ISBN 0-315-59152-8

Charge Particle Method For Thinning Of Binary Images

Akilandeswari Arumugam

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfilment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

August 1990

ABSTRACT

Charge Particle Method For Thinning
Of Binary Images

Akilandeswari Arumugam

A new sequential thinning methodology, namely Charge Particle Method (CPM), is presented. According to this method, the dark pixels in a binary pattern are considered as particles having certain magnitudes of charge of the same type (positive or negative) and negligible mass. The movement of these particles towards the medial line location with respect to the boundary geometry is accomplished by the interaction of the electrostatic forces between the particles and the entire boundary. This movement of particles shrinks the pattern which is used in the CPM to generate the skeletons of binary patterns. The skeletons thus generated are found to possess all the required properties concerning the connectivity, topology and shape of the pattern.

The performance of the proposed CPM algorithm is compared with four other thinning algorithms [15,18,34] published recently. Overall, the CPM is shown to have superior characteristics with respect to processing speed, connectedness and reconstruction. The execution time of the CPM is found to be dependent on the width of the patterns to be skeletonised. For the sake of consistent evaluation of the thinning algorithms, a new bench mark input data set is introduced.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisors, Dr. T. Radhakrishnan and Dr. C.Y. Suen for their patience, valuable guidance and encouragement throughout the course of this work. Their constructive criticisms and help during the preparation of the thesis is well appreciated.

Also I would like to thank my colleagues and friends for their timely help and fruitful discussions.

The financial support provided by a research grant awarded by the Ministry of Education of Quebec for the research period is gratefully acknowledged.

Last but not the least, I would like to thank my husband A. Selvakumar for his untiring support and constant encouragement during the entire period of my studies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Optical Character Recognition (OCR)

Machine recognition of both handwritten and printed characters is becoming a prime requirement in many applications. Typical examples include automatic mail sorting, reading and verification of cheques and transaction sheets at banks, reading of price tags at super markets, capture and process directly from the handwritten records and in the development of interactive information management systems. It has been found that recognition and classification of handwritten or printed characters into discrete classes is highly dependent on the quality and variability of the data used and there is always a margin of error. The major goal in designing a character recognition system is to minimize such errors and achieve a low probability of misclassification and/or misidentification. With this goal in mind, this thesis looks into an important aspect of an OCR system, namely thinning, the output of which plays an important role in the later stages of recognition, classification and identification of characters.

A block diagram of an OCR system is given in the Figure 1.1. An OCR consists of the three main stages viz.,

Digitiser

Preprocessor

Feature Extraction/Detection

Classifier

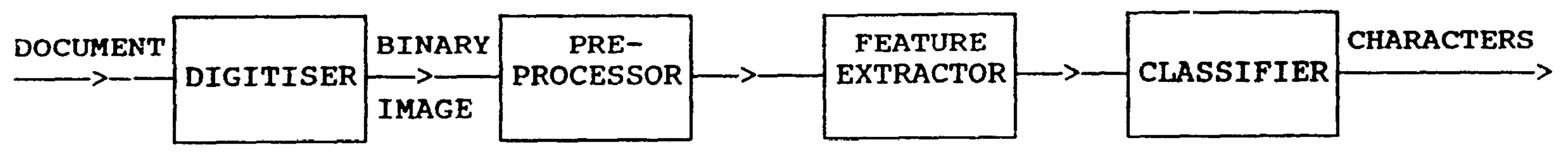


Figure 1.1: An OCR System

Digitiser: In order to process a given pattern by a computer, it is necessary to convert the original analog pattern into an array of numbers. This conversion is carried out at the digitiser stage by the processes of 'sampling' and 'quantization'.

Preprocessor: This stage performs transformations on the input character structure in order to increase the quality of the data. Thinning, smoothing and normalisation are the three primary functions performed during the preprocessing stage. Of these, thinning plays a very important role in the successful realisation of an OCR system. Thinning generates skeletons from the scanned or digitised input patterns. The redundant characteristics in the input image pattern are eliminated, retaining only essential characteristics and preserving the basic structure of the pattern. Thinning helps to reduce processing time and storage space in the subsequent stages of the OCR.

The smoothing step of the preprocessor stage removes the noise in the input image. Typically such noise is random in nature and is dependent on diverse factors such as the resolution of the scanner or digitizer, the quality of the pen used for writing and also on the style of the writer. The smoothing algorithm consists of essentially moving a 3 x 3 window across the binary image and comparing the state of the central element of this window with its eight neighbours to decide whether this state(pixel value) should be retained or changed. Smoothing helps to avoid spurious tails and distortions which would otherwise affect the performance of the classifier.

Finally, normalisation is carried out to remove the distortions introduced by variation in size, orientation and skew. One of the normalisation operations is centring the character in the image field. Characters may have varying sizes and hence normalisation of the character dimension in order to have uniformity in the size of the characters, thereby making the pattern matching process easier. Also the slopes introduced by the writer have to be eliminated to obtain a better classification. This is achieved by rotating the characters.

Feature Extraction: The character recognition process depends on a set of a priori determined features. The feature extraction is a stage in which the features are extracted from a thinned pattern. The thinned image is decomposed into its subpatterns or primitives. Primitives, for instance, could be horizontal, vertical lines, curved lines, number of loops, end-points or intersections [1].

Classifier: The preprocessed image is generally classified in two stages namely,

i) The structural classification method

and

ii) The relaxation matching.

Under structural classification methodology, classification of a character is based on the particular features present in its skeleton as well as the relative interconnection among those features. Based on the configuration of the primitives and extracted features, the patterns with simpler structures are classified with the use of decision trees [2].

In relaxation matching the patterns that are not recognised by the structural classifier are matched against a set of reference patterns using a relaxation process. The likelihood of the match is decided based on the proximity of the features. A number of relaxation matching algorithms exist and are reviewed in [3].

1.2. Thinning

The subject matter of the thesis, namely Thinning, transforms the input binary pattern to thin line drawings known as 'skeletons' which retain the connectivity and the original shape of the pattern. The advantages of deriving the skeletons for patterns are:

i) to reduce data handling (both storage and processing time)

and

ii) to reduce transmission requirements [4,5].

In most of the applications the width of the pattern is not of importance as it contains very little useful information. Hence the skeletons can be used to represent the patterns, thereby reducing the storage space and time for later processing. The skeletons are in general used for the structural shape analysis needed in the recognition of line drawings [6] and alphanumeric characters. Other typical applications are include:

1. Biomedical systems (eg. chromosome analysis) [7],
2. Fingerprint classification [8],
3. Logic and electrical circuit interpretation [9],
4. Soil Crack Analysis [10].

Extensive research has been done in the area of thinning for the past three decades and numerous thinning algorithms exist. We first review certain basic definitions.

1.2.1. Definitions

Pixels: In picture processing, the binary digitised pattern is normally represented by a matrix where each element is assigned a value of either '1' (dark-point) or '0' (white-point). These points are also called as **pixels**. The actual pattern consists of all the dark-pixels in the matrix.

Thinning: It is the transformation of a digitized pattern to a unit width line, preserving the shape of the original pattern and lying along the medial axis known as 'skeletons'.

Medial-Axis: of a pattern is the locus of points within the pattern such that for each point P on the locus, there exists at the least two points on the pattern boundary that are equidistant from and are closest to P. The medial axis is thus another definition used to refer to the 'skeleton' of a pattern.

Medial-lines and Core-lines: are other terms that are commonly used to refer to skeletons.

8-neighbours: Figure 1.2 defines the 8-neighbours of an arbitrary pixel P. These are the pixels P_1 to P_8, that are adjacent to the pixel P.

P_8	P_1	P_2
P_7	P	P_3
P_6	P_5	P_4

Figure 1.2: Neighbours of a Dark Pixel

4-neighbours: The pixels P_1, P_3, P_5 and P_7 are known as the direct 4-neighbours or D-neighbours of pixel P whereas the pixels P_2, P_4, P_6 and P_8 are known as the indirect 4-neighbours or I-neighbours of pixel P.

Edge-Point: A dark-point having at the least one of its D-neighbours a white-point is termed as an edge-point. The edge-points are further classified as left, right, top and bottom edge-point depending on which of the 4-direct neighbours P_7, P_3, P_1 or P_5 respectively are white-points. Edge-points are also sometimes referred to as Boundary or Contour Points/Pixels.

Border or Contour: The set of edge-points of any pattern form the contour or the border of the pattern.

End-Point: End point is a dark point having utmost one dark 8-neighbour. An end-point test, during thinning ensures that the end-points are retained and are not eroded in the process of thinning.

Break-Point: This is a dark-point the deletion of which would result in breaking the connectedness of the pattern.

8 or 4-connectedness: Digitised picture is 8 or 4-connected if for any two points p and q there exists a sequence of points $p = p_0, p_1, p_2, \ldots, p_{n-1}, p_n = q$ such that p_i is a 8 or 4 neighbour of p_{i-1} for $1 < i \le n$.

Crossing Number: This is the first measure of connectivity introduced by Rutovitz [11]. For any pixel P, the crossing number is defined as

$$\lambda = \sum_{j=1}^{j=8} | P_j - P_{j+1} | \qquad (2.4a)$$

This is used to count the number of black-to-white transitions in the 8-neighbourhood of pixel P.

Connectivity Number: The connectivity number CN introduced by Yokoi in 1975 [12], is defined for any dark pixel P as

$$CN = \sum_{j \epsilon S} (P_j - \prod_{k=j}^{k=j+2} P_k) \text{ for 4-connectedness}$$

$$CN = \sum_{j \epsilon S} (\bar{P}_j - \prod_{k=j}^{k=j+2} \bar{P}_k) \text{ for 8-connectedness}$$

where

P_j's correspond to an 8-neighbour of pixel P

$S = \{1,3,5,7\}$; if $k > 8$ then $k = k - 8$.

$\bar{P}_j = 1 - P_j$

The dark pixel P is classified as an isolated, edge, connecting, branching or crossing-point depending on the value of CN computed being 0, 1, 2, 3 or 4 respectively. If the value of CN = 1, then the dark pixel is an edge point and can be deleted.

Distance Transformation: It is the transformation of the binary pattern by which each dark-point in the input binary pattern is assigned a value equal to the length of the shortest path from that dark-point to the nearest white-point. If the path consists exclusively of horizontal and vertical steps of unit length, then the distance transform is known as city block or 4-distance transform. On the other hand, if the distance is computed based on the 8-neighbours, the transform is known as chess board transform.

Maximal Blocks: Any given binary pattern can be considered to be made up of a series of non-overlapping blocks of varying sizes. If all such blocks are chosen such that:

a) they are of maximum possible size,

b) all of them touch the boundary at least once, and

c) they do not cross the boundary,

then they are referred to as 'Maximal Blocks'. Depending on whether the maximal blocks are derived from circles or squares, they are respectively termed as maximal circles or maximal squares.

1.2.2. Classification of Thinning Algorithms

Thinning algorithms are classified primarily based on the method by which the skeletons are derived. Further sub-classification is done based on the implementation or processing method.

Under the primary classification, there are two subclasses viz., the medial-axis transforms (or skeletonising algorithms) and the thinning algorithms.

The medial-axis transform was first proposed by Blum [13] to describe a figure or shape of an object. This method generates the skeleton by joining the centres of the maximal blocks defined across the pattern. It does not involve the process of iteratively deleting the contour points. The skeletons thus derived are isotropic, sensitive to boundary noise and have the ability to reconstruct the original pattern from the skeletons.

The thinning algorithms consist of removing contour points at each iteration, thereby shrinking the original pattern until a unit width line is formed. The contour points are normally tested against a mask of given size (usually a 3x3 window) such that the removal of the contour points does not affect the connectivity of the pattern. The scanning can be done either by conventional method (raster scanning) or by contour tracing. In contour tracing method, only the successive border points of the pattern are processed; whereas in the raster scan method the entire pattern is scanned during every iteration.

Each of the above categories can be further sub-classified as Sequential algorithms or Parallel algorithms.

Sequential Thinning Algorithms: In this case the edge-points are processed one at a time and the result of processing a dark point in the k^{th} iteration depends on the set of points for which the result of the k^{th} iteration is known.

Parallel Thinning Algorithms: Parallelism is attained by breaking a given iteration into several distinct subiterations (process partitioning) or by segmenting the input pattern (data partitioning). In this method all the edge-points are processed simultaneously. The result of processing a dark-point at the k^{th} iteration depends on the values assigned to the point and its 8-neighbours at the $(k - 1)^{th}$ iteration.

1.2.3. Literature Survey

Ever since the generation of skeletons for discrete images was proposed by Rosenfeld [14], enormous work has been done in this area. In this thesis we review only the well known sequential thinning algorithms of the recent past (since 1984). Naccache [15] has reviewed and compared the most commonly referenced thinning algorithms that have been proposed up to 1984.

In general all the thinning algorithms are based on local operations in the 8-neighbourhood of a pixel (3x3 window) and they differ in the method used for ensuring the connectivity in the skeleton and preserving the end-points of the pattern. Naccache et al have proposed a new thinning algorithm namely, the Safe Point Thinning algorithm (SPTA) [15] for binary patterns. It has been concluded in [15] that SPTA algorithm is found to be generating good quality skeletons and the fastest among the algorithms used for comparison with the ability to reconstruct the original pattern from the skeleton.

In 1985 Arcelli et al [16] proposed a method which first derives a set of points symmetrically placed with respect to the contour parts of a figure and are further thinned by using the topology preserving operation. Finally a pruning stage is included to eliminate the noisy spurs. From computational point of view this method is independent of the size of the pattern and requires the same fixed number of passes.

A two phase method of thinning resulting in good quality skeletons and having good resistance to noise was derived by Chu and Suen in 1986 [17]. The first phase of this algorithm consists of alternate smoothing and stripping of contour points and the second an adjusting phase to obtain the medial skeletons. This method is time consuming because of the smoothing process involved in each iteration.

Suzuki et al came up with a width independent thinning algorithm [18] using distance transformation to locate the medial lines. In this methodology contour tracing is used to process the pattern and hence a priori knowledge of the contours in the pattern is required. In addition a post-processing stage is needed to derive unit width medial lines.

Dill and Levine [19] provided an algorithm to compute multiple resolution skeletons of noisy images which is used for the study of pseudopods in leucocyte locomotion. This algorithm generates skeletons at different resolutions of the picture from which a filtered version is obtained. This algorithm is found to be as fast as the algorithm of Arcelli [16].

A sequential thinning algorithm based on contour tracing was proposed and implemented by Govindan [20]. This algorithm incorporates shape adaptivity to eliminate the asymmetry in the skeletons of even thickness patterns with T-corners, and to preserve the exact shape at right-angle and acute-angle corners of patterns. Such shape preservation reduces the complexity of some of the pattern recognition problems.

Kwok [21] proposed a thinning algorithm based on contour generation which is much faster than the [15], [22] and [23]. The chain codes are generated for every closed contour and safe-points are detected for every contour by using the values assigned to the pixels as they are visited.

A connectivity preserving shrinking algorithm based on Hilditch [7] was proposed by Ye et al [24] for the visual inspection of printed circuit boards. The error indications due to the sharp corners, digitisation noise and small protrusions can be eliminated by using extra connectivity preserving shrinking operations.

Xia implemented for the first time, Blum's idea of fire propagation to the thinning of discrete images [25]. The wave propagation is simulated by means of contour traversals and is relatively faster as it does not involve much computations because of the elimination of neighbourhood testing operations.

Kwok [26] proposed a connectivity preserving shrinking algorithm based on contour generation. The algorithm uses the distance transformed pattern and removes successive contour as new contours are being generated. The distance value is used to for the isotropic erosion.

Wang [27] proposed a thinning algorithm based on contour tracing which is faster than those of Naccache [15], Lu [28] and Zhang [21]. This algorithm is flexible in that it can be implemented either in sequential or in parallel mode and structure preserving.

Xia explains in detail the implementation of the thinning methodology based on Blum's model for discrete images [29]. It is found that the algorithm does not result in unit width lines at the curved branchings and intersections. Further since the algorithm is based on contour tracing it is not suited for the processing of composite characters or patterns.

Table 1.1 shows the characteristics of the various algorithms that are reviewed in this thesis. The table lists the method employed for processing, the type of connectivity in the skeletons generated and reported testing as to whether an extensive, average or minimum amount of data set is used for testing the algorithms.

1.3. Scope of the Thesis

In this thesis a new thinning algorithm namely, the Charge Particle Methodology (CPM) is proposed. The detailed explanation of the algorithm which is based on the force of interaction between charged particles, its implementation and analysis are given in Chapter 2. The reconstruction ability of the algorithm is also given in this chapter.

In Chapter 3, the selection criteria used for choosing the four thinning algorithms for the comparative study with the proposed algorithm are described. Then the four algorithms and their implementation are detailed.

Table 1.1: Salient Features of the Reviewed Algorithms

Algorithm	Processing Method	Connectivity	Endpoint Erosion	Reconstruct-ability	Reported Testing
Naccache 84 [15]	RS	perfect 8	Minimum	Yes	Extensive
Arcelli 85 [16]	CT	perfect 8	Unknown	Yes	Unknown
Chu 86 [17]	CT	perfect 8	Minimum	No	Extensive
Suzuki 86 [18]	CT	perfect 8	Minimum	Yes	Average
Dill 87 [19]	CT	perfect 8	Unknown	No	Minimum
Govindan 87 [20]	CT	imperfect 4	Unknown	No	Unknown
Kwok 88 [21]	CG	imperfect 4	Minimum	No	Unknown
Ye 88 [24]	RS	imperfect 8	Minimum	No	Unknown
Xia 88 [25]	CT	imperfect 8	Minimum	Yes	Unknown
Kwok 89 [26]	CG	imperfect 8	Minimum	No	Minimum
Wang 89 [27]	CT	imperfect 8	Minimum	No	Minimum
Xia 89 [29]	CT	imperfect 8	Minimum	Yes	Minimum

RS: Raster Scanning CT: Contour Tracing CG: Contour Generation

In Chapter 4, the Data set selection criteria and the basis on which the thinning algorithms are compared are discussed initially. Then the essence of the comparative study based on the experimental results is presented.

Finally in Chapter 5, the concluding remarks on the work done in this thesis and the scope for future work are given.

