8, both are of size $(384 \times 256)$. One can visually see the effect of modifications. It can be easily seen in figure 2.7 that the effect of noise is quite different from each other. A discussion on noise handling capabilities of PGTA and modified PGTA and a method to preprocess such noisy images is as given below:

### Handling Noisy Images

In case of PGTA, it should be noted that due to averaging of direct neighbors for calculation of threshold, an smoothing effect occurs in the image. So, when the given input image is noisy then also it can thin the image upto certain extent, but not upto single pixel thickness; otherwise, important image features may get destroyed due to reasons discussed earlier. But, in case of modified PGTA, we always assumes that the given image is a noise free image. So, the modified PGTA can not handle a noisy image automatically. Indeed, We have to first remove noise from the image separately and then input for thinning.

It should be noted that smoothing is not a good method for noise removal, specially when the image is to be used for thinning. Because, by smoothing a small localized noise get dispersed into a large region, and if we thin such image then the effect of noise may be more pronounced.

A better method is thresholding. The image has two types of regions: background and object. Where, each of these regions have a range of graylevels and in graylevel thinning we want to thin the object keeping grayness of the object as well as background intact. So, the total graylevel range of the image can be divided into several subranges and only a few subranges (may be only one) can be thresholded to a suitable graylevel, say mean or mode of graylevels in that subrange. The mode may be a better measure in this case; because, total number of pixels affected by noise will generally be smaller than total number of unaffected image pixels. The minimum and maximum graylevels of a subrange, say $k\_th$ subrange of the image can be obtained using following two equations, respectively.

$$lgray = mingray + (k-1) * \frac{(maxgray - mingray)}{n} \tag{2.2}$$

$$ugray = mingray + k * \frac{(maxgray - mingray)}{n} \tag{2.3}$$

where, $mingray$ = minimum graylevel of the image.
$maxgray$ = maximum graylevel of the image.
$lgray$ = lower limit of the graylevel in the $k\_th$ subrange.
$ugray$ = upper limit of the graylevel in the $k\_th$ subrange.
n = Number of subranges and $1 \leq k \leq n$.

Then, the mean or mode of that subrange can be found out and all pixels graylevel in that subrange can be thresholded to that graylevel.

## 2.6 Discussions

In this chapter, a thorough study of PGTA has been carried out. It has been proved that the graylevel of the candidate pixel is the unique value which can be used as threshold. Also, terminating condition has been modified and a more efficient version of PGTA has been presented.

In image processing jargon there is a saying, "What you see is what you get". But, human visual system is not powerful enough to recognize very small variations. Not only this, but, the display systems commercially available at present are also not having sufficient resolution for distinctly displaying each of the graylevel of the image. Due to these reasons, it is very difficult to recognize small variations of graylevel caused by noise. So, one may claim that the given image is a very nice image(in the sense that it is a visually noise free image), and expect that the thinned image should also be a single pixel thick image in the visual sense. But, due to noise, he/she will get a single pixel thick image in mathematical sense(where actual graylevel is of concern) which may not be visually single pixel thick and hence, a preprocessing for noise removal is required.

As far as I know, there is no robust method known till today which can automatically handle a noisy image. Further research can be carried out in this area to develop such thinning algorithms.

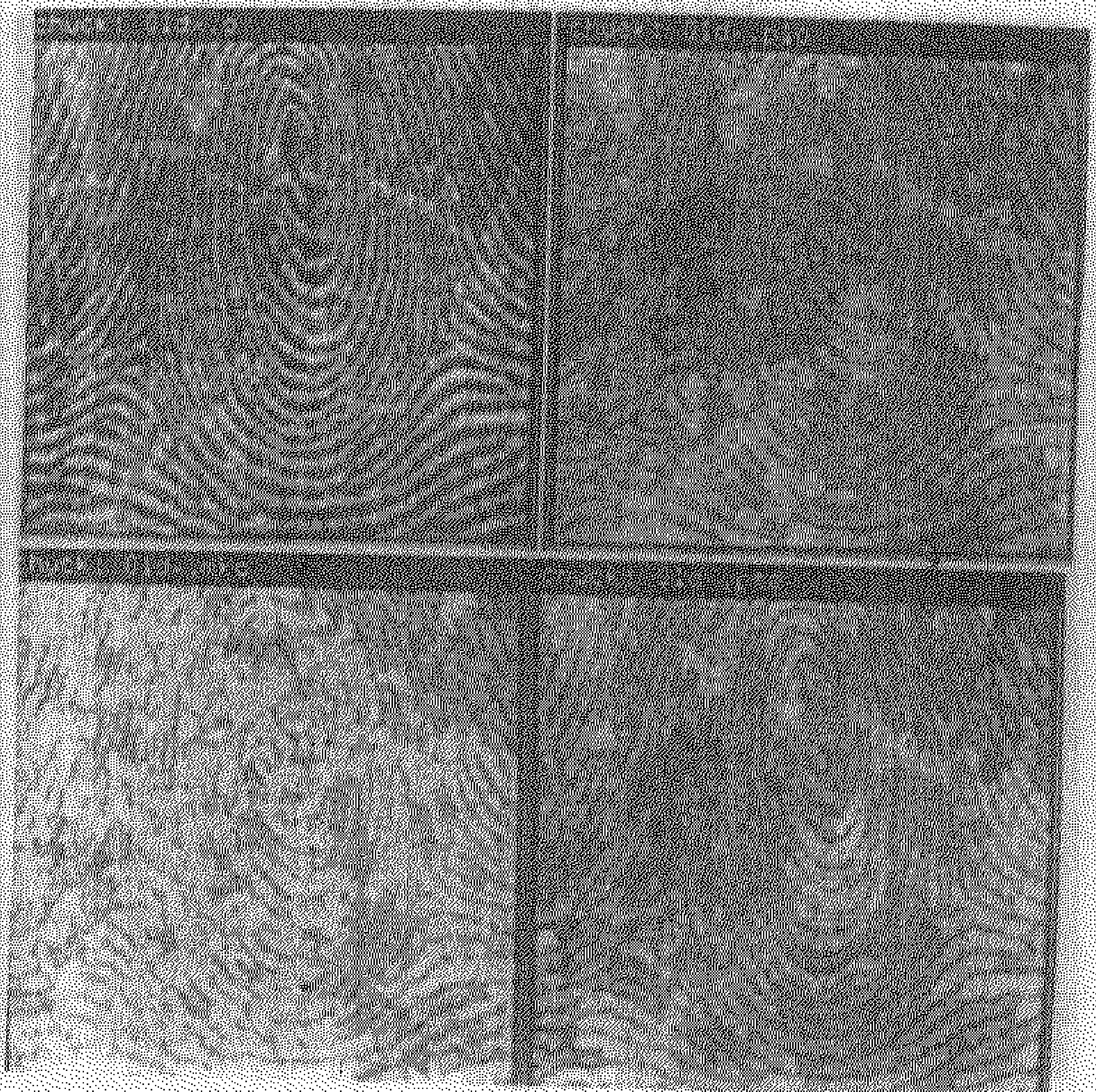


Figure 2.5: finger print A: (i) Input image, (ii) output of PGTA after 10 iterations for $n = 0$, (iii) output of PGTA at the end for $n = 0$, (iv) output of modified PGTA.