2. PROPOSED ALGORITHM

The aim of any thinning scheme is to obtain the connected skeleton of unit width along the medial axis of the given binary pattern. In addition, in order to obtain good quality skeletons, a thinning scheme should have the following characteristics:

i) The resulting skeletons should inherit all the topological features of the original pattern,

ii) No undue erosion of the ends of the skeletons,

iii) Be immune to the salt and pepper noise,

iv) Have good reconstructability,

v) Be robust enough to produce identical skeletons irrespective of the orientation of the patterns, and

vi) Have a fast throughput.

The new sequential thinning methodology presented in this thesis, namely Charge Particle Method (CPM), meets the above requirements. In this chapter the formulation, implementation, as well as the performance of the CPM are detailed.

2.1. Basis of the Charge Particle Method (CPM)

In the Charge Particle Method, the dark pixels in a binary pattern are considered as particles having certain magnitudes of charge of the same type (positive or negative charge) and negligible mass. The movement of these particles towards the medial line location with respect to the boundary geometry is accomplished by the interaction of the electrostatic forces between individual particles and the entire boundary. According to Coulomb's law, the electrostatic force of repulsion between any two charged particles of the same polarity is proportional to the product of the charge magnitudes and is inversely proportional to the square of the distance between the two particles. The resultant force acting on a particle due to several neighbours is a vector sum of the forces due to all the surrounding particles. Physically the particles are free to move anywhere on the continuous plane. However, by suitably charging the boundary of the binary pattern, the charge particles representing the dark pixels can be forced to move away from the boundary. This results in a shrinking of the pattern, which is used in the CPM to generate the skeletons of binary patterns.

For example, consider the binary pattern shown in Figure 2.1. The dark pixels in the pattern are considered to represent charged particles of the same polarity, (say positive) and having the same charge magnitude. The boundary is maintained at a higher potential. Due to the electrostatic force of repulsion between the particles and the boundary, the particles within the boundary tend to move away from the boundary, thereby resulting in a thinning of the pattern.

The pixels marked as '1' along the boundary of the pattern are pushed towards the centre as indicated by the arrows since the

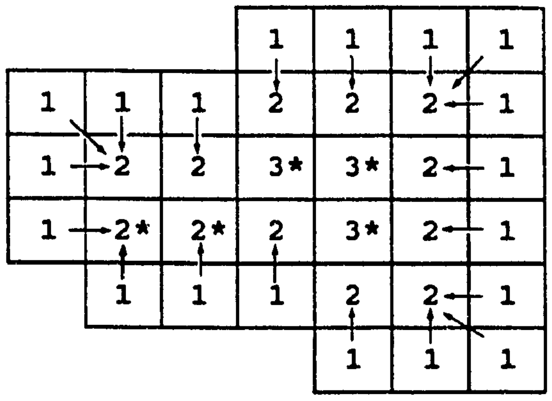


Figure 2.1: CPM Concept

force due to the boundary is greater than that of the particles inside. These particles now take up the positions marked as '2'. The process continues till the charged particles attain a state of equilibrium at which time the net force acting on them is zero. The pixels marked by '*' are the final retained pixels representing the skeleton.

2.2. Analogy to Grassfire Concept

The CPM can be considered as somewhat analogous to the well known Grassfire concept proposed by Blum [13], in that both are based on a phenomenon which occurs over a continuous plane, suitably adopted to the discrete process of thinning. As per the Grassfire concept, the region comprising the dark points of a binary pattern is considered to be analogous to a grassland topology.

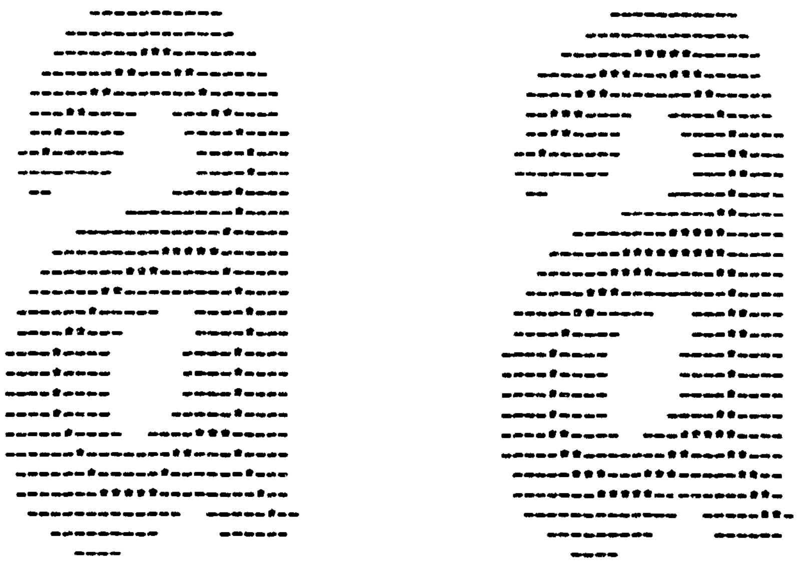
A fire is started along the perimeter or the boundary. The fire spreads uniformly from the boundary towards the interior of the grassland. This spreading of fire burns the grass along the boundaries in layers. The "propagation" of the fire front models the motion of the dark pixels. As the fire front moves across the pattern, the dark pixels on its path are eliminated. Finally when the fire fronts from different directions meet, the fire is considered to be quenched. The locus of the "quench points" corresponds to the required skeleton. Figure 2.2 shows the initial patch of grassland. Once the fire has started along the perimeter, the pixels marked '1' in Figure 2.2 will burn first, followed by those marked as '2' and so on. Finally the pixels marked by '*' form the quench points where the two fire fronts meet. These final quench points form the skeleton. Xia [29] has adopted this concept to thinning.

			1	1	1	1
1	1	1	2	2	2	1
1	2*	2*	3*	3*	2	1
1	2*	2*	2	3*	2	1
	1	1	1	2	2	1
				1	1	1

Figure 2.2: Grassfire Concept

In the case of CPM, the movement of the particles starts from the boundary and proceeds towards the interior of the pattern, which is analogous to the inward movement of the fire front in the Grassfire concept. The quench points in Grassfire concept at which the fire extinguishes correspond to the skeletal points; whereas in CPM this corresponds to the equilibrium position reached by the charged particles. A charge particle reaches an equilibrium position when the resulting force acting on it is equal to zero. In the case of Grassfire concept, the deletion of dark pixels along the border is due to burning whereas in CPM, their deletion is due to the resultant electrostatic force acting on them being not equal to zero.

Comparing the skeletons generated by the CPM and those by Blum's Grassfire model [29], it can be seen that the skeletons generated by the CPM are of much better quality. For example, for patterns having curved intersections the skeletons given by the Grassfire model (Figure 2.3b) have more multiple pixels than the skeletons given by CPM (Figure 2.3a).



a) CPM b) Grassfire Model [29]

Figure 2.3: Skeletons Generated by the CPM and the Grassfire Model

2.3. Description of the CPM Algorithm

In this section, the adaptation of the CPM concept to thinning, is detailed.

Theoretically, the resultant force acting on a particle, is due to the cumulative effect of all the other particles in the plane. However, in our adaptation of the concept for thinning, only the forces due to the immediate eight neighbours of a particle are considered to be significant. This assumption is valid since the force between particles is inversely proportional to the square of the distance between them and hence the effect of forces due to the pixels outside the first eight neighbourhood are negligible.

Following the above assumption the particle motion is also restricted to be in one of these eight directions. The boundary which would define the shape of the object under analysis is conceived to be a series of suitably charged particles.

Force Calculation

With the above assumptions it is necessary <u>to determine only the effective force on a particle due to the eight immediate neighbours.</u> The force of repulsion between any two particles is given by Coulomb's law as:

$$F = \frac{Q_1 \; Q_2}{4 \; \pi \; \epsilon \; R^2} = k \; \frac{Q_1 \; Q_2}{R^2} \qquad (2.1)$$

where

Q_1 = magnitude of charge of particle 1

Q_2 = magnitude of charge of particle 2

R = separation distance between particles

ϵ = permittivity constant, a property of the surrounding medium

k = proportionality constant = $1/(4 \pi \epsilon)$

Consider the particle P_i surrounded by the particles $P_1, P_2, ..P_8$ as shown in Figure 2.4. To establish the condition for the movement of the particle P_i, the resultant force on particle P_i from all other particles P_j (j = 1 to 8) is required. The force which particle P_1 exerts on the particle P_i lies along the unit vector drawn from particle P_i to particle P_1. When many particles are present, net force is the vector sum of all the forces.

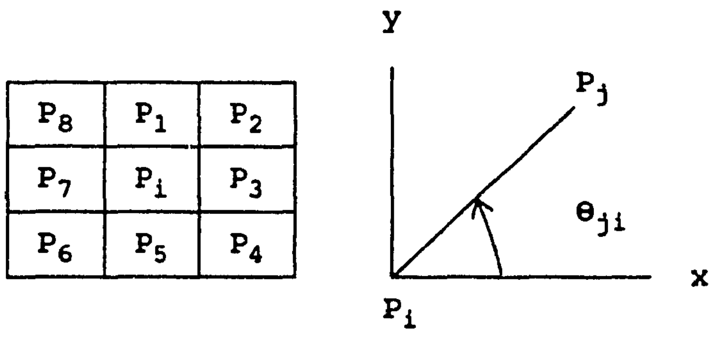


Figure 2.4: Eight Neighbours of a Dark Pixel

For each particle P_j a separation distance D_{ji} to the particle P_i can be found as:

$$D_{ji} = \sqrt{(P_{xi} - P_{xj})^2 + (P_{yi} - P_{yj})^2} \qquad (2.2)$$

where

P_{xi}, P_{yi} = x and y coordinates of point P_i

P_{xj}, P_{yj} = x and y coordinates of points P_j for j varying from 1 to 8.

Using the equation (2.1), the force F_{ji} which P_j exerts on P_i can be written as

$$F_{ji} = k \frac{Q_j \, Q_i}{D_{ji}^{\,2}} \qquad (2.3)$$

Lastly, the net force on particle P_i, due to the immediate eight neighbours can be expressed in x and y components as

$$F_{xi} = \sum_{j=1}^{j=8} F_{ji} \, . \, \mathrm{Cos}\ \theta_{ji} \qquad (2.4a)$$

$$F_{yi} = \sum_{j=1}^{j=8} F_{ji} \, . \, \mathrm{Sin}\ \theta_{ji} \qquad (2.4b)$$

where

F_{xi} = x component of the resultant force acting on P_i due to P_j.

F_{yi} = y component of the resultant force acting on P_i due to P_j.

θ_{ji} = orientation angle of particle P_j with respect to particle P_i.

From Figure 2.4,

$$\cos \theta_{ji} = (P_{xi} - P_{xj})/D_{ji} \qquad (2.5a)$$

$$\sin \theta_{ji} = (P_{yi} - P_{yj})/D_{ji} \qquad (2.5b)$$

Substituting the values for F_{ji}, Sin θ_{ji} and Cos θ_{ji} in equations 2.4a and 2.4b we get the following equations for F_{xi} and F_{yi}.

$$F_{xi} = k \sum_{j=1}^{j=8} Q_j Q_i (P_{xi} - P_{xj})/{D_{ji}}^3 \qquad (2.6a)$$

$$F_{yi} = k \sum_{j=1}^{j=8} Q_j Q_i (P_{yi} - P_{yj})/{D_{ji}}^3 \qquad (2.6b)$$

From the x and y components of the force, the motion of the particle P_i can be determined.

CPM Algorithm

Thinning of a binary pattern in this method is accomplished by a number of passes. In each pass, a set of edge-points are deleted from the input pattern. The algorithm consists of the steps 1 to 4 stated below.

Step 1: Scan the input binary picture from top to bottom and left to right. When a dark pixel is encountered, check if it is an edge-point or not. If it is an edge-point then go to step 2, else continue the scanning till the lower right end of the pattern is reached.

Step 2: Each edge-point is tested against the following conditions for their removal:

(a) The number of dark pixels in the first eight neighbourhood is greater than one.

(b) The eight connectivity number CN [12] is one.

(c) The net force as calculated from equation (2.6) is not equal to zero.

The edge-points satisfying the above conditions are flagged for their deletion.

Step 3: Repeat steps 1 and 2 till there is no deletable pixel in any pass.

Step 4: This is a postprocessing stage where the redundant pixels in the binary pattern are deleted. This is achieved by scanning the result of step 3 and deleting each of the dark pixels satisfying any of the configurations [30] given in Figure 2.5.

The final retained pixels correspond to the skeleton. In the above formulation, it is found that after each pass the boundary is reduced by unit depth. Hence the maximum number of passes required would be equal to $W_m/2$, where W_m is the maximum width of the object.

2.4. Implementation of the Algorithm

The CPM algorithm has been implemented and tested on a Cyber

CDC830D system using Pascal language. The input binary pattern is stored in a two dimensional matrix of integers. In order to distinguish the boundary from the pattern to be thinned, the dark pixels are assigned a positive charge value of unity and the white pixels are assigned an arbitrary negative charge value of - MAX, where MAX corresponds to a high integer value.

The whole process of thinning is accomplished on this single matrix. Thinning of the object is done by successively removing the boundary pixels in distinct successive iterations or pass. The scanning can be done either in a rowwise or a columnwise fashion. Thus choice of scanning direction does not affect the resulting skeleton. During each iteration only the dark points are processed from which the edge-points are detected. The edge-points are then flagged for their removal if the three conditions listed below are satisfied. The flagged edge-points are assigned a value of (i - MAX), where 'i' corresponds to the iteration number.

(a) Test for excessive erosion of End points

In order to avoid the excessive erosion of end points, each edge-point is tested against the immediate eight neighbours to determine if it is an end point or not. This is done by counting the number of dark pixels in the eight neighbourhood and if this number is greater than one then the pixel under consideration is not an end point.

(b) Test for Connectivity/Breakpoint

Another important criterion for a skeleton is to preserve the connectivity of the original object. This condition is tested by determining the eight connectivity number CN [12] given by

$$CN = \sum_{j \in S} (\bar{P}_j - \prod_{k=j}^{k=j+2} \bar{P}_k) \qquad (2.7)$$

where

$S = \{1,3,5,7\}$; if $k > 8$ then $k = k - 8$

$\bar{P}_j = 1$ if $P_j < 0$

$\quad 0$ if $P_j > 0$

If the value of CN = 1, then the pixel can be deleted.

(c) Force Calculation

The net force due to the eight neighbours is determined using the equation (2.6). If the resultant force is not equal to zero then the pixel can be deleted. The direction of motion of the particle is determined from the relative magnitudes of x and y components of the net force.

The edge-points satisfying the above conditions (a) through (c) are flagged but not removed, as their removal in the same pass would affect the force calculation for the rest of the dark pixels in the image. At the end of each pass we have a set of undeleted dark-points with a value of unity, flagged points with a negative value (i - MAX) and the white points of value (- MAX). During the next pass the points with values less than or equal to (i - MAX) are considered as white points. The flagging technique [15] used thus eliminates the necessity for an additional scan through the binary pattern to delete the flagged points. In addition it helps in the reconstruction of the original pattern from the skeleton.

In each pass one layer of boundary pixels are removed. The thinning process stops when no further deletion occurs. To delete the redundant pixels, the resulting thinned pattern is then scanned and every dark pixel is tested against the Boolean expressions given in equations 2.8 to 2.11.

$$P1*P3*\overline{P6} = 1 \tag{2.8}$$

$$P3*P5*\overline{P8} = 1 \tag{2.9}$$

$$P5*P7*\overline{P2} = 1 \tag{2.10}$$

$$P7*P1*\overline{P4} = 1 \tag{2.11}$$

These expressions correspond to the configurations given in Figure 2.5. If any of these expressions has a true value, then the dark point under consideration is deleted.

2.5. Analysis and Performance of the Algorithm

It can be seen from section 2.3 that the steps 1 and 2 can be performed in one raster scan. In every pass every pixel has to be examined for dark point and hence the number of operations is proportional to the area of the input pattern. The number of passes is dependent on the thickness of the object. Thus the time complexity of the algorithm can be written as O(n*m*w) where n and m are the raster scan area measurements and w is half the largest thickness of the object.

	*	x
*	P	
x		

x	*	
	P	*
		x

		x
	P	*
x	*	

x		
*	P	
	*	x

* : Dark Pixel   x : Don't Care

Figure 2.5: Configurations for Redundant Pixel Removal

The skeletons generated by this algorithm have been found to be able to preserve well the connectivity with no excessive erosion of end points. In addition the topology of the object is also maintained by the skeletons. For evaluation purposes, a set of binary patterns from various papers published in the literature have been chosen. We have also added some characters from Chinese and Tamil languages which are typically complex in shape. Table 2.1 shows the test patterns and the processing time required for each of the patterns. The set of binary patterns and the generated skeletons are given in the Appendix. Pixels retained as skeleton are denoted by '*' and pixels deleted from the original patterns are denoted by '-'.

It can be seen from the results that the skeletons produced are lines of single width and are also smooth and the processing

Table 2.1: CPU Times for Thinning of Characters in the Data Set

CODE	CHARACTER	CPU TIME
C1	A	0.546
C2	B	0.218
C3	E	0.692
C4	G	0.217
C5	H	0.446
C6	O	0.884
C7	Q	1.091
C8	R	1.081
C9	S	0.198
C10	Y	0.754
C11	X	1.037
C12	e	0.781
C13	g	0.971
C14	n	1.037
C15	u	0.915
C16	க	1.058
C17	ச	0.662
C18	ள	1.272
C19	ழ	0.740
C20	ஐ	1.702
C21	ஞ	1.270
C22	ஒ	1.110
C23	ட	0.194
C24	2	0.929
C25	4	0.534
C26	5	0.551
C27	6	0.240
C28	8	0.234
C29	大	2.138
C30	南	1.371
C31	的	1.390
C32	孝	0.882

time required ranges from about 0.2 to 2.15 CPU seconds for matrix sizes of 10 x 20 to 60 x 60. The processing time is found to decrease considerably with the patterns produced by a low resolution scanner. Since the algorithm is based on the raster scan of the binary image, a priori knowledge of the contours is not needed as in the case of contour tracing algorithms. This makes the algorithm desirable for characters with islands and for composite characters containing multiple components or radicals like many of the Chinese and Tamil language characters.

Figure 2.6 shows the skeletons produced for two pixel wide horizontal, vertical, diagonal, intersecting lines and squares of dimensions 2x2 and 3x3. These are considered to be difficult test patterns for thinning algorithms [31] and are useful for identifying defects in the connectivity and medial curve preservation. It is found that the algorithm produces no redundant pixels and that the images are reduced to single line thickness for these typical test patterns.

Consider the skeletons generated for the characters 'A' and 'H' respectively embedded with salt and pepper noise, shown in Figure 2.7a and 2.8a. It is found that a spurious tail is generated for the 'H' and a loop is formed in the 'A'. The algorithm treats the presence of any salt noise as an island and so a connection is provided to it. This results in the loop formation as denoted by a circle in Figure 2.7b. By using

```
               --      --                    --
               -*       --                   -*
               -*        **                  -*
               -*         -*                 -*
-----------    -*          -*           ------*-----
***********    -*           -*          ************
               -*            -*              -*
               -*             -*             -*
               -*              -*            -*
               -*               -*           -*
```

```
                               *    *    *
                                *   *   *
                                 *  *  *
                                  * * *
          ---      -----           ***
--        -*-      -***-      *************
**        ---      -----           ***
                                  * * *
                                 *  *  *
                                *   *   *
                               *    *    *
```

* : Skeletal points   - : Original Dark Points

Figure 2.6: Typical Test Patterns with the Skeletons Superimposed on the Original

the smoothing algorithm introduced by Chu and Suen [17] to remove the salt and pepper noise, it is found that the spurious tails can be removed. Thus filtering results in good skeletons (Figures 2.7c and 2.8c).

The skeletons produced by CPM follow the variations along the boundary very closely. In Figure 2.9 the filling of the salt noise in character '2' does not eliminate the spurious tail as seen in Figure 2.9c. The loop in the skeleton of Figure 2.9b due to the salt noise is eliminated but the tail still exists because of the protrusion along the boundary. In situations where the input patterns contain thick patches, such as the character 'Y' and 'the moving man' shown in the Figure 2.10, this algorithm introduces end point erosion.

The performance of the algorithm is tested against four other algorithms reported in the literature and has been found to be the second fastest, next only to the Distance Transformation (DT) method. With thin characters the CPM is found to be faster than the DT method. This makes the algorithm more desirable than other methods for processing of binary patterns generated by a low resolution scanner. Details of the comparison between various algorithm are given in Chapter 4. The CPM has the ability of reconstructing the pattern with the least amount of mismatch with the original pattern. The reconstructability of the algorithm is explained in detail in the following section.

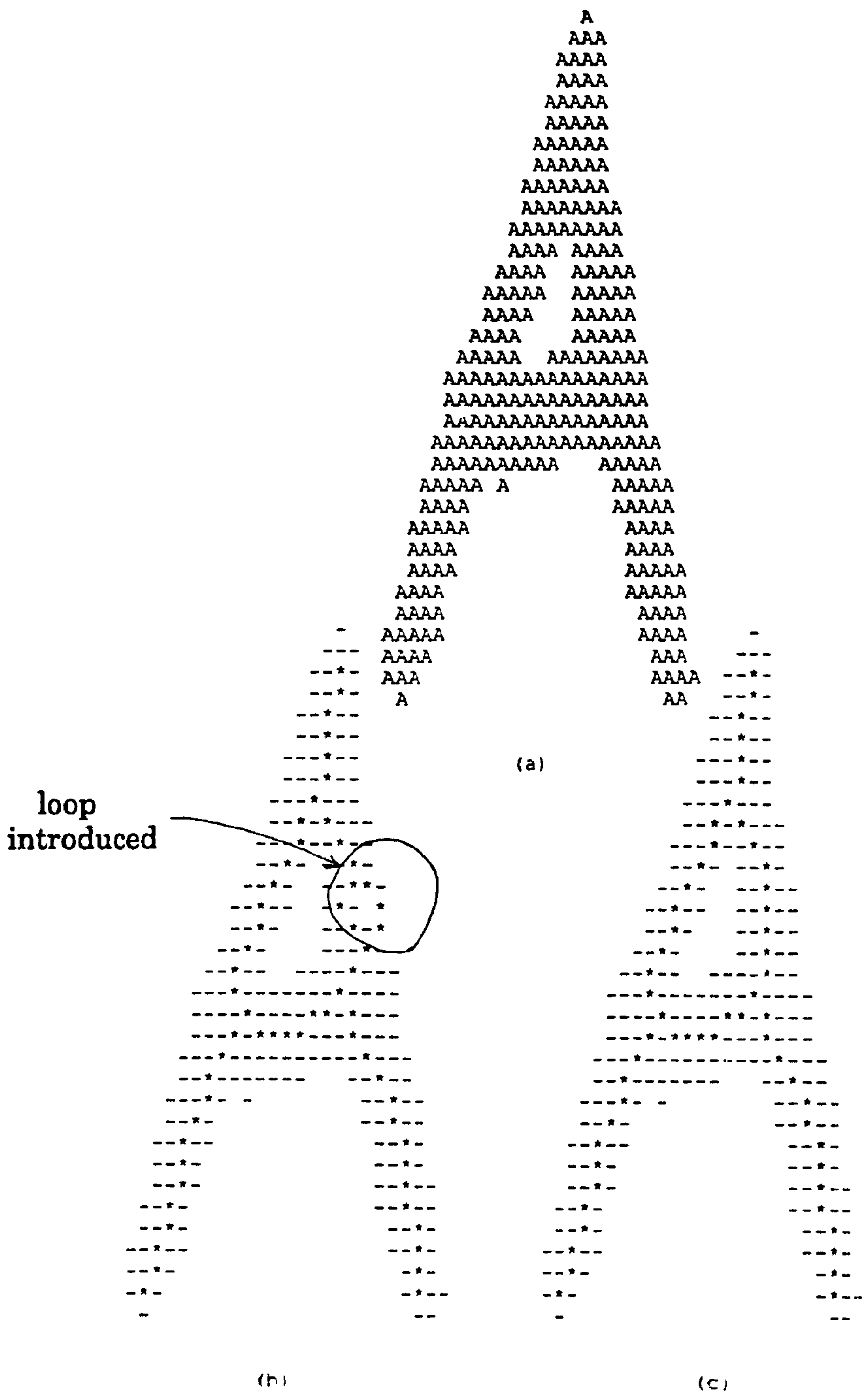


Figure 2.7: Effect of Salt Noise (CPM)
a) Original Pattern
b) Skeleton with Salt Noise
c) Skeleton without Salt Noise

```
                     A
AAAAA          AAAAA
AAAAA          AAAAA
AAAAA          AAAAA
AAAAA          AAAAA
AAAAA          AAAAA
AAAAA          AAAAA
AAAAA          AAAAA
AAAAA          AAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
AAAAA          AAAAA
AAAAA          AAAAA
AAAAA          AAAAA
AAAAA          AAAAA
AAAAA          AAAAA
AAAAA          AAAAA
AAAAA          AAAAA
```

(a)

```
                    *
-----          ----*
-----          ---*-
--*--          --*--
--*--          --*--
--*--          --*--
--*--          --*--
--*--          --*--
--*--          --*--
---*-------------*--
---*------------*---
----************----
---*------------*---
---*------------*---
--*--          --*--
--*--          --*--
--*--          --*--
--*--          --*--
--*--          --*--
-----          -----
-----          -----
```

(b)

```
-----          -----
-----          -----
--*--          --*--
--*--          --*--
--*--          --*--
--*--          --*--
--*--          --*--
--*--          --*--
---*------------*---
---*------------*---
----************----
---*------------*---
---*------------*---
--*--          --*--
--*--          --*--
--*--          --*--
--*--          --*--
--*--          --*--
-----          -----
-----          -----
```

(c)

Figure 2.8: Effect of Pepper Noise (CPM)
a) Original Pattern with Pepper Noise
b) Skeleton with Pepper Noise
c) Skeleton without Pepper Noise

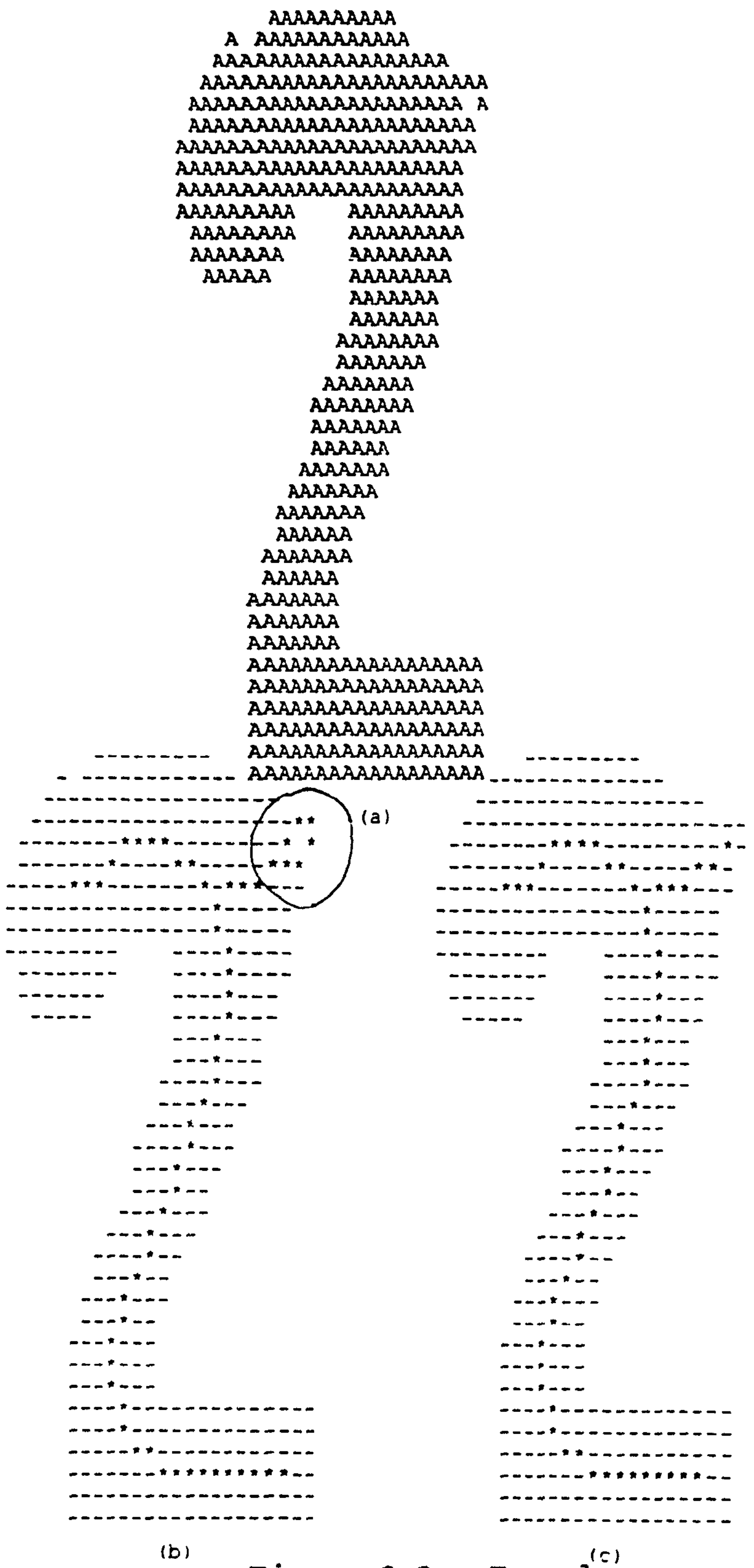


Figure 2.9: Examples

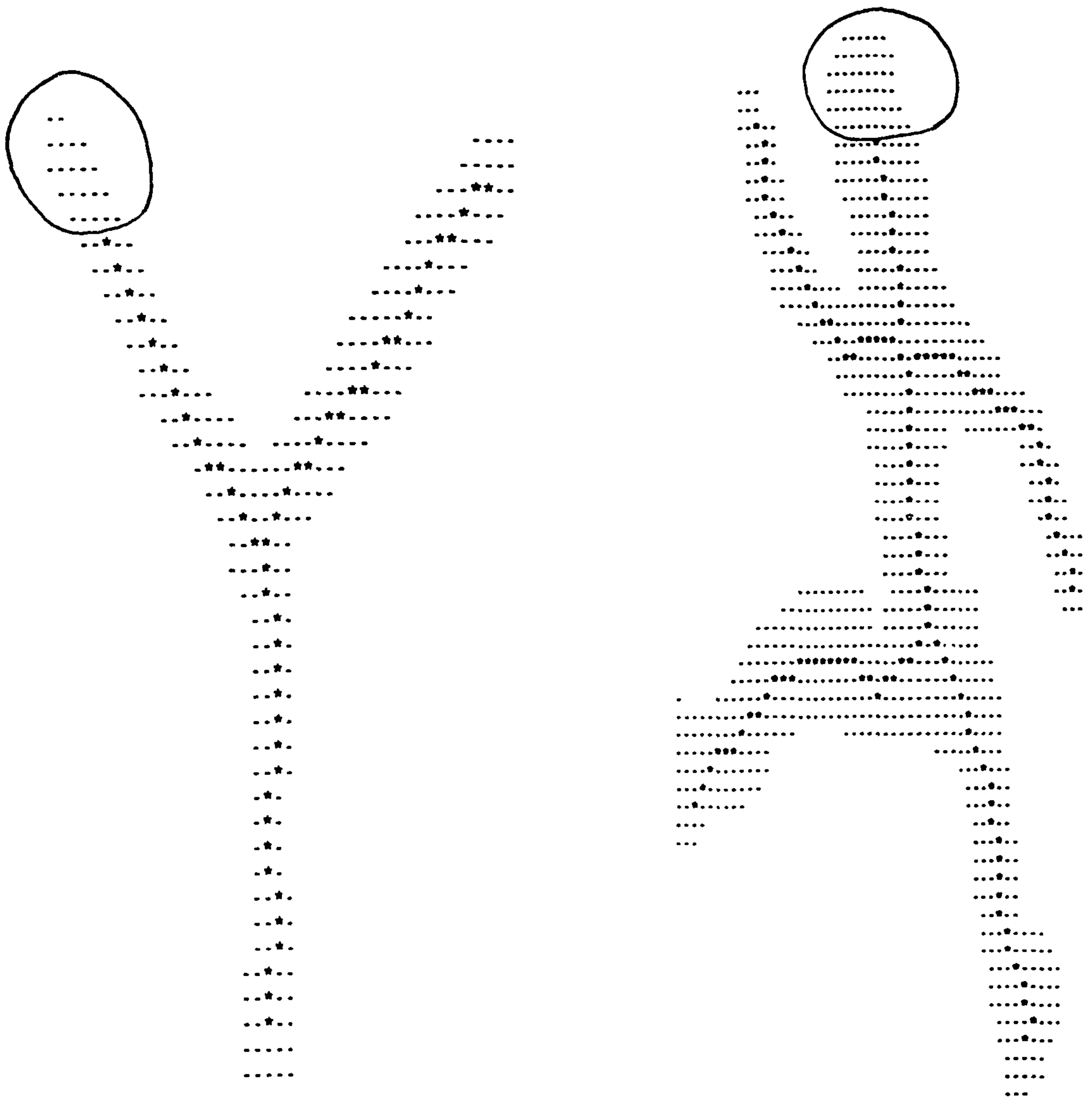


Figure 2.10: Results of CPM Illustrating End Point Erosion

2.6. Reconstructability of the Pattern

Reconstructability is one of the desirable features expected of a thinning scheme. This feature becomes useful in the data compression and transmission applications like storage of maps and facsimile transmission [4,5]. A good skeleton should keep all the topological properties of the original image. Thus starting from an ideal skeleton lying on the ridge of the Euclidean distance transformation, the original image can be reconstructed. However, in practice, the skeletons derived are not ideal since they are based on the discrete approximation of the continuous Euclidean space. Thus exact reconstruction is not feasible. There is always some salt and pepper noise present. Davies et al [32] concluded that the reconstructed pattern should lie within the original pattern. Further they stated that the reconstructed pattern with a one pixel width mismatch along the boundary of the original pattern is an <u>acceptable standard</u>. Applying the Pavlidis algorithm [33] for reconstruction to the skeletons obtained by CPM algorithm, it is found that the reconstructed images are generally of acceptable standard with some exceptions (Figure 2.11).

The Process of Reconstruction:

Initially, as described in section 2.3, thinning is performed on the input binary picture. For every dark-point in the skeleton its 4-direct white neighbours are tested. These 4-white neighbours will have either a value of (- MAX) corresponding to white pixels or a value of (i - MAX) corresponding to the dark pixels deleted from the pattern. The minimum among these values is taken and to this (MAX + 1) is added. The resulting value is assigned to the dark-point under consideration.