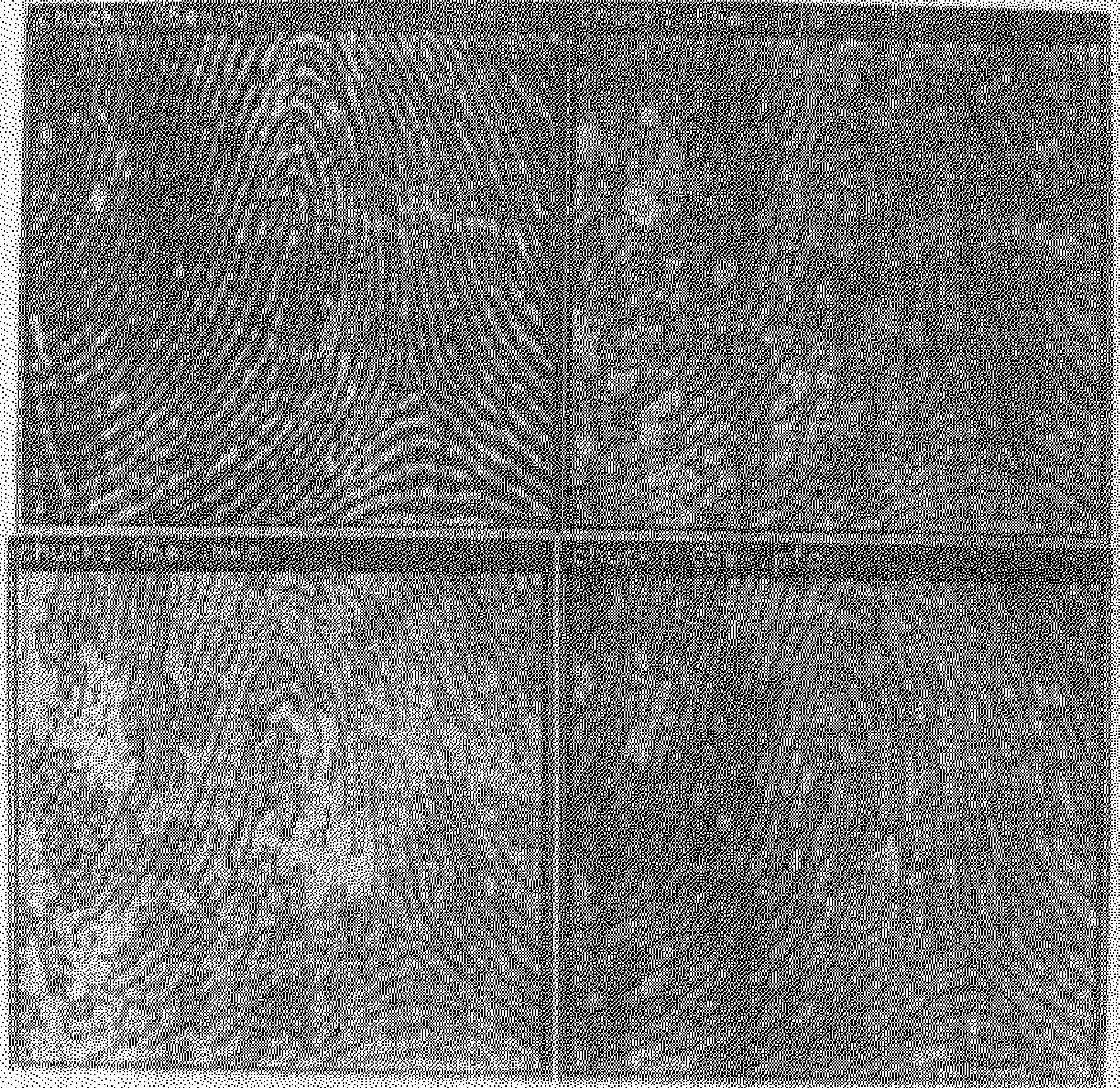


Figure 2.6: finger print B: (i) Input image, (ii) output of PGTA after 10 iterations for $n = 0$, (iii) output of PGTA at the end for $n = 0$, (iv) output of modified PGTA.

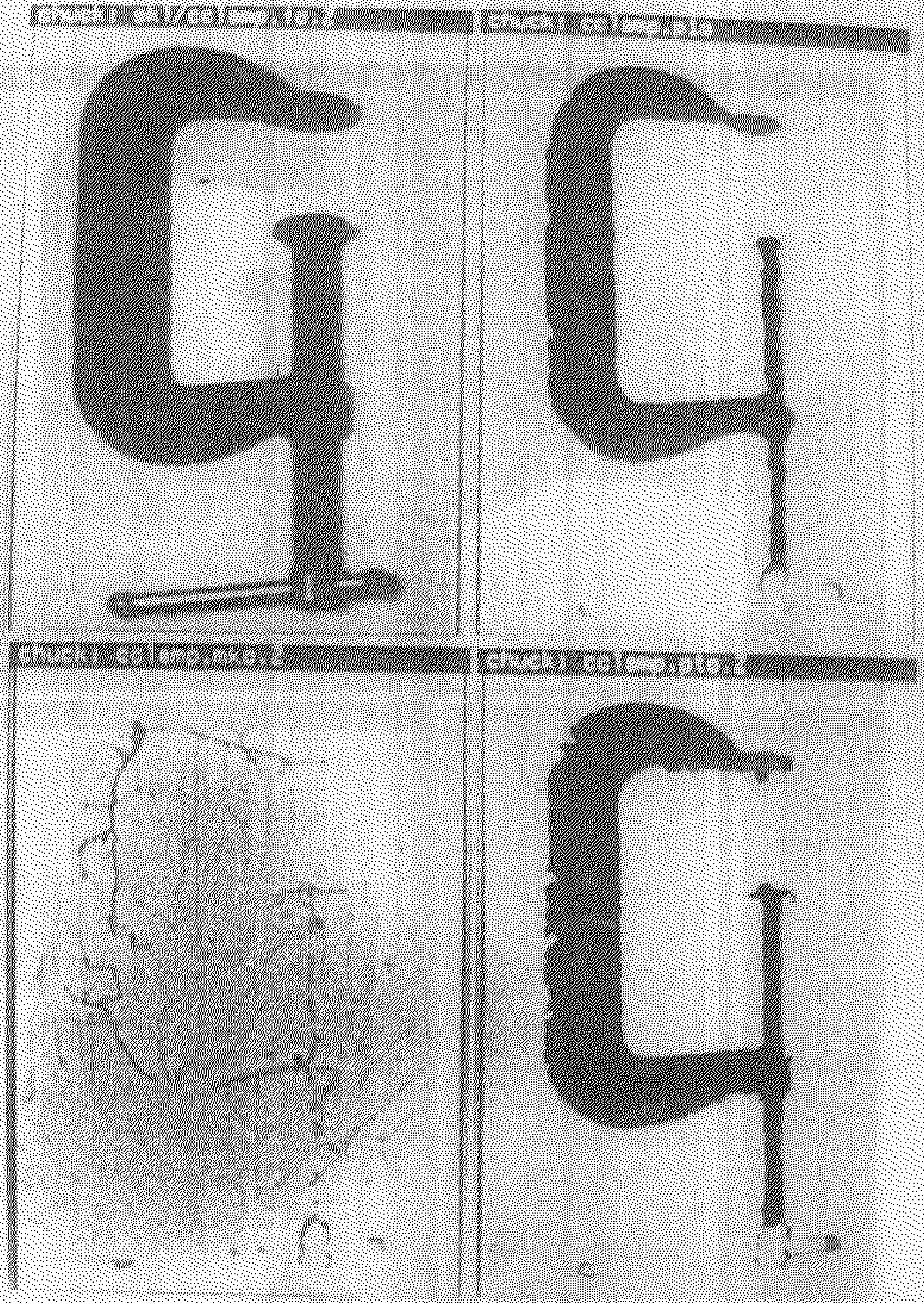


Figure 2.7: A noisy cclamp: (i) Input image, (ii) output of PGTA after 10 iterations for $n = 0$, (iii) output of PGTA at the end for $n = 0$, (iv) output of modified PGTA.

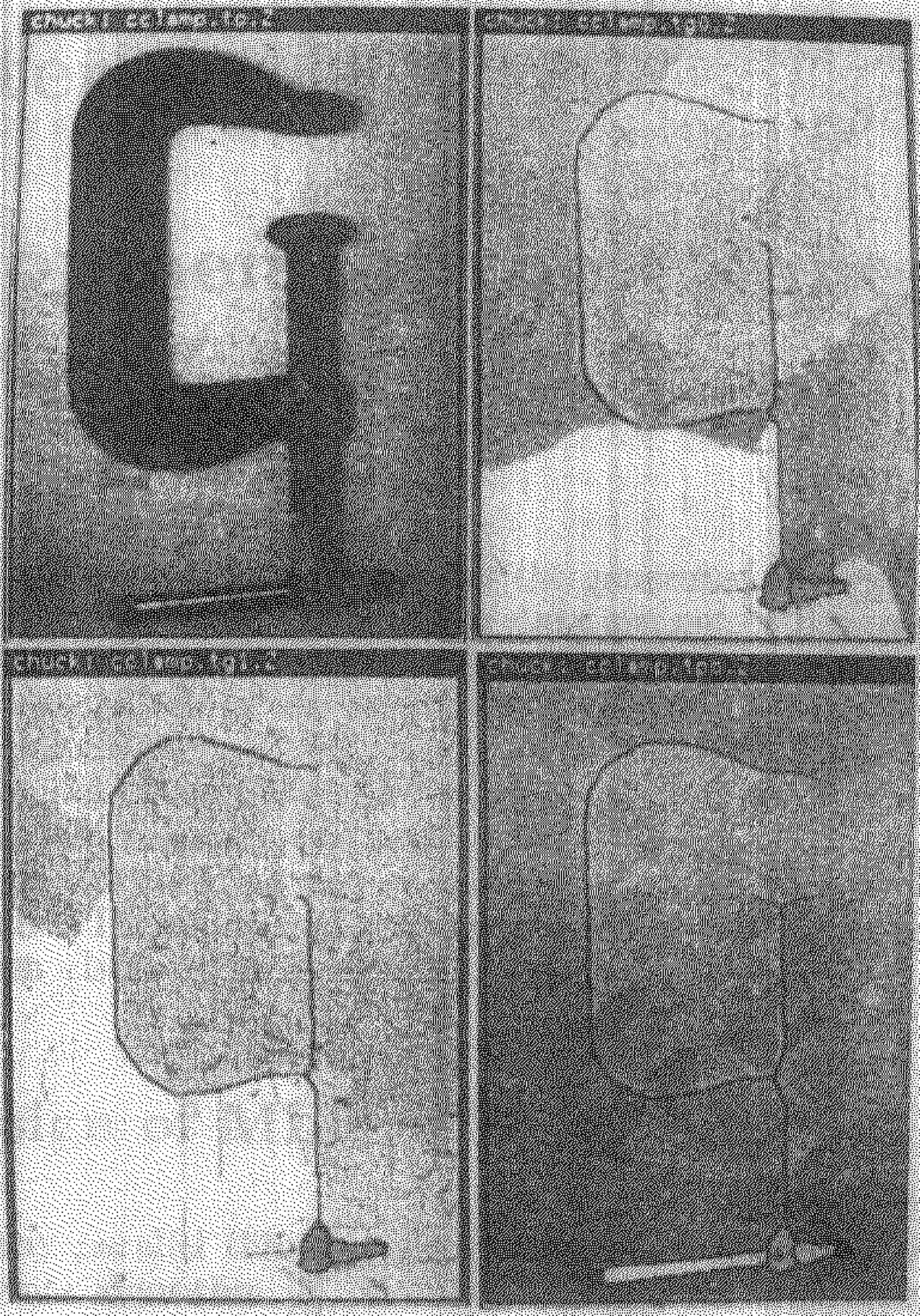


Figure 2.8: A noise free cclamp image obtained after preprocessing: (i) Input image, (ii) output of PGTA after 100iterations for $n = 0$, (iii) output of PGTA at the end for $n = 0$, (iv) output of modified PGTA.

# Chapter 3

# Development of the Connectionist Model

As discussed in section 1.4, we know that the generated binary training data can be used for supervised learning, the training data are not orthogonal to each other and it is not possible to learn all gray patterns. Based on these observations, we found that an MLP can be one of the most suitable model for forming the basic building block. So, in this chapter, we will first design the MLP as the basic building block in the first section, and then the connectionist model will be developed in the following section. There are basically two way of mapping graylevels of an image into [0,1] range; these are thresholding and normalization. Also, each of them can be global, local or dynamic. These different strategies used for thinning using the Connectionist Model has been discussed in section 3.3.

## 3.1 Design of MLP as Basic Building Block

There are many parallel thinning algorithms which generally use a $3 \times 3$ window such as Rutowitz algorithm [3], Modified Zhang-Suen algorithm [4, 5], Holt et al. algorithm [6] etc. or a $4 \times 4$ window such as Zhang-Wang algorithm [11]. If we consider a window of size $N \times N$ for $N \geq 3$ then there can be at most $N^2$ inputs and so, at most $N^2$ input nodes will be required in the ANN model to be developed for this window. Also, There will be only one output node which will give either a new graylevel for the candidate pixel itself or will give an indication about the deletion(or reduction of the graylevel) of the candidate pixel. This ANN model for a window will form the basic building block.

Only binary patterns has been considered for learning with graylevel of a background pixel as 0 and graylevel of an object pixel as 1. the input patterns can be roughly divide into three clusters: one containing those patterns in which the candidate pixel is 0 and hence output of the MLP will also be 0. The second containing those patterns in which the candidate pixel is 1 and do not satisfy all the deletion conditions; so, the output of the MLP will remain 1. The third and last class may contain all those patterns for which the candidate pixel is 1 and satisfy all the deletion conditions so that the output of the MLP will become 0.

Given an n-layer MLP, some authors treats it as an MLP with n layers of nodes excluding input layer, while the other treats as n layers of weight links. Through out the discussion, we will follow this convention. e.g., a 3-layer MLP means it consists of one layer of input nodes, one layer of output nodes, 2 layers of hidden nodes and 3 layers of weight links connecting the adjacent node layers.

If it so happens that our all 3 clusters turn out convex(which is very unlikely) then a two-layer MLP with only 3 hidden nodes is sufficient for proper learning. So, the lower limit for the network architecture is a two-layer MLP with 3 hidden nodes. On the other hand, proper classification of given training data into 3 clusters of arbitrary shape, in worst case, requires a 3-layer MLP with number of nodes in the second hidden node layer greater than or equal to the number of clusters and number of nodes in the first hidden node layer equal to twice the number of nodes in the second hidden node layer. Thus, the upper limit for the network architecture in present case is a 3-layer MLP with first hidden node layer consisting of 6 nodes and the second hidden node layer consisting of 3 nodes.[20]

The training data for learning has been generated using conditions of only one subiteration of the Modified Zhang-Suen thinning algorithm viz. a candidate pixel $P_1$ in a $3 \times 3$ window, as shown in figure 1.1, will be deleted only when all the following conditions are satisfied:

1. $P_1 = 1$
2. $3 \leq B(P_1) \leq 6$
3. $A(P_1) = 1$
4. $P_2 * P_4 * P_8 = 0$
5. $P_2 * P_4 * P_6 = 0$

where, $A(P_1)$ = Number of 01 patterns in the ordered set $P_2, P_3, \ldots, P_9, P_2$.
$B(P_1)$ = Number of nonzero neighbors of $P_1$, and
all $*'s$ denote logical ANDs.

Practically, it is found that with a two-layer MLP model, having 3 or 4 hidden nodes, the rate of convergence becomes very very slow and after sometimes error becomes almost stable. With 5 hidden nodes also the rate of convergence is not so good. But, for 6 onward the rate of convergence is reasonably fast and hence, a two-layer MLP with 6 hidden nodes can be used as the Basic Building block. With 6 hidden nodes, it is also not necessary to learn threshold for individual nodes. All thresholds can be tied to 0. The MLP model used for learning is as shown in figure 3.1. It is basically a semilinear feedforward net as reported by Rumelhart et. al.[22] For learning *The Generalized Delta Rule with Backpropagation of Error*, as described in [21, 26] has been used.

Where, the net input to a node in layer $j$ i.e., in the hidden layer is given by

$$net_j = \sum w_{ji} o_i \tag{3.1}$$

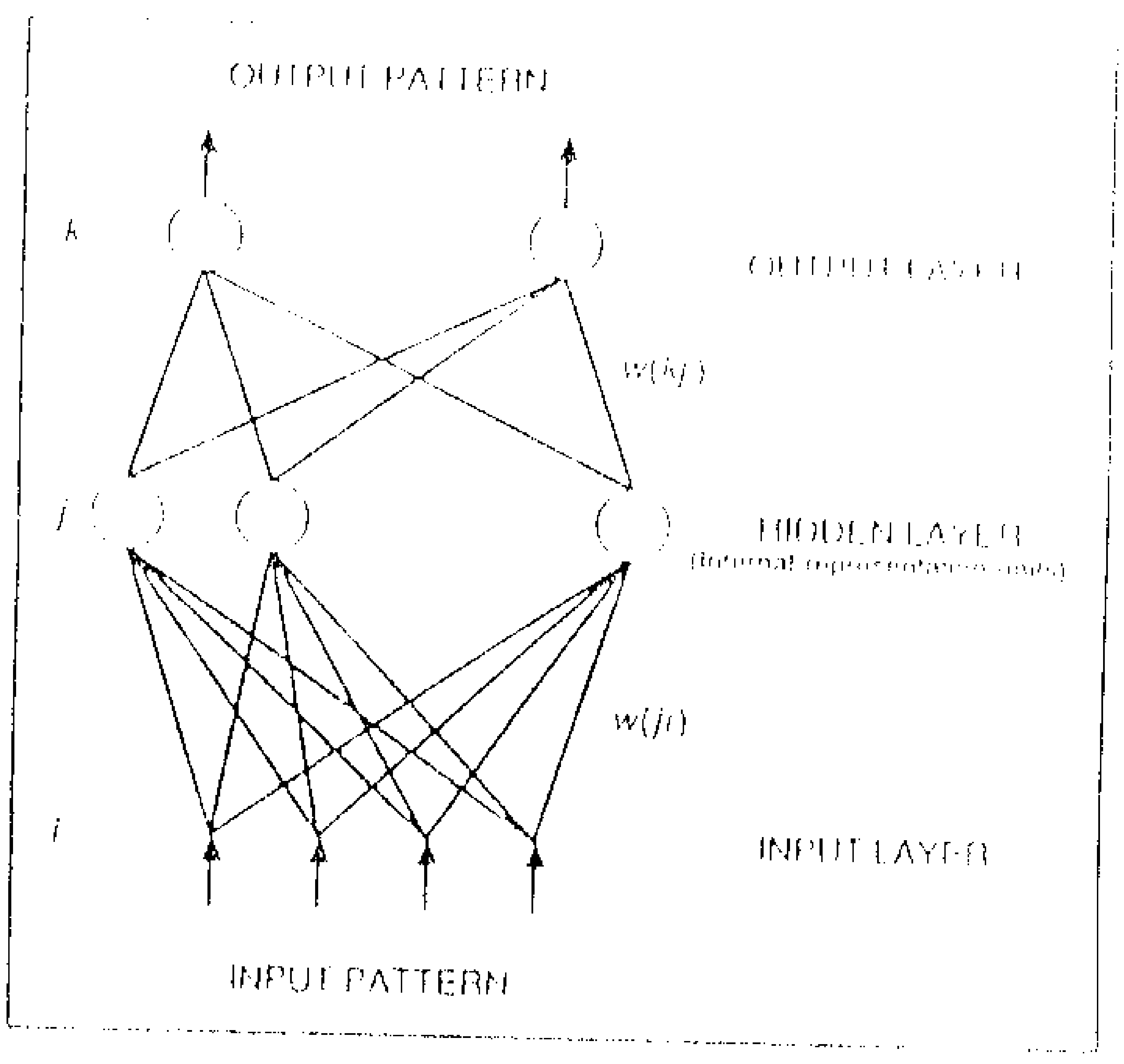


Figure 3.1: MLP: The basic building Block. We have used 9 input nodes, 6 hidden nodes and 1 output node during learning the training data generated by one subiteration of the modified Zang-Suen Algorithm.