This value is used in performing the reconstruction of the pattern. Starting with the highest value among the skeletal points, say i_{max}, the skeleton is scanned for pixels carrying the value i_{max}. For each of these pixels the direct 4-neighbours are examined for their existence in the skeleton. If they do not exist then, they are added to the skeleton with a value of (i_{max} - 1). During the next scan the pixels with the value of (i_{max} - 1) are considered. The above process of checking for the existence of the direct 4-neighbours and adding them to the skeleton is repeated until the value of i_{max} becomes unity. The final resulting pattern made up of the pixels with values greater than zero, corresponds to the reconstructed pattern for the skeleton under consideration. The skeletons and the reconstructed patterns for the characters 'e' and 'n' are shown in Figures 2.11 and 2.12 respectively. The points of mismatch are indicated by '-' in the reconstructed image.

```
            ---                              ---
          -----------                      -1111111-1-
        ---2222222-----                  -112222222121-1
      ---22-------23----               --1221111111232121
      --2--       --3----              -1211       1232321
    ---2-         ---34---           --121         -1234321
    --2-            ---4---          1121            1234321
  ---2-             ---4---        -1221             1234321
  --3--             ---4---        12321             1234321
  --3--             ----4---       12321             -1234321
 ---4---            ---4----      1234321            1234321-
 -----4--------------33-----      12344321111111111111233321--
----5-322222222222222-------     1234543222222222222222221---
----4-----------------------     -1234321111111111111111111----
---4---                          1234321
---3--                           -12321
---3--                           -12321
---3--                           -12321
---4---                          1234321
---4---                          1234321
---4---                          1234321
---4---                          1234321
----4---                         -1234321
----5----                  --1   123454321                  --1
 ----4---               --1       12344321                 -11
 ----5----              --2-      123454321              -121
  ----5----             -2-        123454321              121
  ----55-----         --2--        12345543211         1121-
  ------5---------------2-         -12344543221-----1112121
    -----44-----------32-            1233443332111112222321
    --------44------332--            -12233444322222233221
      -------333333----                1122333333333332211
        -------------                    112222222222211
          ---------                        111111111
```

Figure 2.11: Reconstructed Pattern (CPM)

```
       ----      ---------             ---1      1111--1-1
     -------   -----------           1-1-121   -12222112121
 ---------------3333-------       ---121212321--1233332232321
 --------------22----334-----     --12323234321122222233434321
 --234----54322---------45----    -1234343454322111111223454321
   ---45-5-----        -----4---    1234545432111       112344321
   -----5----            ---4---    -123454321            1234321
     ---4---             ---4---      1234321             1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      -12321              1234321
     ---3--              ---4---      123321              1234321
   ----4---              ---4---     -1234321             1234321
   ---4-4---             ---44---    123434321            12344321
 ---33---33--          ---33--33--  112333233321        1123333321
--222--------22--    --222------22-- -12222221222221-   -12222222222221-
11-------------11    11----------11  1111111-1111111    111111111111111
```

Figure 2.12: Reconstructability (CPM)

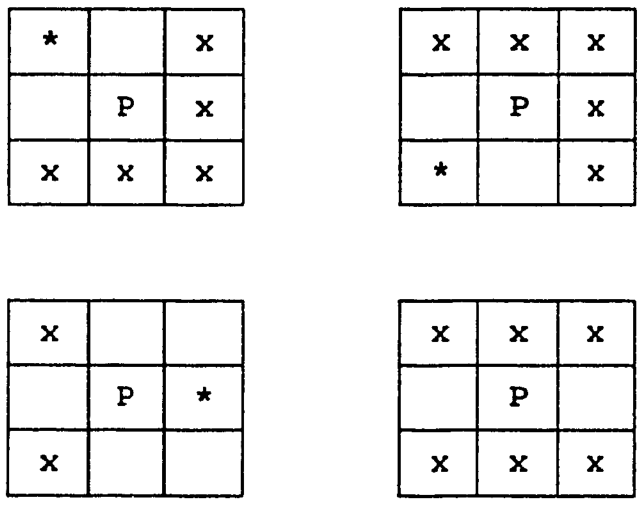
3. SELECTION OF ALGORITHMS FOR COMPARISON

In order to evaluate the performance of the proposed CPM algorithm, four existing algorithms in this area are chosen for comparison. Their selection is based on the different methodologies employed for thinning. As described in section 1.2, algorithms published in the literature can be broadly classified into two groups viz., thinning algorithms and skeletonising algorithms. Thinning algorithms are further classified depending on whether the picture is raster scanned or contour traced in obtaining the skeletons. We have selected the Safe Point Thinning Algorithm (SPTA) since it is a raster scan based thinning algorithm and has been proved to be the fastest [15]. Our second selection is based on the contour tracing which uses the Distance Transformation [18] and is referred as DT. Similarly for the skeletonising methodology, a modified versions of the Maximum Square methodology (MSM) [34] and the Maximum Circle methodology (MCM) [34] have been chosen. In the following section all the above four algorithms and their implementation details are described. All the algorithms are implemented on Cyber CDC830D computer using Pascal language.

3.1. Safe Point Thinning Algorithm (SPTA)

3.1.1. Description of the Algorithm

The Safe Point Thinning algorithm was implemented by Nabil J Naccache [15]. This algorithm is based on the raster scanning of a binary pattern. It produces skeletons with no spurious tails if the input is a smoothed pattern. This algorithm first classifies each edge-point to be either left, right, top or bottom edge-points as described in Chapter 1. For each of these edge-points a Boolean expression is specified which would check if the point is a "safe-point". A safe-point is either an end-point or a break point and the removal of it from the pattern would result in excessive erosion or break the connectedness of the pattern. These Boolean expressions are based on the four windows shown in Figure 3.1, which are the only configurations to be tested for the end-point, break-point and excessive erosion conditions.



x: Don't cares *: Dark points

Figure 3.1: Safe-Point Pixel Configurations [15]

The algorithm consists of two scans per pass and the labelling technique used helps in the reconstruction of the original pattern. In the first scan the left and right edge-points are processed, and in the second scan only the top and bottom edge-points are considered. Scanning can be done in either rowwise or columnwise fashion. The thinning process consists of a number of passes and in each pass a set of dark points are flagged for their deletion from the pattern. The steps involved in each pass are as follows:

Step 1: Scan the binary pattern. Once a dark point is encountered check it for edge-point condition. If the dark point is an edge-point go to step 2, else continue the scanning.

Step 2: Depending on the type of the edge-point (left or right), apply the appropriate boolean expression given below. If the value of S is false, then the dark point under consideration is a safe-point and hence is not flagged for removal.

For **Left Safe-Point:**

$$S = P3.(P4+P5+P1+P2).(P5+\overline{P6}).(P1+\overline{P8}) \qquad (3.1)$$

For **Right Safe-Point:**

$$S = P7.(P8+P1+P5+P6).(P1+\overline{P2}).(P5+\overline{P4}) \qquad (3.2)$$

where P's are dark points and P's are white points.

Step 3: Repeat steps 1 and 2 till the very end of the pattern is reached.

Step 4: Start the second scanning of the same pass and repeat steps 1, 2 and 3 looking for only the top or bottom edge-points now. Identify the safe-points and flag those top and bottom edge-points that are not safe-points by using the appropriate Boolean Expression given below:

For Top Safe-Point:

$$S = P5.(P6+P7+P3+P4).(P7+\overline{P8}).(P3+\overline{P2}) \qquad (3.3)$$

For Bottom Safe-Point:

$$S = P1.(P2+P3+P7+P8).(P3+\overline{P4}).(P7+\overline{P6}) \qquad (3.4)$$

Repeat the steps 1, 2, 3 and 4 until no single flagged point is encountered in a pass. Then the thinning process stops.

3.1.2. Implementation of the Algorithm

The input binary pattern is read onto a matrix of integers with the dark points assigned a value of '0' and the white points a negative value of, say (- MAXINT), where MAXINT is the largest number that can be stored in the computer. As stated earlier each pass consists of two scans and in each scan only the edge-points are tested. Each pass consists of the following two steps.

Step 1: In the first scan the right and left edge-points are identified. The Boolean expressions (3.1) and (3.2) are evaluated depending on the type of edge-point to determine if it is a safe-point or not. If it is a safe-point, then it is assigned a value of 'i', which corresponds to the pass number and a non-safe-point is assigned a negative value of (i - MAXINT).

Step 2: Similarly in the second scan of the same pass, the top and bottom edge-points are identified and tested for a safe-point or not using the equations (3.3) and (3.4) and assigned values accordingly.

Thus at the end of a pass we have a set of white-points marked with value (- MAXINT), flagged non-safe-points marked with a value of (i - MAXINT), safe-points marked with value i and dark-points marked with value 0. During the next pass, only the edge-points with value 0 are processed. Those points below 0 are automatically considered as white points and the necessity to have a third scan in each pass to delete the flagged non-safe-points is avoided. The points having a value greater than zero form the skeleton.

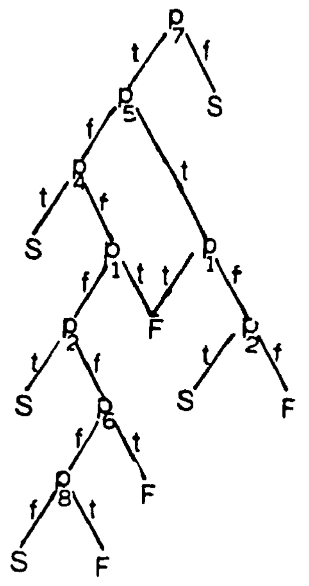
In order to reduce the processing time, the following refinements are done:

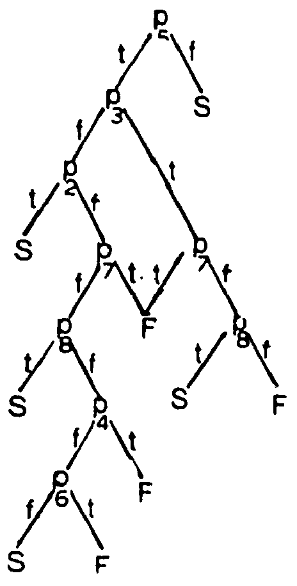
(a). Minimising number of scans: As described in section

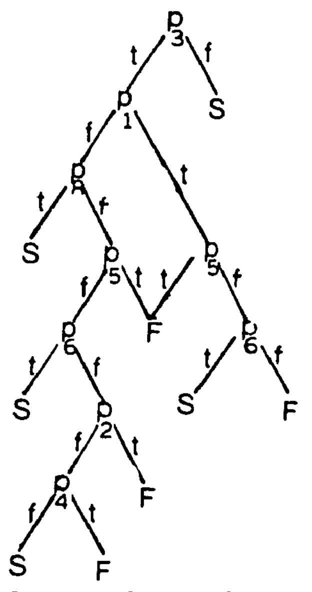
3.1.1., each pass consists of two scans and the first scan flags only the left and right edge-points, whereas the second scan flags only the top and bottom edge-points. If in the first scan of a pass, only the left edge-points are flagged then during the first scan of the next pass only the left edge-points have to be tested for flagging. Also if no points were flagged during a scan, say the second scan of a pass then, during the next pass only the first scan has to be initiated. This is because if the second scan did not flag any points in a pass, then it will not flag any point in the subsequent pass.

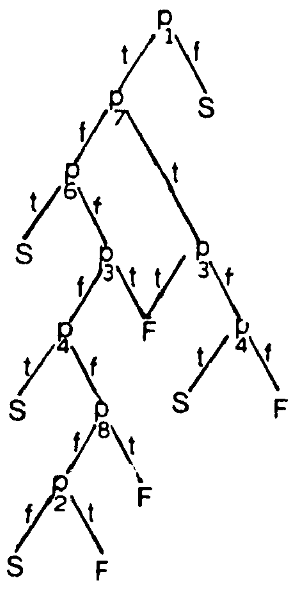
(b). Testing Boolean expressions: The Boolean expressions (3.1) to (3.4) given in section 3.1.1 are evaluated by using the decision trees given in Figure 3.2, thereby reducing the number of points to be checked.

(c). Labelling technique for reconstruction: The labelling technique used in the implementation has the following two advantages: In each pass only the dark points with value 0 are considered for processing. This eliminates the necessity for having a third scan in each pass to delete the flagged non-safe-points which are assigned a negative value of (i - MAXINT). Secondly this labelling technique helps in the reconstruction of the original pattern as described later in section 3.1.3.


a) Tree for Right Safe-Point


b) Tree for Top Safe-Point


c) Tree for Left Safe-Point


d) Tree for Bottom Safe-Point

Pj: 8-neighbour being visited
S : Safe-Point
F : Flag-Point
t: Denotes Boolean value True
f: Denotes Boolean value False

Figure 3.2: Tree Structure for Boolean Expressions [15]

3.1.3. Analysis of the Algorithm

The skeletons produced by SPTA are smooth and have well preserved connectivity. They are of single width thickness and have no redundant pixels. The binary patterns and the corresponding skeletons generated by the algorithm for characters 'A' and 'H' with salt and pepper noise are shown respectively in Figures 3.3 and 3.4. It has been found that the spurious tails are generated due to such noise. Hence the input binary pattern has to be preprocessed. The resulting skeletons are also affected by small protrusions along the border of the input binary pattern as can be seen from the skeleton generated for the character '2' in Figure 3.5.

Since the algorithm is based on the raster scanning of the pattern, the presence of islands or composite characters do not affect the speed of the algorithm "drastically" and hence makes it suitable for Tamil and Chinese languages.

In case of the lines with double width, the algorithm generates fairly good results as illustrated by Figure 3.6. The algorithm is also isotropic as the skeletons generated for various angles of rotation are the same.

The algorithm has the ability for reconstruction. The Reconstruction algorithm used is the same as that of Pavlidis [33], described in Chapter 2. It has been found that the

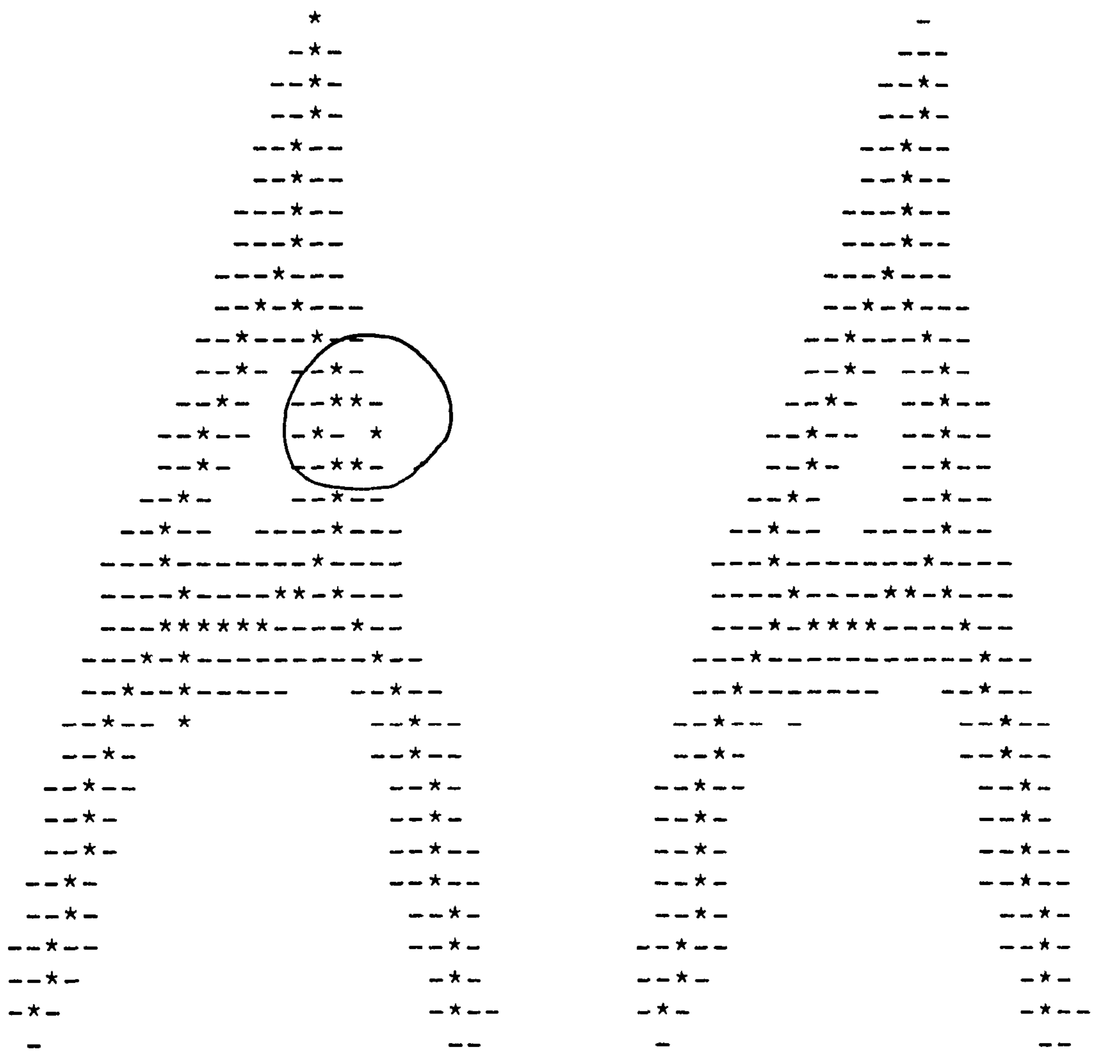

(a) (b)

Figure 3.3: Effect of Salt Noise (SPTA)

```
                    *
-----          ----*               -----          -----
--*--          ---*-               --*--          --*--
--*--          --*--               --*--          --*--
--*--          --*--               --*--          --*--
--*--          --*--               --*--          --*--
--*--          --*--               --*--          --*--
--*--          --*--               --*--          --*--
--*--          --*--               --*--          --*--
---*-------------*--               ---*------------*---
---*------------*---               ----*-----------*---
----************----               -----***********----
---*------------*---               ----*-----------*---
---*-------------*--               ---*------------*---
--*--          --*--               --*--          --*--
--*--          --*--               --*--          --*--
--*--          --*--               --*--          --*--
--*--          --*--               --*--          --*--
--*--          --*--               --*--          --*--
--*--          --*--               --*--          --*--
-----          -----               -----          -----
```

Figure 3.4: Effect of Pepper Noise (SPTA)

```
       ---------
    --------------
   ------------------
  ----------------------
 -------*******------*--
 ----*-*------*----*---
------*--------*-**----
-----*----------*-----
-----*-----------*----
---------    ----*----
 --------    ----*----
 -------     ----*---
  -----      ----*---
             ---*---
             ---*---
            ----*---
            ---*---
           ---*---
          ----*---
          ---*---
          ---*--
         ---*---
        ---*---
       ---*---
       ---*--
      ---*---
      ---*--
     ---*---
     ---*---
     ---*---
     ----*--------------
     ----*--------------
     -----**------------
     -------*********---
     -------------------
     -------------------
```

Figure 3.5: Effect of Boundary Noise (SPTA)

```
                -*         -*                     -*
                -*          -*                    -*
                -*           -*                   -*
                -*            -*                  -*
-----------     -*             -*            ------*-----
-*********-     -*              -*           -**********-
                -*               -*               -*
                -*                -*              -*
                -*                 -*             -*
                -*                  -*            -*
```

```
                                        -    -    -
                                         *   *   *
                                          *  *  *
                                           * * *
                   -*-       --*--          ***
         -*        -*-       --*--     -***********-
         -*        -*-       --*--          ***
                                           * * *
                                          *  *  *
                                         *   *   *
                                        -    -    -
```

Figure 3.6: Skeletons for Typical Test Patterns (SPTA)

reconstructed pattern is of acceptable standard. The skeletons and the reconstructed patterns are shown in Figures 3.7 and 3.8 for characters 'e' and 'n'.

3.2. Sequential Thinning of Binary Patterns using Distance Transformation (DT)

3.2.1. Description of the Algorithm [18]

This is a sequential thinning algorithm based on the contour tracing technique. This algorithm makes use of the distance transformation information in order to avoid the overshrinking at the end points. The algorithm consists of the following steps:

Step 1: Apply the "4-neighbour distance transformation" for the input binary picture. According to this transformation, the density value of every dark pixel will become the distance of that pixel to the nearest white pixel i.e. the distance is determined as the length of the path from the dark point to the nearest white point where a path consists of horizontal and vertical steps of unit length.

Step 2: Scan the resulting picture of step 1 till a dark pixel with a density value of one is encountered. Starting from this point, trace the border in a clockwise direction. Delete the border points which satisfy all the following conditions (a) through (d). If either condition (a) or (b)

```
            ---                            ---
          -----------                    -111111111-
        ---22222222----                -11222222222111
      ---22--------3----             --1221111111232221
      --2--       --33--             -1211       1233321
    ---2-         ----4---         --121         -1234321
    --2-            ---4---        1121            1234321
  ---2-             ---4---      -1221             1234321
  --3--             ---4---      12321             1234321
  --3--             ----4---     12321             -1234321
 ---4---            ---4----    1234321            1234321-
 ----4--------------33-----     1234432111111111111233321--
----5-32222222222222-------    123454322222222222222222221---
----4----------------------    -12343211111111111111111----
---4---                        1234321
---3--                         -12321
---3--                         -12321
---3--                         -12321
---4---                        1234321
---4---                        1234321
---4---                        1234321
---4---                        1234321
----4---                       -1234321
----5----                -1-   123454321                 11-
 ----4---               -2-     12344321                121
 ----5----             --2-     123454321              -121
  ----5----            -2-       123454321             121
  -----5-----        ---2-       -1234543211         11121
  ------5--------------22-       --1234543221-----1112221
    -----4------------3--          -12344333211112222321
    -------444-----3332--          --12334443222233221
      --------33333-----             -1223333333322211
        -------------                  1122222222111
          ---------                      111111111
```

Figure 3.7: Reconstructed Pattern (SPTA)

```
         ----       ---------                   1111      11111-111
      -------    -------------                1122221   -12222212221
  ------------------33333------           --1122333321-11233333233321
  --2-----------222------344----          -122234444321222222223444321
  ---3445-55432-------------5----         -1234455554321111111123454321
    -----6-------       -----4---           12345654321-      -12344321
    -----5----            ---4---           -123454321          1234321
      ---4---             ---4---             1234321           1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             -12321            1234321
      ---3--              ---4---             123321            1234321
    ----4---              ---4---           -1234321            1234321
    ---4-4---             ---44---          123434321           12344321
  ---33---33--          ---33--33--       112333233321        1123333321
--222--------22--     --222-------22--  -12222221222221-    -1222222222221-
-1-------------1-     -1------------1-  -1111111-111111-    -11111111111111-
```

Figure 3.8: Reconstructed Pattern (SPTA)

does not hold, then the dark point under consideration is a final point.

(a) The 8-connectivity number of the border point under consideration is one.

(b) The number of dark pixels in the 8-neighbourhood is more than one.

(c) There exists no dark 4-neighbour which satisfies the condition "the value of the border point minus the value of the dark 4-neighbour is one".

(d) The border point under consideration is not a final point.

The density values of the deleted pixels are changed into zero and those of the final points are set to a large value greater than the maximum distance value. This process of border tracing and peeling continues until no more pixels can be deleted.

Step 3: Check the input picture for the existence of any holes or islands. If an is island present in the picture, then repeat step 2 starting from the interior border and working towards the exterior and tracing the border in a counter-clockwise direction.

Step 4: This is a postprocessing stage where the two pixel wide skeletons are reduced to single pixel wide lines. This is achieved by scanning the resulting picture of steps 2 and

3, deleting the dark points which are not candidates for final points and satisfying condition (a). Finally the dark points that are retained form the skeleton of the input picture.

3.2.2. Implementation of the DT Algorithm

The scanned binary picture is represented as a matrix of integers, where the dark points are assigned a value of 0 and the white points are assigned a value of (- MAXINT). This initial binary picture is then distance transformed [36] using step 1 of section 3.2.1.

The transformed matrix is scanned from left to right and top to bottom. Once a dark point is encountered, the contour tracing is started and is done in clockwise direction. The dark points along the contour are tested against the conditions (a) through (d) given in step 2 of section 3.2.1. If the border point under consideration is a final point then it is assigned a value of MAXINT, and for the deletable points the value assigned is 0. the contour tracing is continued until there are no deletable pixels in a complete round of the contour.

The raster scanning is continued looking for islands and additional characters in the input matrix. If there are islands present then the contour tracing and peeling are done for each island in a counter-clockwise direction starting from the interior boundary and working towards the exterior. Similarly for each of the additional character in the input pattern, the exterior and interior (if an island is present) contour tracing and border peeling are carried out.

Finally the postprocessing is done to reduce the two pixel wide skeletons to single pixel wide skeletons. This is achieved by scanning the resulting pattern and deleting the dark points which are not final points and yet they satisfy condition (a) in step 2 of section 3.2.1. The skeleton is formed from the dark points with a value of MAXINT. The distance transformed pattern and the skeleton generated by the algorithm are shown in Figure 3.9.

3.2.3. Analysis of the DT Algorithm

The border peeling and the prevention of excessive shrinking result in medial line for most cases. For patterns with protrusions on the border, as in character '2' of Figure 3.10, this algorithm produces noisy branches. Hence, the algorithm needs a "shape smoothing", prior to the thinning process.

Presence of salt and pepper noise also tends to deteriorate the performance of this algorithm. Figures 3.11 and 3.12 show respectively the effect of salt and pepper noise on the skeletons generated by the algorithm. The results obtained after smoothing [17] are better in the sense that there is a

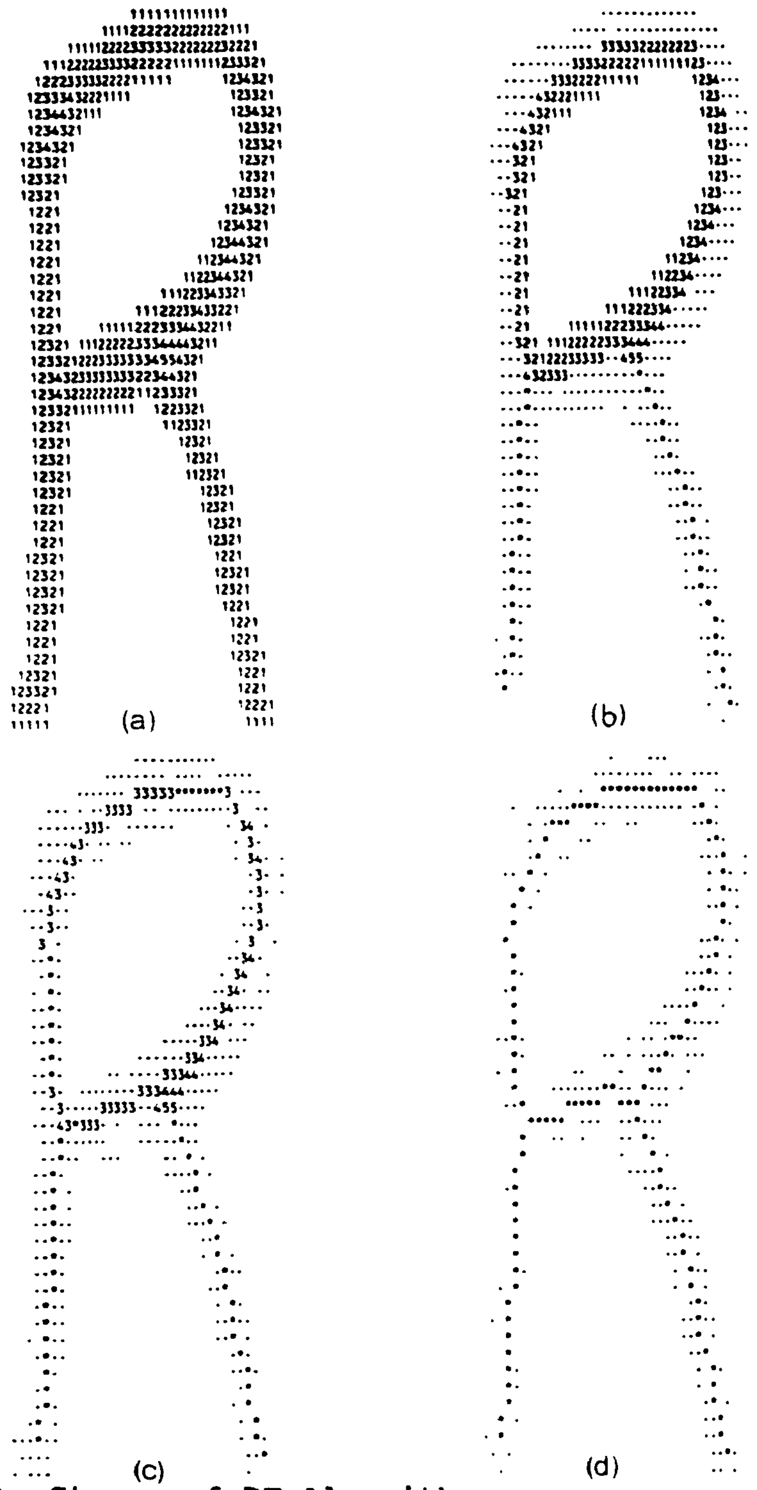


Figure 3.9: Stages of DT Algorithm
a) Distance Transformed Pattern
b) Results After Processing Outer Contour Loops
c) Results After Processing Inner Contour Loops
d) Final Derived Skeleton After Post-processing

```
       ---------
    --------------
   ------------------
  ----------------------
 -------******------***-
 ---*--*------**---*---
-----**---------***----
----------------*-----
-----------------*----
---------    ----*----
 --------    ----*----
 -------     ----*---
  -----      ----*---
             ---*---
             ---*---
            ----*---
            ---*---
           ---*---
          ----*---
          ---*---
          ---*--
         ---*---
        ---*---
       ---*---
       ---*--
      ---*---
      ---*--
     ---*---
     ---*---
     ---*---
     ----*--------------
     -----*-------------
     ------*------------
     -------*********---
     -------------------
     -------------------
```

Figure 3.10: Effect of Boundary Noise (DT)

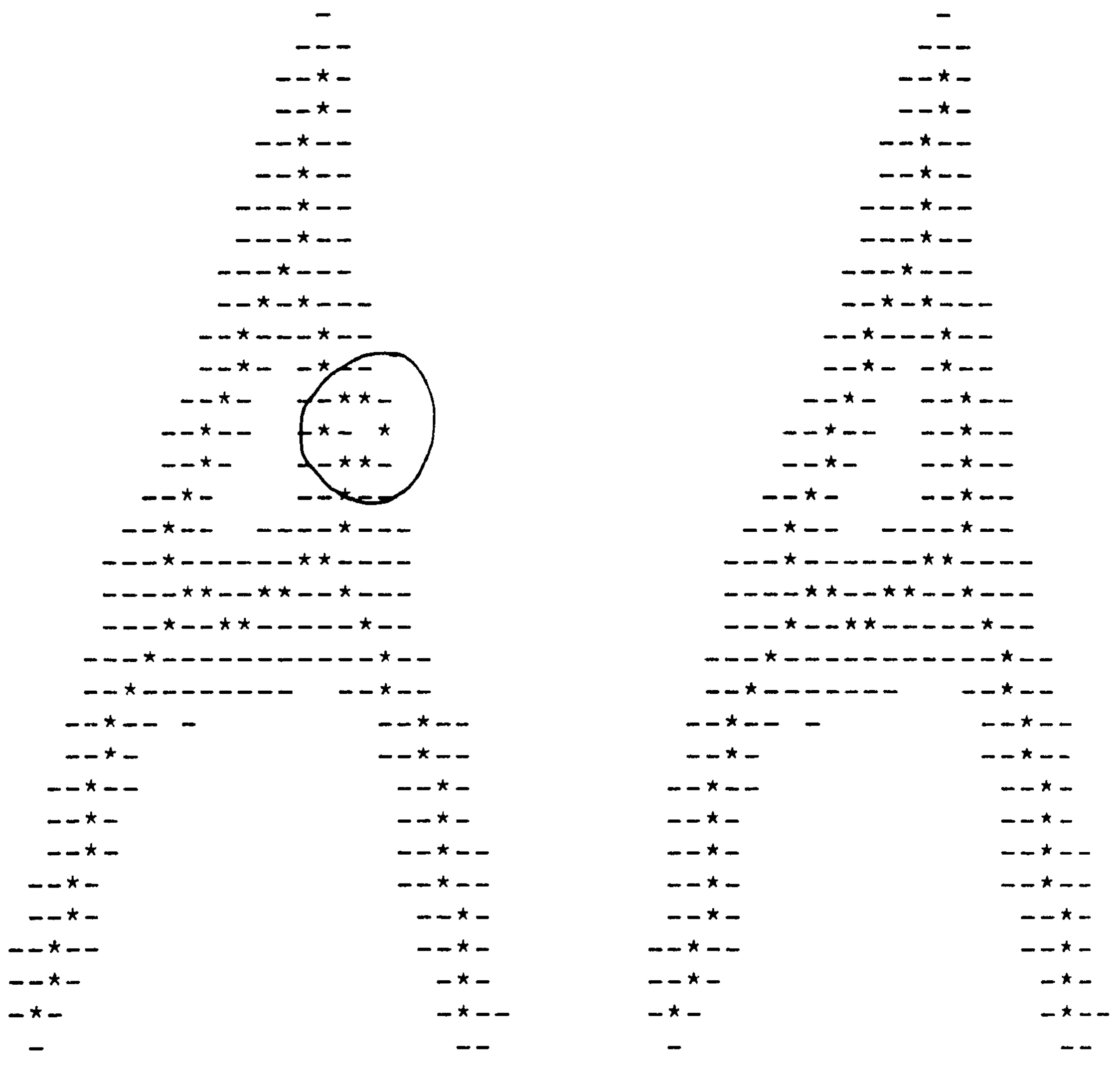


Figure 3.11: Effect of Salt Noise (DT)

```
                    *
-----          ----*                    -----          -----
-----          ---*-                    -----          -----
--*--          --*--                    --*--          --*--
--*--          --*--                    --*--          --*--
--*--          --*--                    --*--          --*--
--*--          --*--                    --*--          --*--
--*--          --*--                    --*--          --*--
--*--          --*--                    --*--          --*--
---*-------------*--                    ---*------------*---
----*-----------*---                    ----*----------*----
-----***********----                    -----**********-----
---*------------*---                    ----*----------*----
---*-------------*--                    ---*------------*---
--*--          --*--                    --*--          --*--
--*--          --*--                    --*--          --*--
--*--          --*--                    --*--          --*--
--*--          --*--                    --*--          --*--
--*--          --*--                    --*--          --*--
--*--          --*--                    --*--          --*--
-----          -----                    -----          -----
```

Figure 3.12: Effect of Pepper Noise (DT)

reduction in the number of noisy branches produced. In the case of patterns with islands and composite characters the algorithm takes more time for processing, because additional border tracing and peeling processes are required. Hence, this algorithm is optimal for languages like Tamil and Chinese as the majority of characters in these languages have multiple characters and one or more islands.

The Distance transformation algorithm requires only four scans of the pattern and border tracing(s) irrespective of the width of the binary patterns. The processing time is found to increase slowly with an increase in the width of the binary pattern. Table 3.1 shows the processing time in CPU seconds for a rectangle as the width is increased from 2 to 10. Hence this method is best suited for patterns which contain highly varying widths.

On the average this method is found to be the fastest among all the five algorithms under consideration. The DT method is however slower than the CPM methodology in the case of thin input patterns of width less than 5 pixels.

By making use of the distance transformation values corresponding to the skeletal points, the original pattern can be reconstructed by applying a reverse distance transformation.

Table 3.1: CPU Times for Thinning Rectangles of 40 x n

n	CPM	DT	SPTA	MSM	MCM
10	0.622	0.404	1.170	1.202	2.005
9	0.540	0.384	1.055	1.108	1.721
8	0.485	0.344	1.012	1.037	1.610
7	0.407	0.327	0.928	0.961	1.421
6	0.360	0.291	0.887	0.906	1.224
5	0.289	0.269	0.814	0.851	1.102
4	0.246	0.236	0.771	0.845	1.031
3	0.190	0.217	0.714	0.825	0.920
2	0.156	0.178	0.695	0.821	0.845

The original and the reconstructed pattern are shown in Figures 3.13 and 3.14 for characters 'e' and 'n' respectively. It is found that the reconstructed patterns have more mismatch points than the CPM.