and the corresponding output is given by

$$o_j = f(net_j) \tag{3.2}$$

where, $f$ is the activation function. We have used a sigmoidal activation function as given below:

$$o_j = \frac{1}{1 + e^{-(net_j + \theta_j)/\theta_o}} \tag{3.3}$$

Similarly, the net input to a node in layer $k$ i.e., in the output layer is given by

$$net_k = \sum w_{kj} o_j \tag{3.4}$$

and the corresponding output is given by

$$o_k = f(net_k) \tag{3.5}$$

In equation 3.3, the parameter $\theta_j$ serves as a threshold for that node. In present case, it has been taken as 0 all nodes. The parameter $\theta_o$ is a constant and has been taken as 0.1. But, it hardly matters, one can take it as 1; the only thing happen is that the weights will become 10 times larger than what is given in table 3.1.

The average system error is given by

$$E = \frac{1}{2P} \sum_p \sum_k (t_{pk} - o_{pk})^2 \tag{3.6}$$

and, the weight change from $n\_th$ iteration to $(n+1)\_th$ iteration is given by

$$\Delta w_{ji}(n+1) = \eta(\delta_j o_i) + \alpha \Delta w_{ji}(n) \tag{3.7}$$

where, $\eta$ is the learning rate,
$\alpha$ is the momentum rate, and
the deltas are given by the following two equations

$$\delta_{pk} = (t_{pk} - o_{pk}) o_{pk} (1 - o_{pk}) \tag{3.8}$$

$$\delta_{pj} = o_{pj}(1 - o_{pj}) \sum_k \delta_{pk} w_{kj} \tag{3.9}$$

for the output-layer and hidden-layer nodes, respectively. The subscript p denotes the pattern number.

## 3.2 The Connectionist Model

Block diagram of the Connectionist Model is as shown in figure 3.2 and a detailed diagram showing input, hidden and output layers and their interconnections is as shown in figure 3.3. It can be build using the basic building block as follow. To input an image of size $(M \times N)$ a two-dimensional grid containing $(M \times N)$ nodes will be required. Similarly, a grid of same size is required for outputting the result.

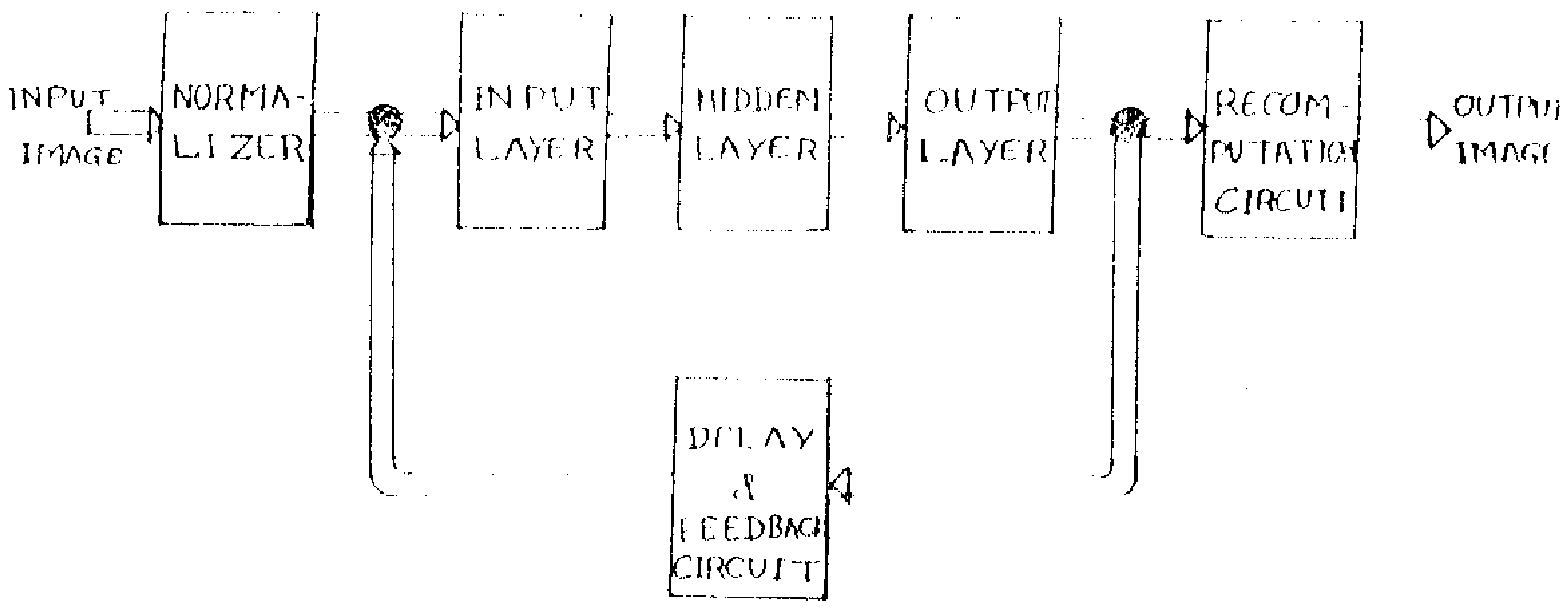
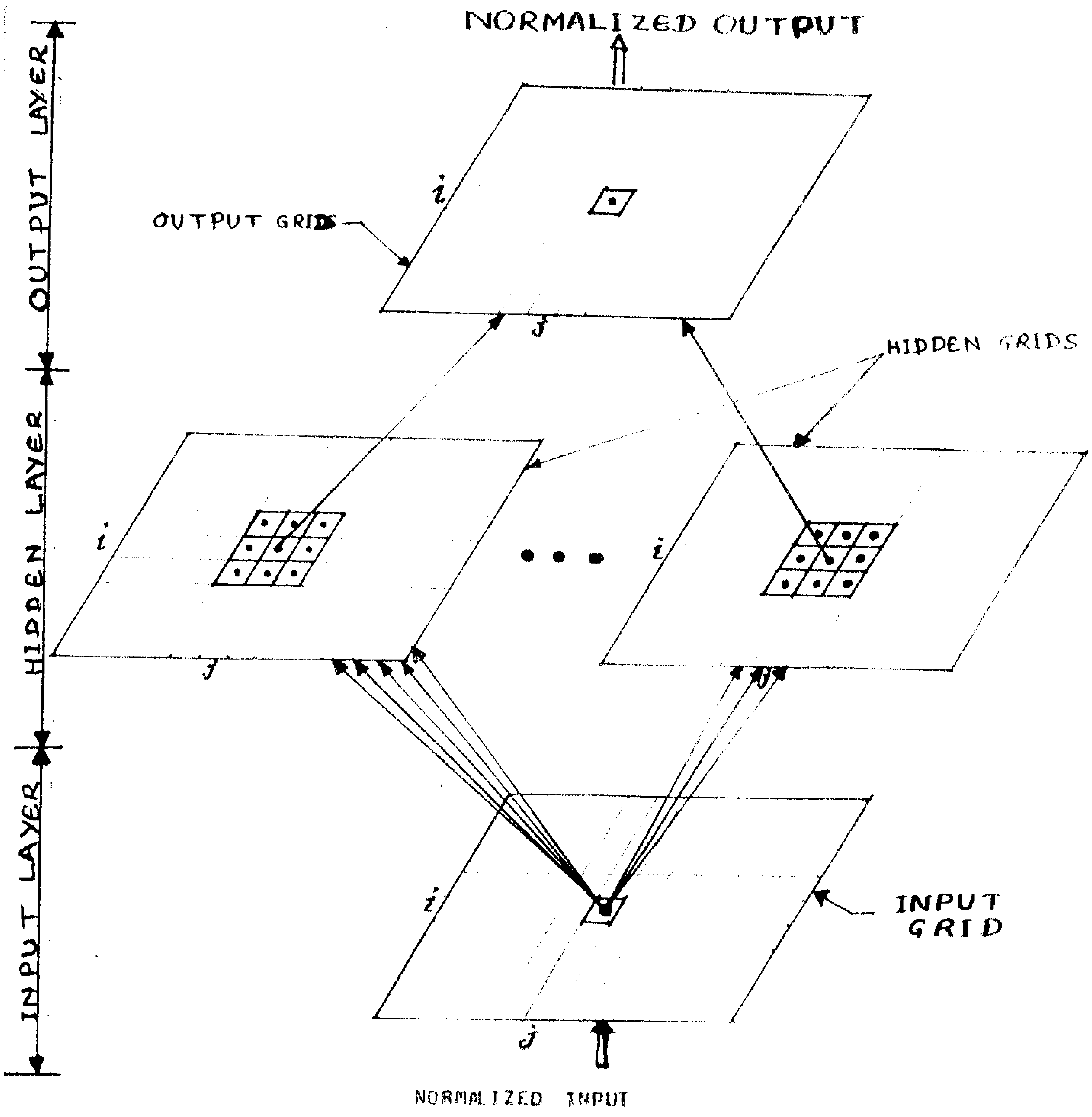


Figure 3.2: Block diagram of the Connectionist Model for thinning.