3.3. Tracing Centre-Lines of Digital Patterns using Maximal Square Moving Algorithm (MSM)

3.3.1. Description of the Algorithm

The Maximal Square moving methodology proposed by Wakayama [37] generates a structural description of the skeletons obtained from the input binary pattern. This algorithm is based on the medial axis transformations applied to discrete images. It is distinct from the conventional methodolc:ies in that it is based on an "input-time tracing principle". A modified version of the algorithm which was implemented and tested by Abairid [34] is used in this thesis for the comparative study. Squares of zero width are initially formed which are then expanded to higher sizes as the scanning proceeds. Hence the name "Maximal Square Moving". The modified version allows the maximal squares to have a few white points on the periphery of the squares and thereby reduces the number of noisy branches in the skeletons as it accommodates ruggedness along the edges of the pattern. The core lines are successively developed as the input binary pattern is read in row-major form and by simultaneously moving maximal squares across the pattern. The algorithm operations

```
            ---                             ---
          -----------                     -1111111-1-
        ---2222222-----                 -112222222121--
      ---22-------23----              --1221111111112321--
      --2--       --3----             -1211       123211-
    ---2-         ---34---          --121         -123221-
    --2-            ---4---         1121            123321-
  ---2-             ---4---       -1221             1234321
  --3--             ---4---       12321             1234321
  --3--             ---4----      12321             1234321-
 ---4---            --3-----     -123321            123321--
 ----432------------23------     1234432111111111111123221---
----5---222222222222--------    12345432222222222222221211----
----4-----------------------    -12343211111111111111-1------
---4---                         1234321
---3--                          -12321
---3--                          -12321
---3--                          -12321
---4---                         1234321
---4---                         1234321
---4---                         1234321
---4---                         1234321
----4---                        -1234321
----5----                 -22   123454321                 111
 ----4---                -2-     12344321                121
 ----5----              -2--     123454321              121-
  -----5----            -2-       123454321             121
  ------5-----        --2--       -1234543211         1121-
  ------54-------------2--        --1234543221111111111221-
    ------4----------23--           -12343433222222222321
    -------4433333333---            --1232344333333333221
      -----------------               -121233222222222221
        -------------                   1-12211111111
          ---------                       -11--------
```

Figure 3.13: Reconstructed Pattern (DT)

```
        ----       ---------                   -111     111111111
       -------   ------------               1-12221   112222222221
  --------------22333333-----           -1-121233321112233333333321
  -2----------22--------44----          1212323444322211222233444321
  --234---5543-----------5----          -1234343454321--1111123454321
    ---456------       -----4---          12345654321-      -12344321
    -----5----           ---4---          -123454321          1234321
       ---4---           ---4---            1234321           1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            -12321            1234321
       ---3--            ---4---            122321            1234321
    ----4---             ---4---          -1234321            1234321
    ---4-43--            ---443--         123434321           12344321
  ---33----32--        ---33---32--      112333232321        11233323321
--222--------222     --222-------222  -122222212121211    -12222222121211
22-------------      22------------   11111111-1-1-1--    111111111-1-1--
```

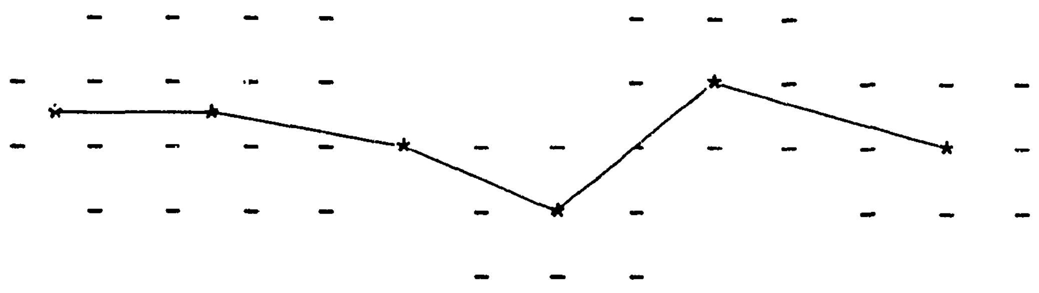
Figure 3.14: Reconstructed Pattern (DT)

are composed of the definitions of a square and enlargement of square. It has the advantage of reconstructability of the original pattern from the core lines.

Consider the digitised binary pattern shown in Figure 3.15. The ordered pair <(i,j),l> is used to denote a square where i and j correspond to the row and column of the upper left most corner of the square and l is the side length. In Figure 3.15 as the first row is input, two squares <(1,3),0> and <(1,10),0> are initially defined. When the second row is input, the squares <(1,3),0> and <(1,10),0> are enlarged to squares <(1,3),1> and <(1,10),1> respectively. Similarly when the third row is input, the above squares are enlarged to <(1,3),2> and <(1,10),2>. When the fourth row is input the square <(1,3),2> is enlarged to <(1,3),3>. However, the square <(1,10),2> cannot be enlarged and it is the only maximal square at that point. The pixels on the sides of the maximal squares are further used to define and derive new squares. Enlarging operations are carried out on these newly derived squares. Thus from the maximal square <(1,3),3> two new squares <(2,2),1> and <(3,7),0> are derived. From the maximal square <(1,10),2>, the square <(2,12),2> is derived. The derivation from <(3,7),0> and <(1,10),2> leads to the maximal square <(3,3),2>. The adjacent maximal squares are successively connected to each other and the core-lines are formed by joining the centres of the adjacent squares as shown

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

Figure 3.15: Digitised Binary Pattern



- :Original Pattern * :Skeletal Points

Figure 3.16: Core-line Representation

in Figure 3.16. Thus the core-line connectivity is defined in terms of the maximal square adjacency relation instead of the 4 or 8 neighbour connectivity.

3.3.2. Implementation of the MSM Algorithm [35]

As a first step, the input binary pattern is stored in a link-list structure instead of the matrix representation by using the following definition of Groups.

Group: A group is a set of consecutive dark points in the same row and each row may contain more than one group. Each group consists of a beginning point B and an ending point E. Let Mat(K,N) be the input matrix of size KxN.

The mth group G in the Jth row is defined as

$$G_J^m = \{ \mathrm{Mat}(J,P) \mid \mathrm{Mat}(J,P) = 1 \}$$

where

$$P = B_J^m,\ (B+1)_J^m,\ \ldots\ldots\ E_J^m.$$

With reference to Figure 3.15, there are 9 groups for the input pattern. Row 1 has 2 groups, Row 2 has 2 groups, Row 3 has 1 group, row 4 has 3 groups and row 5 has 1 group. This input pattern can be finally represented by a link-list structure as shown in Figure 3.17, by storing the beginning B and ending E points of all the groups, taking care to maintain the adjacency between groups of the same row and between rows

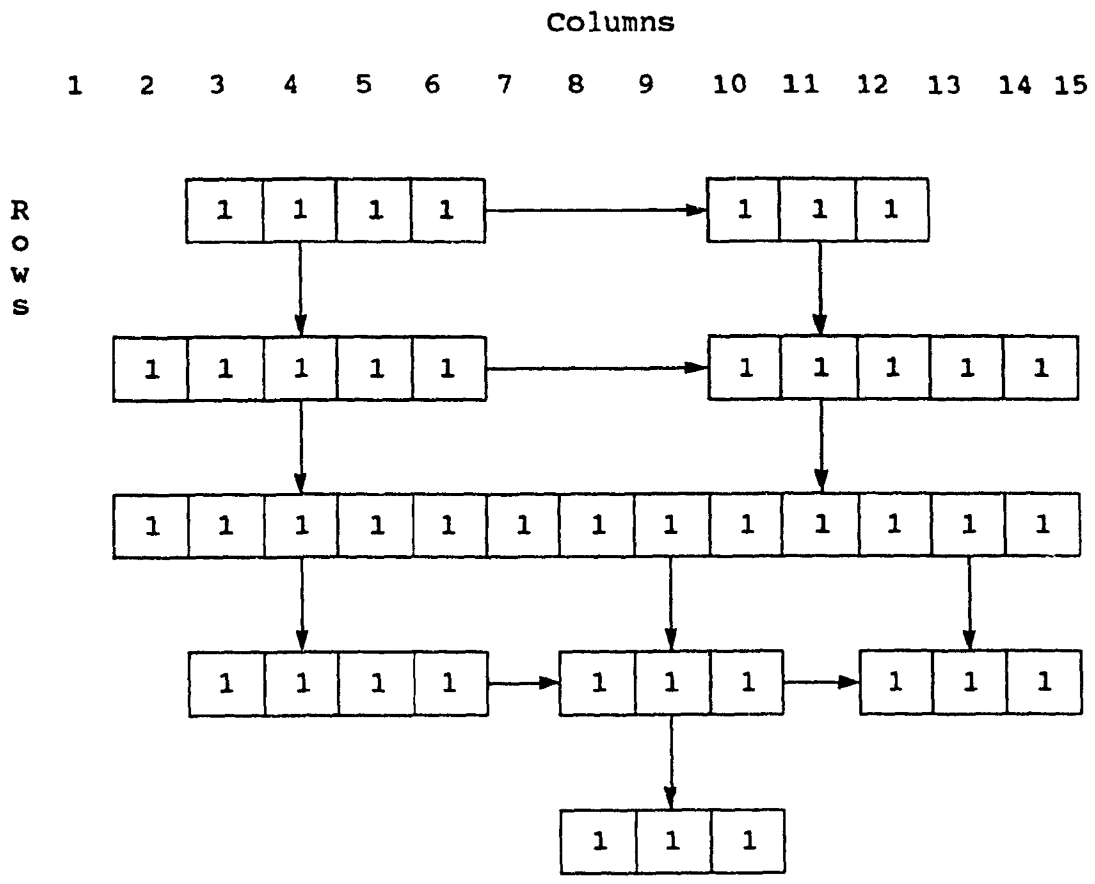


Figure 3.17: Link-list Representation for Figure 3.15

by applying the Rule 1 given below. This link-list structure is used to define the squares.

Rule 1:

$$\{ (B_{J+1}^{m} \le E_{J}^{n}) \quad \text{and} \quad (E_{J+1}^{m} \ge B_{J}^{n}) \}$$

where

J represents the row number and m,n are group indices.

Step 1: Define a Square:

At the beginning of each group in every row, a square of size zero width is initially defined.

Step 2: Enlarge the Squares:

The above defined squares of unit size are further enlarged by using the B and E of the current group in the J^{th} row along with a group in the $J+L^{th}$ group where L satisfies the Rule 1 and Rule 2 given below. The value of L is incremented by 1 if both Rules 1 and 2 hold and another group from the row J+L is tested by rules 1 and 2 for a possible bigger square. This iterative process of enlarging the square continues until one or both the rules fail. At this point L also represents the number of groups comprising the square. When the squares cannot be further enlarged then go to step 3.

Rule 2:

a : $\{ (B_{J+L}^{m} \le P_{L}^{n}) \quad \text{and} \quad (E_{J+1}^{m} > P_{J}^{n} + L) \}$

b : The pixel at position (P+L) in all groups comprising a size (L) square must be black where

m,n : group indices in rows J and J+L

P : B, B+1, ..., E-1, E.

Step 3: Redundancy Check for the Squares Derived:
The newly derived square is added to the maximal square list depending on whether it is a duplicate or is inside a larger square previously defined and derived.

Step 4: Propagation:
Advance to the next pixel of the current group and repeat steps 1 through 3 unless the following condition is true:

<u>Maximal Square Condition:</u>
The Ending point (E) of the first group in the square is less than or equal to (P+L).

If the above condition is true then advance to the group in the next row which satisfies the Rule 1 and invoke step 1.

It is found that the presence of spurious tails in the output can be eliminated or reduced by having a few white pixels on the periphery of the maximal squares. However, the number of white pixels allowed on the periphery should be $\leq$ L, where L is the size of the maximal square. The outputs for a pattern is shown in Figure 3.16. It is found that the maximal square centres do not always lie on the pixel in the binary picture due to the following reasons:

(C1). The centre of the maximal squares of even pixel width (2,4,..) cannot be represented in digital co-ordinates.

(C2). Overlapping squares are not always the same size.

The pattern is expanded so that the centres of the maximal squares fall on a pixel. This takes care of the problem C1 stated above. Secondly to provide 8-connected centre-lines, a filling algorithm is used which is explained in the following section.

Filling Algorithm: The implementation considered makes use of a 66-pixel window as shown in Figure 3.18a. Generally the size of the window is chosen based on the longest path between the two centres of the maximal squares encountered.

Step 1: The window is moved along the pattern from one maximal square to the other from left to right and from top to bottom. As the window is moved the current maximal square's centre is aligned with the point 'C' in the window.

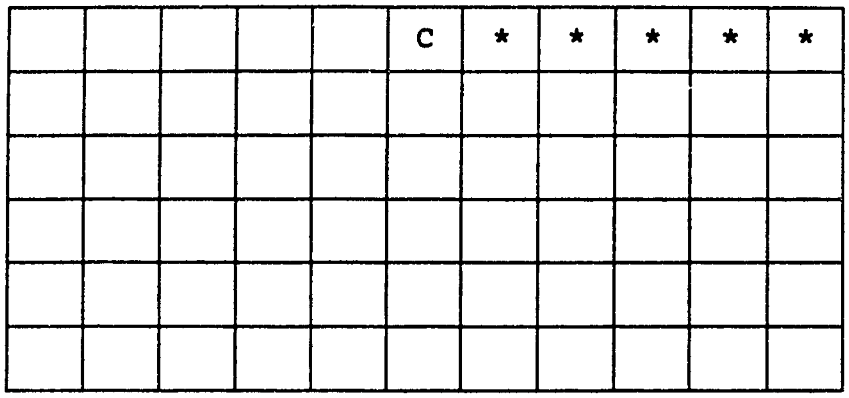
Step 2: If a maximal square's centre 'S' occurs within the points marked as '*' and the 'C' then fill the gap between the 'C' and the 'S' with '*'.

Step 3: Ignoring the first row of the window, look for the occurrence of another maximal square centre within the window with the shortest path from point 'C'. If this is found then fill the gap. If there exists two centres which are equi-distant from the point 'C' and the distance between them is greater than 3 pixels, then fill both the paths; otherwise fill the gap between the point 'C' and the leftmost centre.

Step 4: Repeat steps 1 through 3 until all centres are covered. An example of the filling algorithm is shown in the Figures 3.18b and 3.18c, for both the original and the expanded pattern.

3.3.3. Analysis of the MSM Algorithm

In the implementation of this methodology, instead of the conventional way of storing the binary pattern in a matrix form, a link-list structure is employed and so the image memory required is less. However, the skeletons generated and the processing speed very much depend on the sequence in which the binary pattern is read. Generally, this algorithm produces good medial lines for some patterns but noisy ones for others as can be seen from the Figures 3.19 and 3.20. Also it is found that the medial lines follow closely the variations along the boundary of the input pattern. Further in certain cases as can be seen from Figure 3.21, double centre lines are generated. This algorithm sometimes introduces connectivity



(a): 66-Pixel Window

Original Pattern not filled

```
    ---
o-  o-o-oo-
o--o--o  --
--o-- -o-
 ---   --
```

Expanded pattern not filled

```
           - - -

- -      - - o - o - -
 o        o          o
- - - - - - - - -    - -
 o     o     o
- - o - -    - - -
                o
   - - -       - -
```

(b)

Original pattern filled

```
   ---
o-  oooooo-
o--o--o- --
-oo-- -o-
 ---   --
```

Expanded pattern filled

```
           - - -

- -      - ooooooo - -
 o        o  o     ooo
-o- - - o - -o-    - -
 o     o     o
- ooooo -    - o -
              o
             - -
```

(c)