Since, in figure 3.1, each input node is connected to all the hidden nodes, if there are $h$ nodes in the hidden layer of the basic building block, then either $(M \times N)$ grids each consisting of $h$ nodes or $h$ grids each consisting of $(M \times N)$ nodes can be taken. Since, the second alternative gives uniformity in the architecture as well as there will less number grids to manage, in this case; it is a better choice. So, in the Connectionist Model $h$ grids each consisting of $(M \times N)$ nodes has been used to form the hidden layer.

The weight connections will be as follow: If a $(n \times n)$ window is used for learning the basic building block, then for outermost $\lfloor \frac{n}{2} \rfloor$ rows and columns of the grid, all neighborhood pixels are not available. So, for these boundary nodes a 1-1 connection should be made. i.e., connect a boundary $node(i, j)$ in the input grid to the corresponding boundary nodes in all the hidden layer grid and connect the boundary $node(i, j)$ in all these hidden layer grids to the output boundary $node(i, j)$. All connection weights of these connection links are equal to unity.

In case of internal nodes, connect an internal $node(i, j)$ in the input grid to the corresponding $node(i, j)$ in the hidden grid $h_1$ by a connection link having weight equal to the weight of the link connecting $P_1$ to $h_1$. Also, connect all the neighbors within the $(n \times n)$ neighborhood (because, we have taken $(n \times n)$ window for learning) of the internal $node(i, j)$ in the input grid to the $node(i, j)$ of the hidden grid $h_1$ by weight links having weights equal to the weight of the corresponding link in the basic building block. e.g. weight of the link between the input node $node(i-1, j)$ and the internal $node(i, j)$ of hidden grid $h_1$ is equal to weight of the link connecting $P_2$ to $h_1$ in the basic building block. Similarly, weight of the link between the input node $node(i-1, j+1)$ and the internal $node(i, j)$ of hidden grid $h_1$ is equal to weight of



Eachinput node (i,j) is connected to node (i,j) and its direct neighbors in each of the hidden grids. Also, each node (i,j) in each of the hidden grid is connected to output node (i,j).

Figure 3.3: A detailed diagram of input, hidden and output layers of block diagram of the Connectionist Model for thinning.

the link connecting $P_3$ to $h_1$ in the basic building block, and so on. Also, the internal *node*$(i,j)$ of the hidden grid $h_1$ will be connected to the internal *node*$(i,j)$ of the output grid by a weight link having weight equal to weight of the link joining node $h_1$ to the output node in the basic building block.

Similar connections will be made for each of the internal *node*$(i,j)$ in the input and the output grids with all hidden grids using weights of the corresponding weight links in the basic building block.

The result obtained from this network by processing image from input grid to output grid will be equivalent to the result that can be obtained from the thinning algorithm used for learning, after one iteration. In case of learning using training data generated by Modified Zang-Suen thinning algorithm, where, each iteration consists of 2 subiterations, it is equivalent to the result that can be obtained after one subiteration.

So, the result has to be feedback to the input grid for further processing. In case, when Zang-Suen like thinning algorithms used for generating training data, where each iteration consists of 2 subiterations, the transpose of the output image will be feedback to the input. i.e., an output *node*$(i,j)$ will be fedback to the input *node*$(j,i)$, for all $i,j$. In all other cases where the thinning algorithm for generating training data consists of only one iteration, feedback the output directly to input. i.e., feedback output of an output *node*$(i,j)$ to the input *node*$(i,j)$.

## 3.3 Thinning Using the Proposed Model

one approach for thinning a graylevel image is to first convert it into a binary image by suitable thresholding and then apply any binary level thinning algorithm to get the result. The same approach can be adopted here. First threshold the given image and then use the MLP for thinning this thresholded two-tone image. The thresholding can be done globally or locally. But, the problem associated with this method is that the information other than the object outline is lost forever and further processing in the graytone domain is not possible. Also, the method is very sensitive to noise [1, 8]. All these shortcomings could be eliminated to a great extent if the thresholding is done dynamically by thresholding only the window to be applied into the input of MLP, just before applying it. Then separately check whether the value of the candidate pixel has been changed or not. If changed then replace the candidate pixel's value by minimum of its direct neighbors [1]. In this method, we will not only be able to thin the image but we can also retain the grayness of the image.

Normalization based methods is rather more suitable for graylevel thinning. In case of global normalization, we can first normalize the whole image in [0,1] range using the equation:

$$P_i^n = \frac{P_i - mingray}{(maxgray - mingray)} \tag{3.10}$$

where, $P_i^n$ = Normalized graylevel of a pixel $P_i$ in the given image,
$mingray$ = minimum graylevel of the image and
$maxgray$ = maximum graylevel of the image.

Then thin this image by MLP. After thinning is over, the actual graylevel can be obtained by following equation:

$$P_i = P_i^n * (maxgray - mingray) + mingray + 0.5 \qquad (3.11)$$

Since, MLP has been learned only with the two-tone patterns, during thinning the normalized gray image will slowly change into two-tone thinned image, if allowed to terminate the processing voluntarily. So, one has to check after few iterations whether the desired thinning has been obtained or not. The main problem with global normalization is due to change in graylevel of the background itself. In real life images, the background may not be of single graylevel but may have some range. Similarly, the object also have a range of graylevels. Also, it may happen that the two ranges may overlap on each other upto certain extent. So, instead of global normalization, if we do some sort of local normalization then the result may be of better quality. This leads to the method of local normalization.

In case of local normalization, a candidate pixel is normalized over its small local neighborhood. e.g., In case of thinning using any $3 \times 3$ window method, we can take slightly larger window, say $5 \times 5$, for normalization. But, if all the pixels have same graylevel in this $5 \times 5$ window then a larger window, say $7 \times 7$ can be taken, and so on. After getting normalized image it is used for thinning and then the actual thinned image is computed back from this normalized thinned image. Note that during computation of the actual thinned image, the local *maxgray* and *mingray* values of the given input image will be used.

Since, the local *maxgray* and *mingray* values of a pixel in the input image may not be same as the local *maxgray* and *mingray* values respectively of that pixel in the corresponding thinned image that we are intended to obtain. As a consequence, it will result distortion in the actually computed image. This problem can be avoided, if normalization is done dynamically.

In case of dynamic normalization method, the candidate pixel's window is normalized locally just before applying it to the MLP, then actual new value of the candidate pixel is computed back immediately after getting the output of the MLP. This method has been found most suitable compared to others mentioned above, specially when we not only want to thin the given image but also want to control the grayness of the object as well as background. If the processing will be allowed to stop naturally at the end when there is no graylevel change, then the resulting thinned image will be a two-tone image.

## 3.4 RESULTS

Both, the MLP used as Basic Building Block and the Connectionist Model are simulated on SUN SPARC station as well as on VAX8650 under VAX/VMS operating system environment.

The weight matrix obtained by learning the training data generated using only one subiteration of the Modified Zang-Suen thinning algorithm with the MLP having 9 input nodes, 6 hidden nodes, 1 output node and all thresholds of tied at 0 is as shown in table 3.1. The input images used and its corresponding output images

| $h_1$ | $h_2$ | $h_3$ | $h_4$ | $h_5$ | $h_6$ |
|---|---|---|---|---|---|
| -0.723541 | -0.874873 | -0.871492 | 1.439476 | 1.196587 | -0.598361 |
| 0.533080 | -0.207651 | 0.365619 | 0.011588 | 0.496461 | -0.027020 |
| 0.760572 | -0.072775 | -0.015679 | 0.016497 | 1.231702 | 0.277667 |
| -0.192178 | 0.133114 | -0.578482 | 0.393233 | 0.800195 | 0.939326 |
| 0.254573 | 0.214549 | -0.772841 | -0.202602 | -0.062248 | -0.032122 |
| -1.035402 | 0.629234 | -0.350085 | -0.493559 | -0.425912 | -0.157464 |
| -0.834833 | -0.032003 | 0.035095 | -0.950506 | -0.986254 | 0.020174 |
| -0.425561 | 0.147288 | 0.397973 | -0.739835 | -0.172671 | -0.616102 |
| -0.165715 | -0.013634 | 0.526720 | 0.549618 | -0.479734 | -0.413329 |
| -2.781597 | -3.611564 | -5.332625 | 1.899429 | 2.633353 | -3.270761 |

Table 3.1: The Weight matrix : $h_i$ means hidden layer node $i$. The first 9 rows indicate weights between input nodes and hidden layer nodes and the last row indicates weight between hidden layer nodes and the output node.

obtained from the connectionist model are as shown in figures 3.4-3.7. These are two finger print images each of size ($256 \times 256$), a noisy cclamp ($384 \times 256$) and the same cclamp image ($384 \times 256$) after noise removal. The corresponding output images are obtained from the Connectionist model using normalization based method over a $3 \times 3$ window. The results obtained from PGTA and Modified PGTA can be compared with the result obtained from the Connectionist Model using dynamic normalization. Since, in case of dynamic normalization, the input image slowly converts into a two-tone thinned image, if you want to retain the grayness of the background as well as object, then it should be terminated after certain number of iterations. In most of the practical cases, actual thinning of the image happens only in few iterations, say 25-30 iterations. After that, only conversion of image from graytone to two tone takes place. So, it is a better practice to terminate the processing after 25 (say), and then check the result. If it is not satisfactory then go for another 25 iterations and so on. Otherwise, if it is satisfactory then stop.

It should be noted that if the transpose of the image is fedback to the input then the result obtained will be either the thinned image or transpose of it. This problem can be avoided by using a 5 stage Connectionist Model consisting one input layer, a hidden node layer, an intermediate result layer, another hidden layer and output layer. But, the problem associated with this method is the Cost of the Network. It will be almost as costly as twice that of the proposed Connectionist Model. A more elegant way to solve this problem is to output the result of the proposed Connectionist Model only after odd number processing cycles.

## 3.5 Discussions

In this chapter, a new Connectionist Model for Image Skeletonization has been proposed. There are basically two ways of preprocessing the input before inputting to the Connectionist Model. It has been found that the dynamic thresholding is the best method among global, local, and dynamic thresholding. Similarly, in case normal-

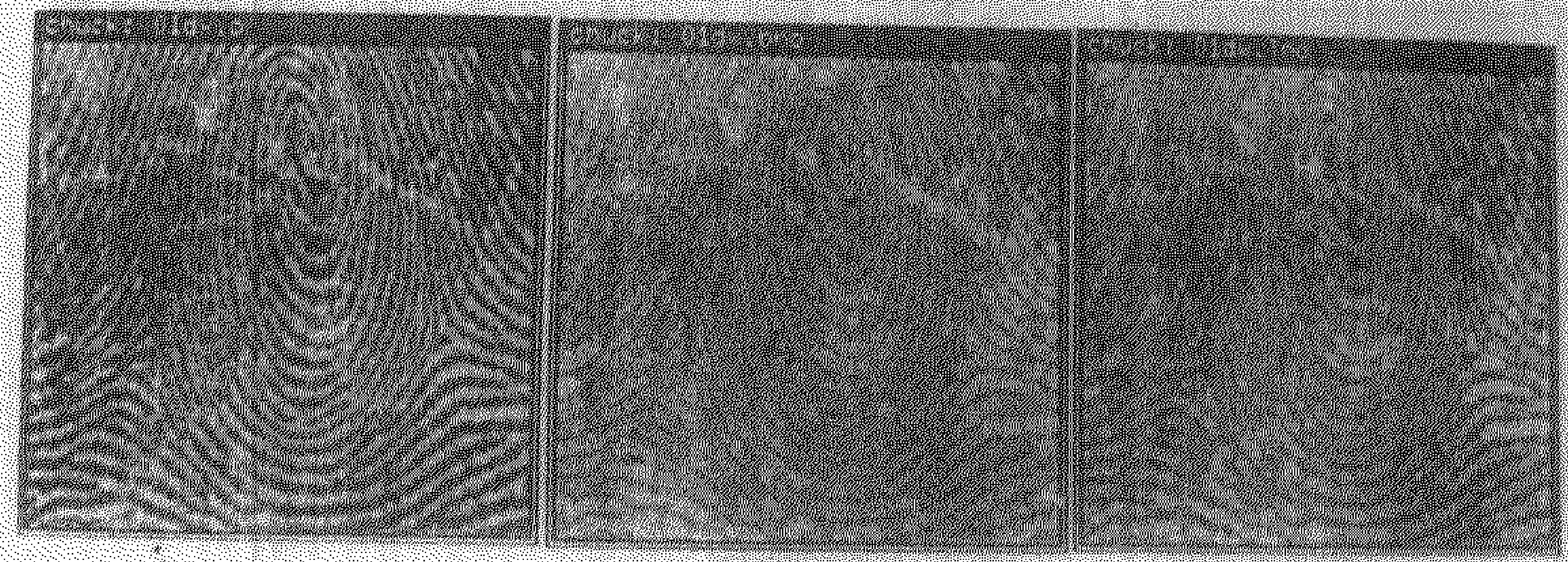


Figure 3.4: finger print A: (i) Input image, (ii) output of the connectionist model when: (a) dynamic normalization is used. (b) local normalization is used.

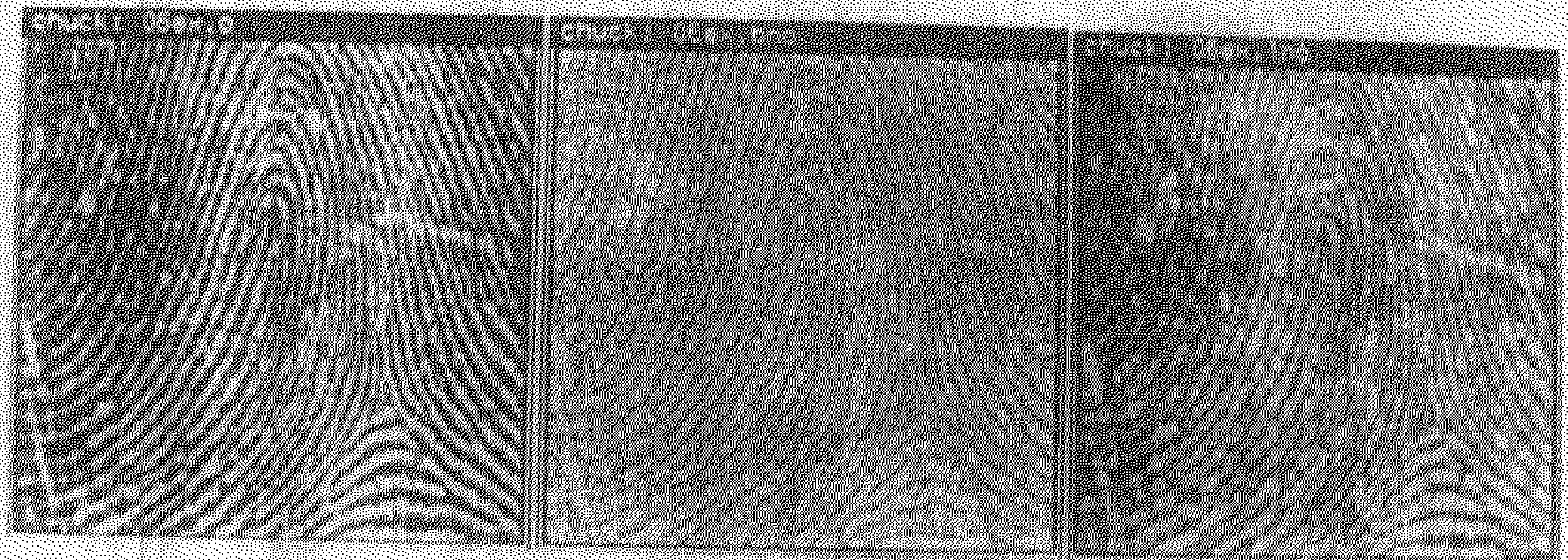


Figure 3.5: finger print B: (i) Input image, (ii) output of the connectionist model when: (a) dynamic normalization is used. (b) local normalization is used.