Figure 3.18: Illustration of the Filling Algorithm

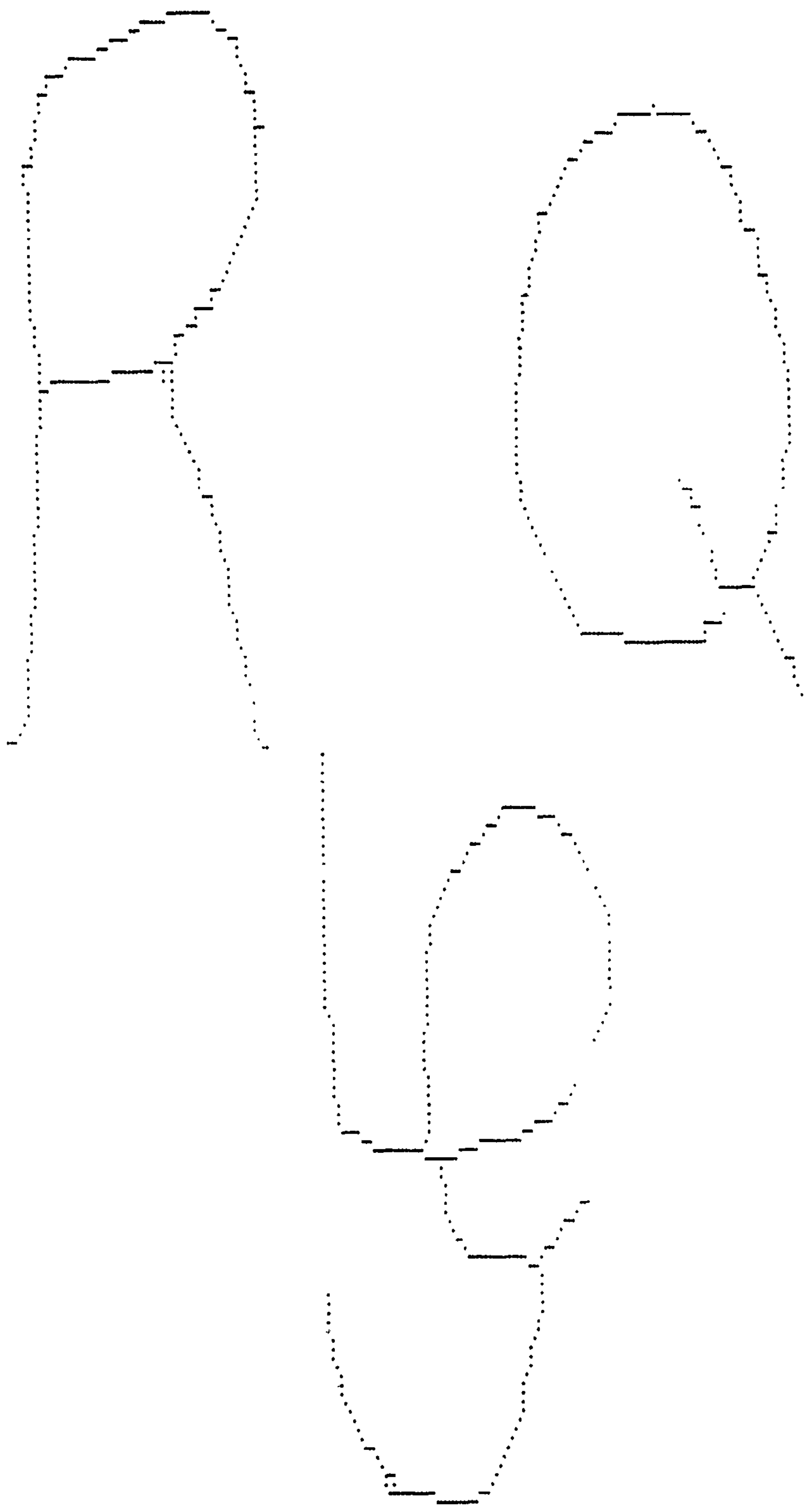


Figure 3.19: Skeletons Generated by MSM Algorithm

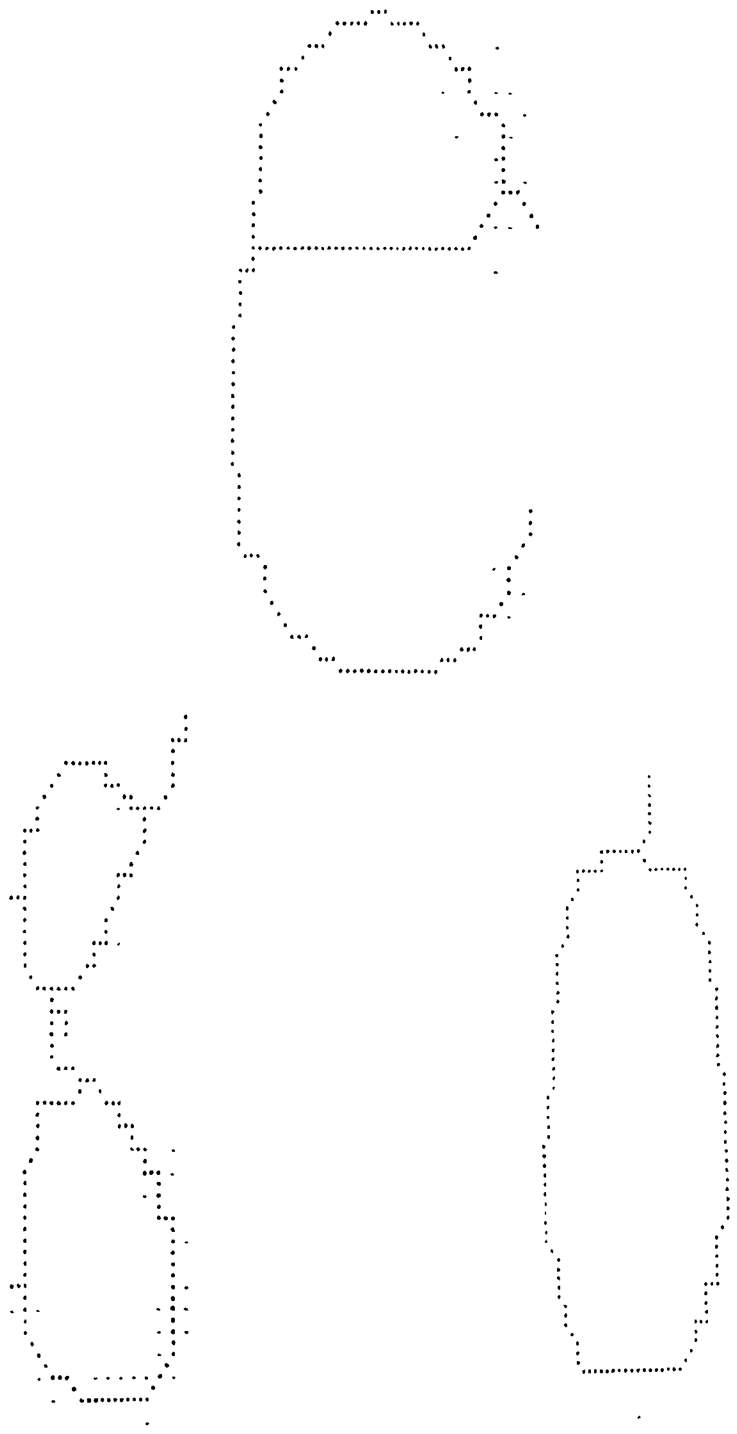


Figure 3.20: Examples of Noisy Skeletons Generated by MSM

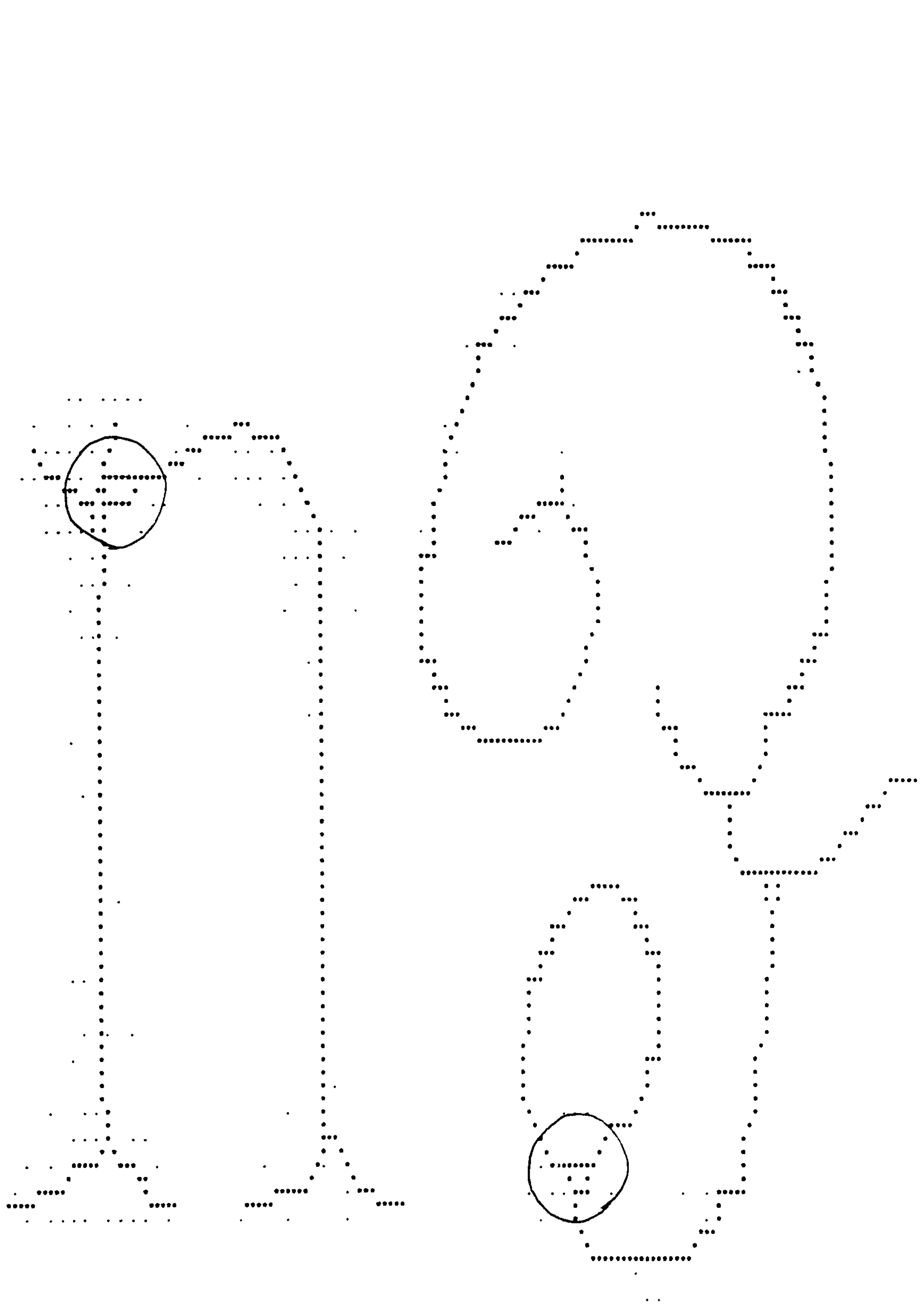


Figure 3.21: Double Centre Lines Generated by MSM

between two disjoint strokes or parts of a pattern as can be seen from the Figure 3.22. This is due to the filling algorithm which does not take into consideration the adjacency relation between groups of the same row or between rows. When patterns have straight edges (horizontal or vertical) as in Figure 3.23, the MSM algorithm produces better results. As the thickness of the character is reduced, the processing time also decreases since the time required for the enlargement phase of the algorithm is less. The processing time is found to increase with an increase in the number of branches in the input pattern, because each branch is processed independently.

The algorithm has the ability of reconstructing the original pattern from the core-line information. The core-line information generated by the algorithm consists of the coordinates of maximal squares defined along with the size of the squares. From this, the original pattern can be reconstructed by initially starting with a matrix of white pixels and then adding dark pixels that make up each maximal square. The reconstructed patterns for the characters 'e' and 'n' are shown in Figures 3.24 and 3.25 respectively. The reconstructed pattern for the character 'e' is better than that of the other algorithms at the rounded corner and also the number of mismatch points is in general less than that of the CPM, SPTA and DT algorithms.

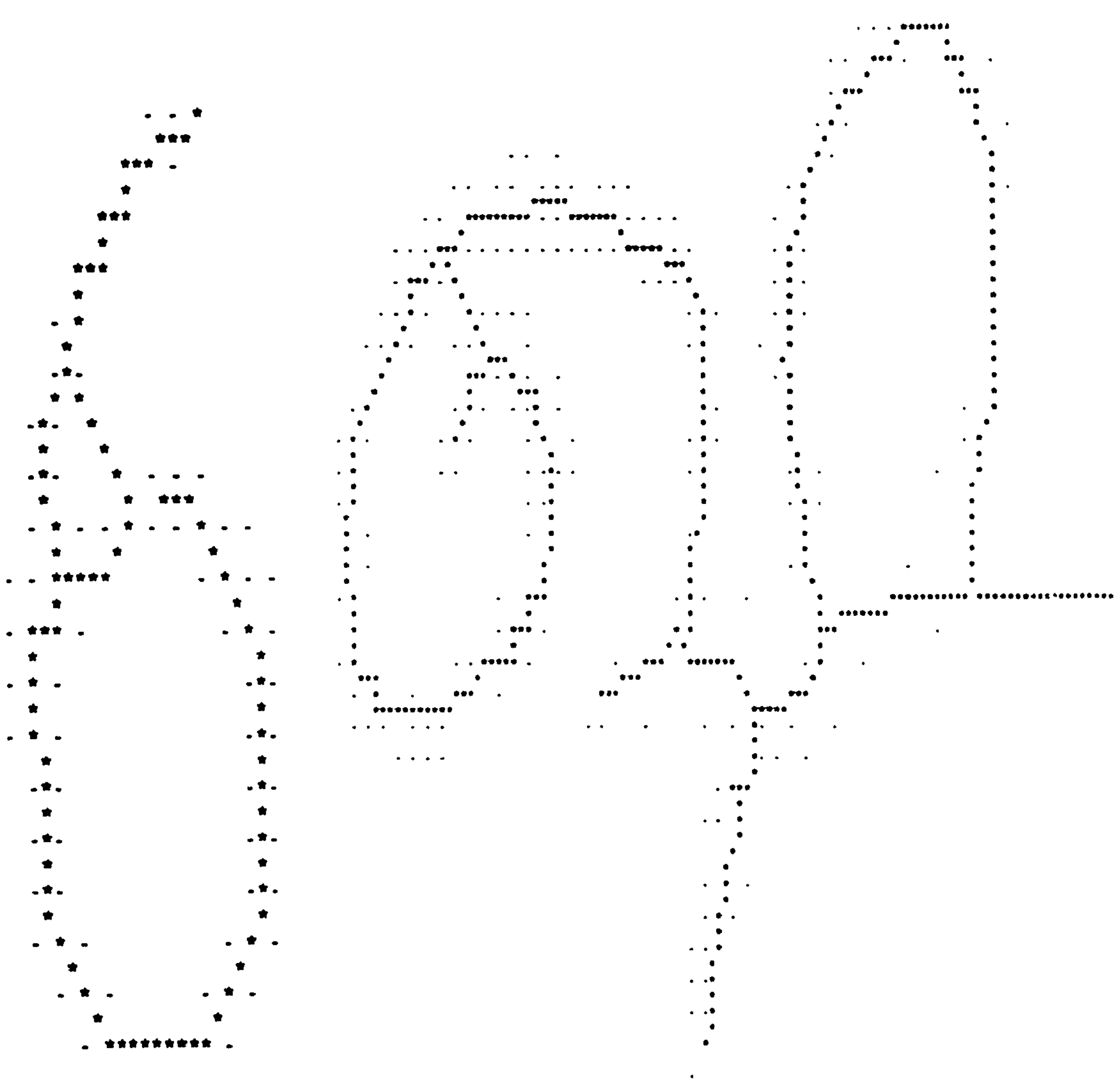


Figure 3.22: Connectivity Introduced for Disjoint Parts

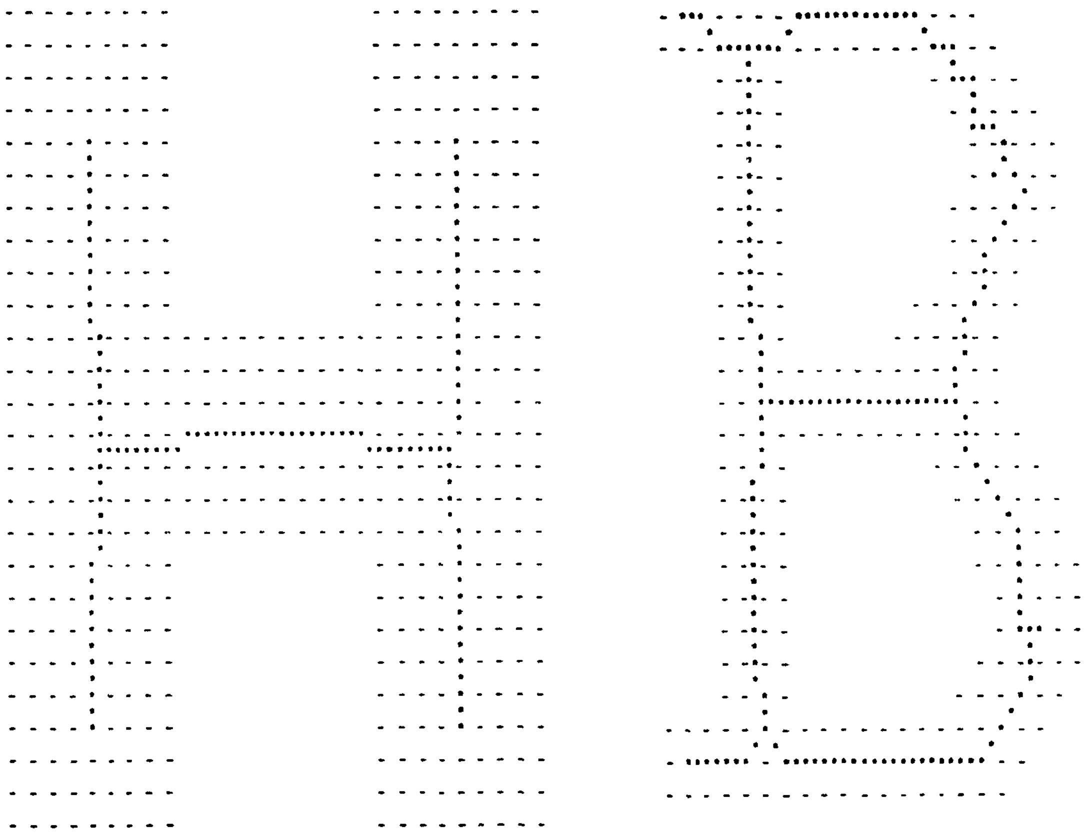


Figure 3.23: Skeletons by MSM for Straight Line Patterns

```
            111-
          11111111111
        111111111111111
      111111111111111111
      11111-     -1111111--
    11111-        11111111-
    1111-         --1111111-
  11111-           -1111111-
  11111-            1111111-
  11111-            11111111
 111111-            11111111
 111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
1111111-
111111-
111111-
111111-
1111111
1111111
1111111-
1111111-
11111111
111111111                 111
-11111111-               111
-111111111-             1111
 -111111111             111-
 -11111111111         11111
 -111111111111111111111111-
  -1111111111111111111111-
   -11111111111111111111-
    --1111111111111111-
      --111111111111--
        --11111111--
```

-: represents mismatch points between the original pattern and the reconstructed pattern

Figure 3.24: Reconstructed Pattern (MSM)

```
        1111-     111111111
      1111111   111111111111
  111111111111111111111111111
  11111111111111111111111111111
  11111111111111111111111111111
   -111111111111-   --111111111
   -1111111111--      --1111111
    --1111111-         -1111111
     -1111111-          1111111
     -111111-           1111111
      111111-           1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111            1111111
      111111-           1111111
      111111-           1111111
    11111111-           1111111
      111111111         11111111
  111111111111        11111111111
1111111111111111    111111111111111
1111111111111111    111111111111111
```

-: represents the mismatch points between the original and
   the reconstructed pattern

Figure 3.25: Reconstructed Pattern (MSM)

3.4. Tracing Centre-Lines of Digital Patterns using Maximal Circle Moving Algorithm (MCM)

3.4.1. Description of the Algorithm

This algorithm is a modified version of the MSM algorithm described in section 3.3, in that Circles instead of the Squares, are moved across the pattern, the centres of which form the skeleton of the pattern. The use of circles instead of squares allows curved shapes to be included.

3.4.2. Implementation of the Algorithm [35]

This methodology follows the steps 1 through 4 given in section 3.3.2 of the MSM algorithm except for the following changes in rule 2 and the maximal circle condition needed in step 4.

Rule 2:

$$\{ (B_{J+1}{}^{m} \leq P_J{}^{n} - C) \quad \text{and} \quad (E_{J+L}{}^{m} \geq E_J{}^{n} + C) \}$$

where

C : fixed value for each group in the circle.

J : current row.

L : current circle size.

m,n: group indices in rows J and J+L.

P : B, B+1, B+2, ... , E-1, E.

For the test patterns under consideration, the largest circle size is found to be 15 pixels wide at the centre and the constant 'c' in the above rule is predetermined and given in Table 3.2 [35].

Maximal Circle Condition :

For all groups passing through the maximal circle $\{ (E_{R+i} \le E_C) \}$ is true

where

R : is first group of the circle.

i : 0, 1, ..., L (circle size).

C : group passing through circle centre.