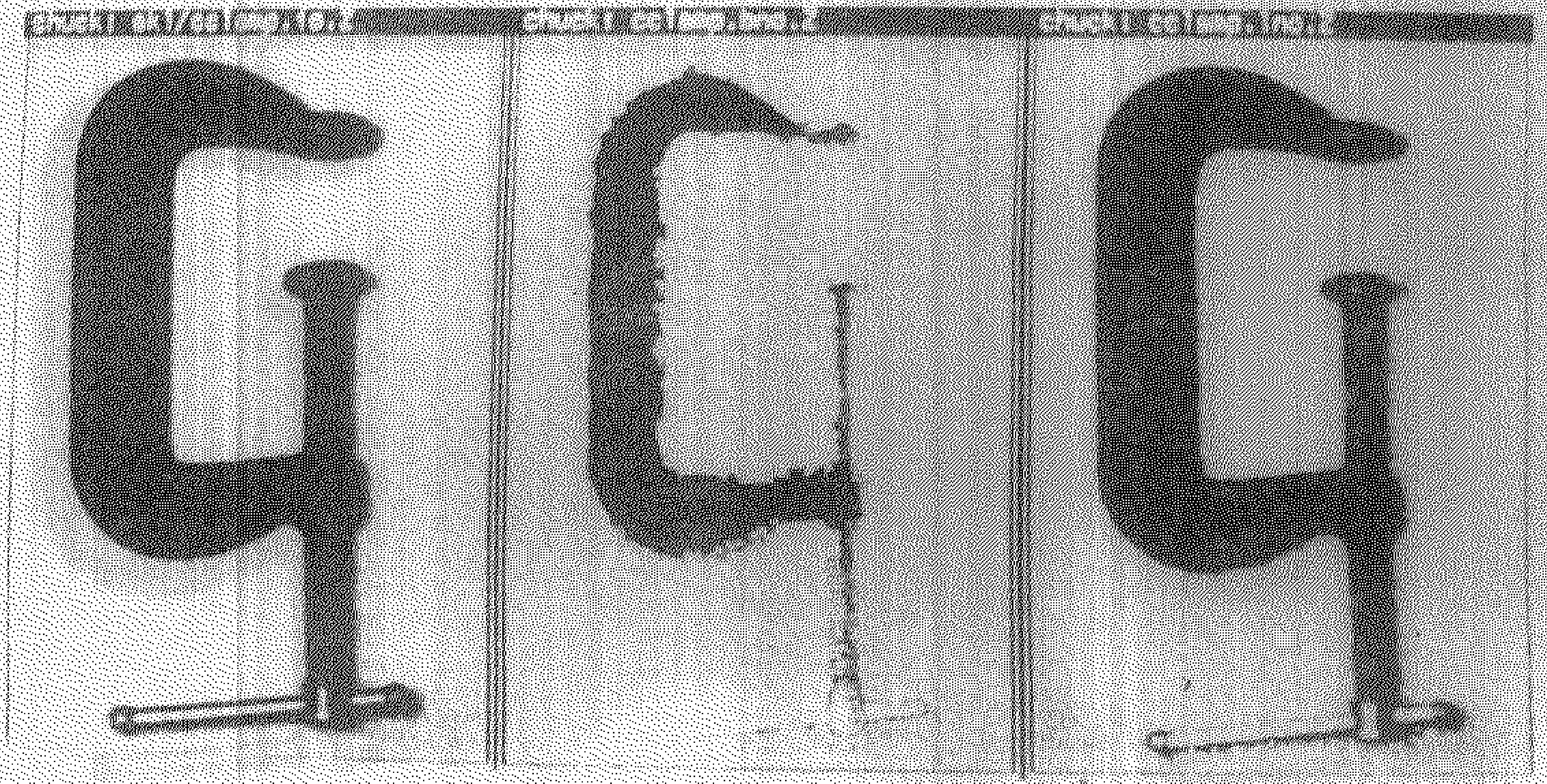


Figure 3.6: A noisy cclamp: (i) Input image, (ii) output of the connectionist model when: (a) dynamic normalization is used. (b) local normalization is used.

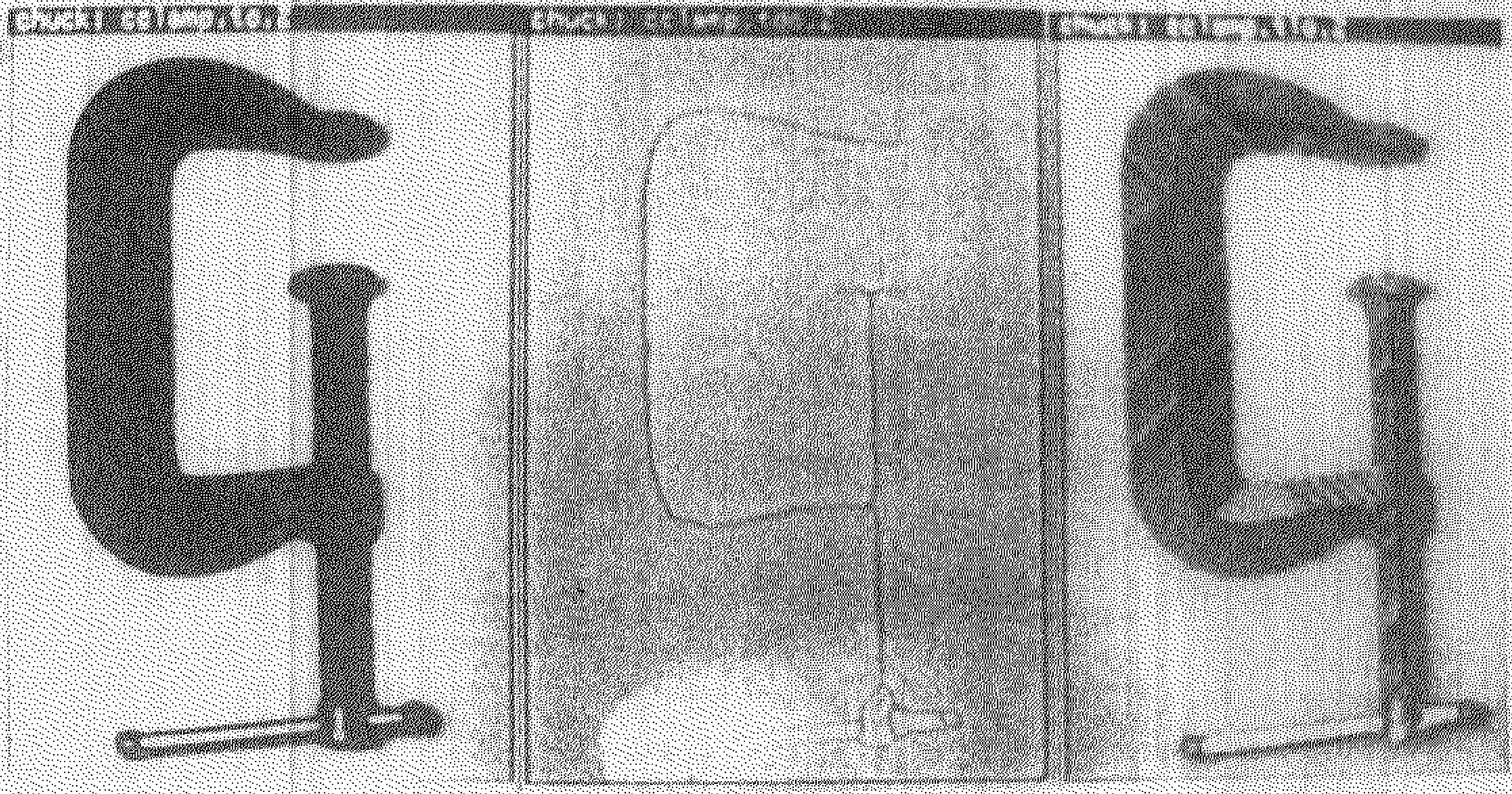


Figure 3.7: A noise free cclamp image obtained after preprocessing: (i) Input image, (ii) output of the connectionist model when : (a) dynamic normalization is used. (b) local normalization is used.

ization, it has been found that the dynamic normalization is the best method among global, local, and dynamic normalization. Also, if we compare a thresholding based method with the corresponding normalization based method, then the thresholding based method is more efficient than its counterpart; but, the quality of output image is better in case of normalization based method than its counterpart. In fact, the dynamic thresholding based method will give a result which will very much similar to the output obtained from the Modified PGTA, keeping the grayness of the image at its maximum possible limit. On the other hand, the dynamic normalization based method is very flexible and will be able to control the grayness of the output image

# Chapter 4

# CONCLUDING REMARKS

In this thesis, a thorough study of PGTA has been carried out. Some definitions like connection points, end points, single points, skeletal points, etc. have been generalized from two-tone domain to gray-tone domain. These generalizations are based on weighted definition of connectedness as given by Rosenfeld [2]. Based on these definitions, a new and unique value of threshold has been proposed which preserves connectivity in the local neighborhood of a candidate pixel. Also, termination condition of the PGTA has been modified, need of two subiterations in the PGTA has been eliminated and a modified version of PGTA has been presented.

In chapter 3, a new Connectionist Model for Image Skeletonization has been developed. There are many ways of applying the input image into the connectionist model. All these methods has been implemented and their relative advantages and disadvantages has been studied.

In classical algorithms e.g., Zhang-Wang algorithm, PGTA, modified PGTA etc., properties and conditions has been defined. On the other hand, an ANN captures these properties and conditions within the weights of the links.

In the first chapter, it has been pointed out that an ANN suitable supervised learning and does not demand orthogonality among the training data, and has ability to generalize the learned weights using binary patterns for thinning graytone images, can be tried for forming the Basic Building Block. Here, we have selected only one such network i.e. MLP, for this purpose. So, further research can be carried out in this area to investigate possibility of use of other ANNs as Basic Building Block.

♠

# Appendix A

# An algorithm for simulating the Modified PGTA

There are two image buffers, one holding the current graylevels of image pixels and the other holding the new gray level of these pixels obtained as a result of current iteration. Each iteration consists of two passes one for computing results(i.e. new graylevels) and the second pass for copying the result from new image buffer to current buffer. Note that the second pass starts only after the first pass has been finished and first pass of next iteration starts only after the second pass of the first iteration has been finished. We will distinguish the current and new buffers by cur and new extensions respectively. The step are as follow :

```
step 1. reset $flag$.
step 2. if $B(P_i) = 3$ for all i then there are only $2 \times 2$ image points. stop.
step 3. for all pixels in the current buffer do step 4.
step 4. if all the following conditions are satisfied

     1. $3 \leq B(P_1.cur) \leq 6$
     2. $A(P_1.cur) = 1$
     3. $P_2.cur < P_1.cur * P_4.cur < P_1.cur * P_8.cur < P_1.cur$ or $P_{11}.cur \geq P_1.cur$
     4. $P_2.cur < P_1.cur * P_4.cur < P_1.cur * P_6.cur < P_1.cur$ or $P_{15}.cur \geq P_1.cur$ then

               $$P_1.new = min(P_2.cur, P_3.cur, \ldots, P_9.cur) \qquad (A.1)$$

        set $flag$ to $TRUE$.
step 5. if $flag = TRUE$ at the end of step 3 then
               copy new buffer into current buffer and go to step 1.
        else   images in the current buffer as well as new buffer are the
               desired thinned images. Output any one of them and stop.
```

# Appendix B

# Simulation of the Connectionist Model

There are two image buffers, one holding the current graylevels of image pixels and the other holding the new gray level of these pixels obtained as a result of current iteration. Each iteration consists of two passes one for computing results(i.e. new graylevels) and the second pass for copying the result from new image buffer to current buffer. Note that the second pass starts only after the first pass has been finished and first pass of next iteration starts only after the second pass of the first iteration has been finished. We will distinguish the current and new buffers by cur and new extensions respectively. Also, the two buffers can be implemented as a two-dimensional array of a structure in 'C' or record in PASCAL, where the structure consists of two fields: one for holding the current graylevels of image pixels and other for holding the new graylevels of those image pixels obtained as output of the Connectionist Model. Let, *P*[*ROW*][*COL*] be the two-dimensional array of structure; where, *ROW* and *COL* are the number of rows and columns of the image to be thinned. Also, let a $(n \times n)$ window is used for learning. The algorithms are as given below :

## B.1 Global Normalization Method

The step are as follow :

```
step 1. Input the original image in the current buffer.
step 2. Normalize graylevels of the current buffer in [0,1] range using equation 3.10
step 3. {if graylevel of image background > graylevel of object then
        enter ($background =' W'$) else enter ($background =' B'$).}
        $read(background)$;
step 4. $If$ ($background =' W'$) $then$
        $for$ each pixel $P_{ij}$ in the current buffer $do$
        $P_{ij}.cur \leftarrow (1.0 - P_{ij}.cur)$;
step 5. Construct the Basic Building Block and assign the learned
        weights to the corresponding links.
```

```
step 6. $flag \leftarrow FALSE;$
step 7. $for(i \leftarrow \lfloor n/2 \rfloor;\ i < ROW - \lfloor n/2 \rfloor;\ i \leftarrow i+1)$
            $for(j \leftarrow \lfloor n/2 \rfloor;\ j < COL - \lfloor n/2 \rfloor;\ j \leftarrow j+1)$
              $begin$
                $OUTPUT \leftarrow forward(P_{ij});$
                $if((OUTPUT - P_{ij}.cur) > \epsilon) then$
                  $begin$
                    $P_{ij}.new \leftarrow OUTPUT;$
                    $flag \leftarrow TRUE;$
                  $end\{if\}$
              $end\{for\}.$
step 8. $if(flag = TRUE) then$
            $begin$
              $for(i \leftarrow 0;\ i < ROW;\ i \leftarrow i+1)$
                $for(j \leftarrow 0;\ j < COL;\ j \leftarrow j+1)$
                  $if$(Training data is generated by Zang-Suen thinning algorithm)$then$
                    $P_{ij}.cur \leftarrow P_{ji}.new;$
                  $else$
                    $P_{ij}.cur \leftarrow P_{ij}.new;$
              $goto$ step 5.
            $end.$
          $else$ $goto$ step 9.
step 9. $If\ (background = 'W')\ then$
          $for$ each pixel $P_{ij}$ in the current buffer $do$
          $P_{ij}.cur \leftarrow (1.0 - P_{ij}.cur);$
step 10. recompute the actual thinned image from the normalized thin image using equatio
         3·11
```

In case of global thresholding method, only the step 2 will change as we have to threshold the image using a global threshold instead of normalizing it. Rest will remain unchanged.

# Bibliography

[1] Kundu M.K., Chaudhuri B.B., and Majumdar D.D., "A parallel graytone thinning algorithm (PGTA)." pattern recog. letters 12,(1991) 491-496, North Holland.

[2] Rosenfeld A.,"On connectivity properties of grayscale pictures." Pattern recog. 16,(1983), 47-50.

[3] Rutowitz D.,"Pattern Recognition," J. Royal Statist. Soc. 129(1966), 504-530.

[4] Zhang, T.Y. and Suen, C.Y., "A fast parallel algorithm for thinning Digital patterns", CACM, 27, No.3, March 1984, pp.236-239.

[5] Lu, H.E. and Wang, P.S.P., "A comment on *Fast parallel thinning algorithm for digital patterns*," CACM, 29, No.3, March 1986, pp.239-242.

[6] Holt C.M., Stewart A., Clint M. and Perrott, R.H., "An improved parallel Thinning Algorithm," Commun.ACM, 30.2 (Feb. 1987). pp.156-160.

[7] Li B. and Suen C.Y., "A knowledge-based Thinning Algorithm," pattern recognition Vol.24, No.12, pp.1211-1221, 1991.

[8] Paler K. and Kittler J., "Graylevel edge thinning: A new method," pattern recognition letters 1, 409-416 (1983).

[9] Yu S.S., Tsai W.H., "A new thinning algorithm for grayscale images by the Relaxation Technique," pattern recog., Vol.23, No.10, pp.1067-1076, 1990.

[10] Dyer C.R. and Rosenfeld A., "Thinning algorithm for grayscale pictures," IEEE Trans. Pattern Anal. Mach. Intell. 1, 88-89 (september 1979).

[11] Zhang Y.Y. and Wang P.S.P., "A Modified Parallel Thinning Algorithm," CH2614-6/88/0000/1023$01.00 ©1988 IEEE, pp.1023-1025.

[12] Smith R.W., "Computer processing of line images: a survey," Pattern Recognition 20, 5-7, 1987.

[13] Arcelli C., Cordella L.P. and Levialdi S., "From local maxima to connected skeletons", IEEE Trans. Pattern Analysis Machine Intelligence, PAMI-3(1981), pp. 134-143.

[14] Pavlidis T., "A Thinning Algorithm for Discrete Binary Images", CGIP, 13 (1980), pp. 142-157.

[15] Rosenfeld A., "A characterization of Parallel Thinning Algorithms", Information and Control, 29(1975), pp.286-291.

[16] Stefanelli R. and Rosenfeld A., "Some Parallel Thinning Algorithms for Digital Pictures", JACM, 18(1971), pp.255-264.

[17] Pavlidis T.,"Algorithms for Graphics and Image Processing," Berlin- Heidelberg, Springer-Verlag, 1982.

[18] Gonzalez R.C. and Woods R.E., "Digital Image Processing," Addition- Wesley, 1992.

[19] Rosenfeld A. and Kak A., "Digital Picture Processing Vol. II," Academic Press, 2ed., 1982.

[20] Lippmann R.P., "An Introduction to Computing with Neural Nets," IEEE ASSP MAGAZINE APRIL 1987, pp.4-22.

[21] Pao Y.H., "Adaptive Pattern Recognition and Neural Networks," Reading Massachusetts, Addition-Wesley, 1989.

[22] Rumelhart D.E., Hinton G.E. and Williams R.J., 1986, "Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the microstructures of Cognition.* Vol. 1: *Foundations*, pp.318-362, MIT Press, Cambridge, MA.

[23] Chua L.O. and L. Yang, "Cellular Neural Network: Theory," IEEE Trans. on Circuits and Systems, Vol.35, No.10, pp.1257-1272,(Oct. 1988).

[24] Chua L.O. and L. Yang, "Cellular Neural Network: Applications," IEEE Trans. on Circuits and Systems, Vol.35, No.10, pp.1273-1290,(Oct. 1988).

[25] Matsumoto T., Chua L.O. and Yokohama T., "Image thinning with a Cellular Neural Network," IEEE Trans. on Circuits and Systems, Vol.37, No.5, pp.638-640,(May 1988).

[26] Wassermann ,"Neural Computing: Theory and Practice," New York: Van Nostrand Reinhold, 1990.

[27] Ooyen A.V. and Neinhuis B., "Improving the convergence of the back-propagation algorithm," Neural Networks, 5(3):465-471, 1992.