3.4.3. Analysis of the Algorithm

In general it is found that the processing time for the Maximal Circle algorithm is greater than that of the Maximal Square algorithm. This is due to the fact that additional time is taken by the MCM algorithm for the table look up to determine the number of pixels in the various groups of a circle as it is being enlarged. However, this algorithm generates smoother medial lines than that of the MSM algorithm. As can be seen from the Figure 3.26, the noisy branches in Figure 3.20 are eliminated. In the case of patterns with straight edges (comparing Figures 3.21 and 3.27) the MSM algorithm gives better results than the MCM algorithm. Figure 3.28 shows the medial lines generated by the MCM algorithm for the characters 'n' and '[illegible]'. It is found that in these test patterns, the MCM algorithm does not generate

Table 3.2: Maximal Circle Sizes [35]

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

This table shows the maximal circle sizes. The second column specifies the number of allowed white pixels in each group needed for the MCM algorithm. The third column specifies the number of pixels in the first group and the rest of the columns specify the constant 'C' in rule (3).

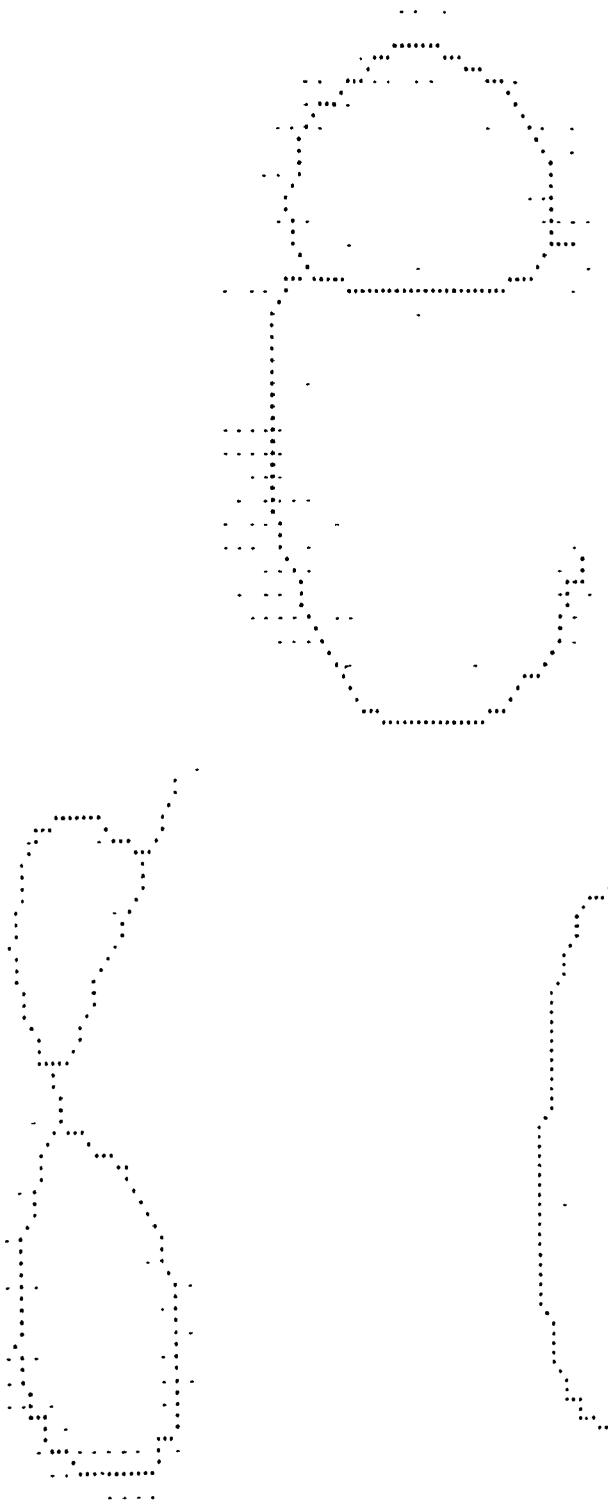


Figure 3.26: Skeletons Generated by MCM

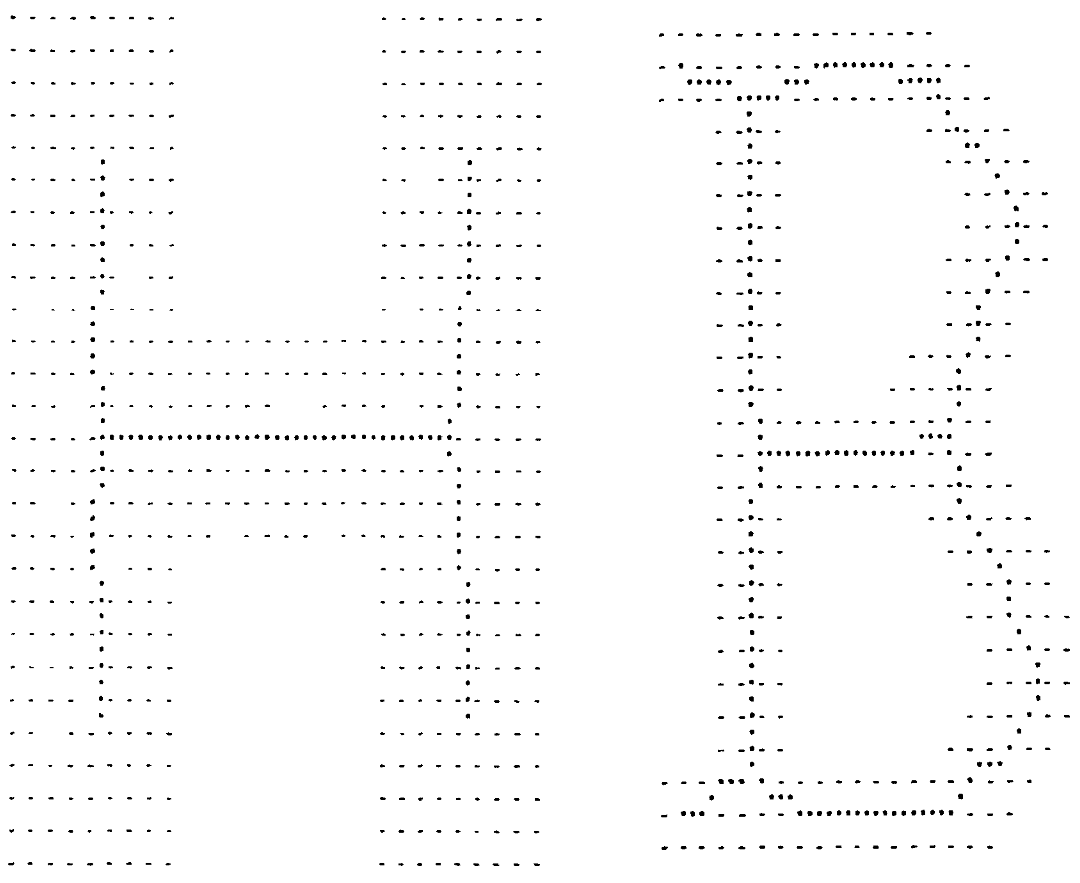


Figure 3.27: Skeletons Generated by MCM for Straight Line Patterns

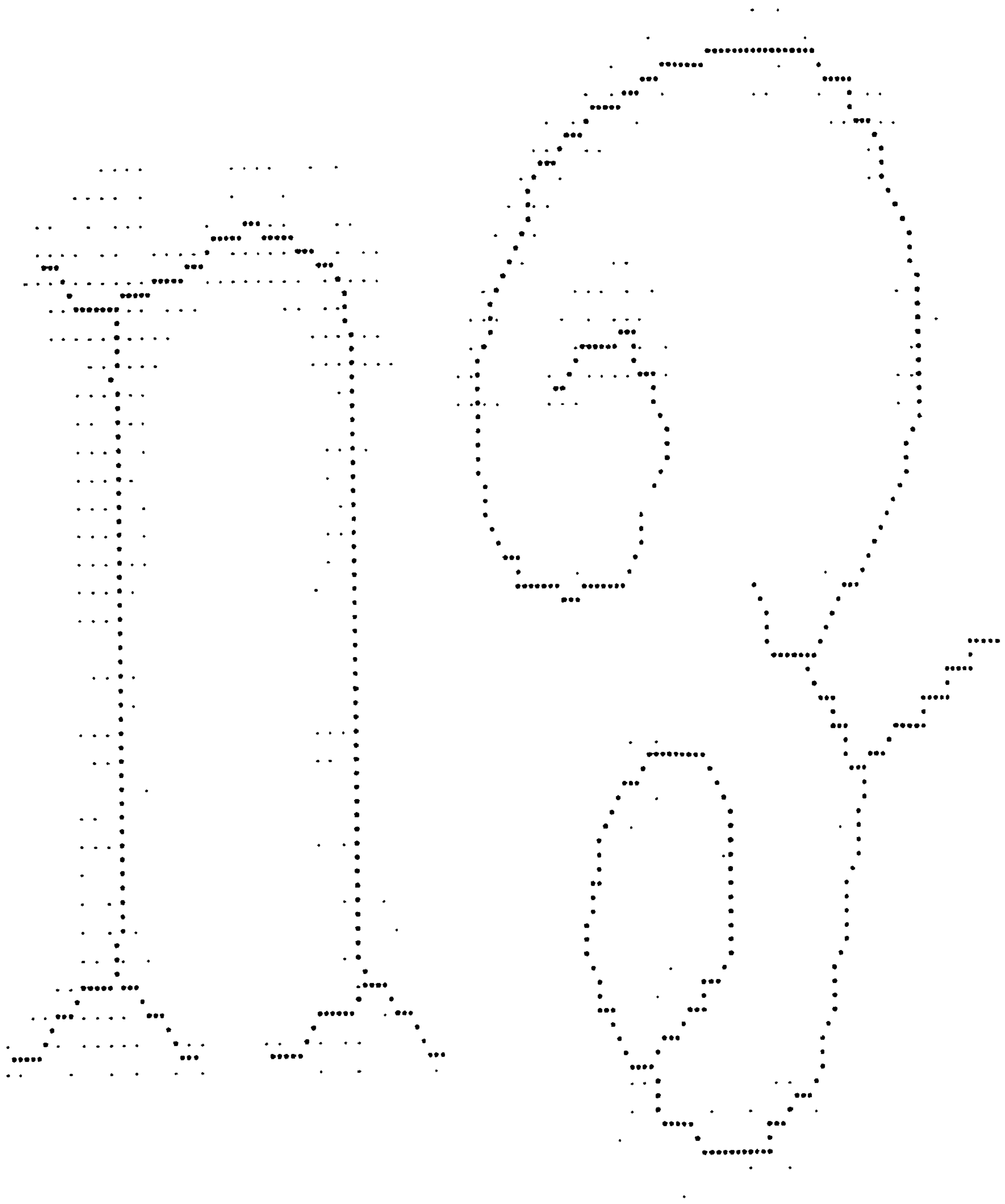


Figure 3.28: Skeletons Generated by MCM

double centre lines in contrast to the MSM algorithm. Unlike the MSM, the algorithm does not introduce connectivity to disjoint parts of the original pattern as can be seen from Figure 3.29. The MCM algorithm also has the reconstructability feature. This is achieved in the same manner as that of the MSM algorithm except that the pixels comprising maximal circles are determined from Table 3.2.

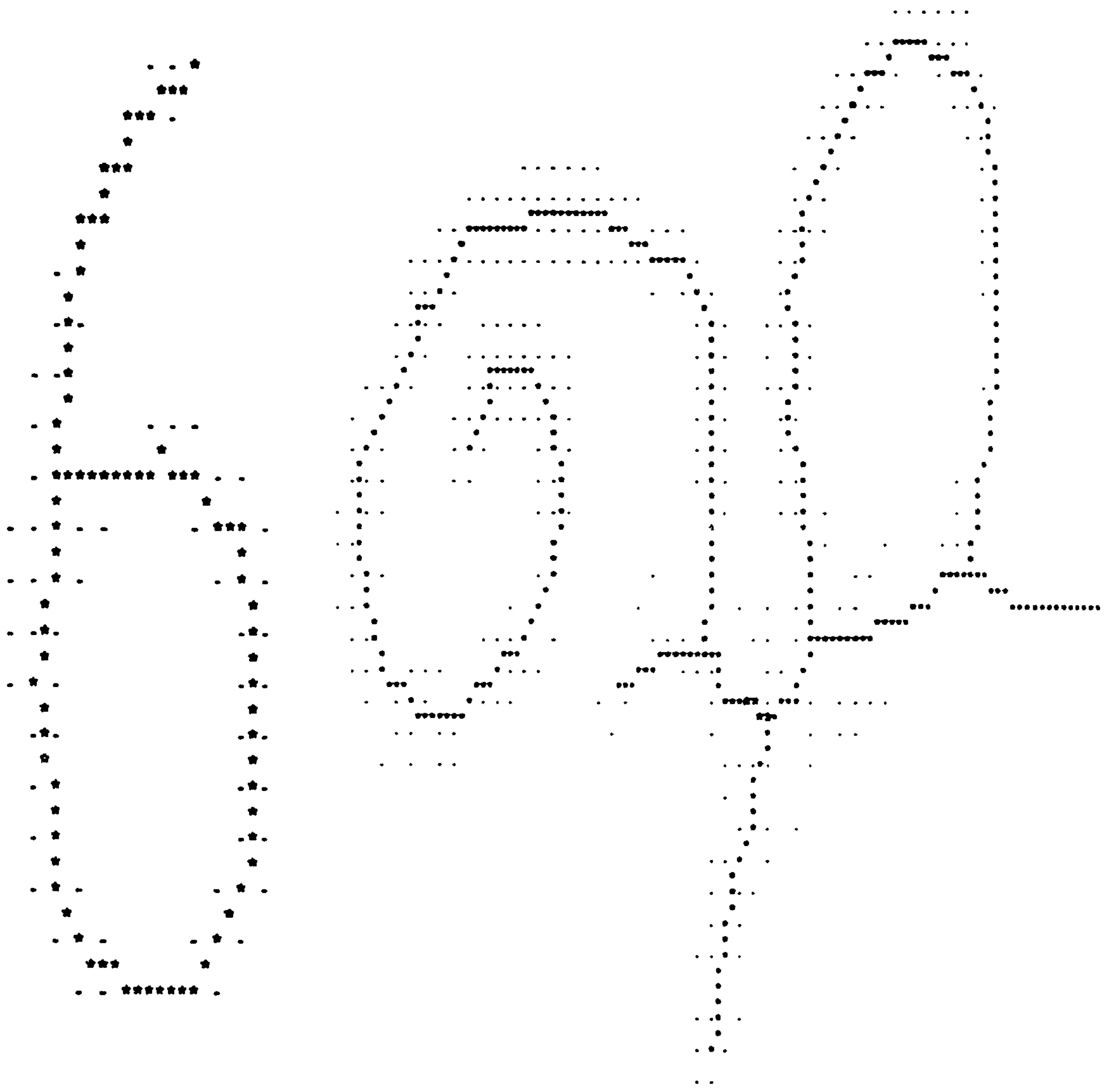


Figure 3.29: Skeletons Generated by MCM

4. EXPERIMENTAL COMPARISON OF THE ALGORITHMS

In this chapter we compare the performance of the newly proposed Charge Particle Methodology with four other algorithms, namely, the Sequential Thinning of Binary Patterns using Distance Transformation, Safe Point Thinning Algorithm, Maximal Square Moving and Maximal Circle Moving Methodologies. Each of these algorithms is described in detail in chapter 3. Here, first we propose a standard set of data to establish consistency in the evaluation of the algorithms. Using this data set the selected thinning algorithms are experimentally compared with respect to the following factors:

a. Need for preprocessing.

b. Quality of the skeletons generated.

c. CPU time required for the processing.

d. Reconstructability of the original pattern from the skeleton.

4.1. Data Set Selection

In order to achieve consistent evaluation of the selected thinning algorithms, a new bench mark input data set is introduced in this thesis. The characters and binary patterns in the data set, listed in Table 4.2 and given in Appendix, have been selected in such a way that the following characteristics occur at least 5 to 6 times in the entire set:

1. Multiple branches.
2. Characters with single or multiple islands.
3. Characters with uneven line thickness (as in character 'e', 'ஓ' and the moving man).
4. Thin characters or output from low resolution scanner.
5. Composite characters (ச், ட், ணை, ஞீ).

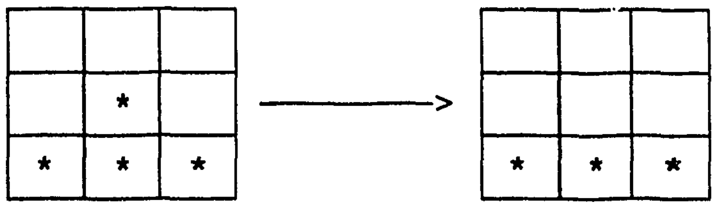
Multiple branch patterns are included to study the behaviour of thinning algorithms at the junction of branches. It is of importance as it provides valuable information such as the location and the number of branches at the junction points needed for the recognition stage. In addition the effect of multiple branches on the processing time can also be studied. The necessity for including thin characters is to test the performance of the algorithm in order to preserve the end points and connectivity of existing thin lines. Characters with multiple islands and composite characters are primarily included to establish the processing time requirement of each of the algorithms. Further they are also used to study the ability of algorithms to provide connected skeletons without merging adjacent islands.

The data set consisting of the various input patterns that satisfy the above mentioned requirements are compiled and used here for evaluation purpose. It is designed to represent features of characters from different languages such as Tamil and Chinese language characters consisting of a number of islands, arches and composite strokes. As such, in addition to a number of characters and other binary patterns from various papers published so far, a set of newly scanned characters from Tamil(Indian), Chinese and English languages are also included in the data set.

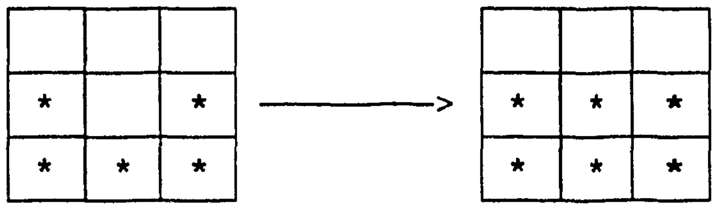
This data set and the corresponding skeletons generated by the different thinning algorithms are given in the Appendix. It is hoped that this bench mark data set will also be useful to other researchers working in the field of thinning and skeletonisation.

4.2. Performance Evaluation of the Algorithms

Preprocessing: From the discussions in chapters 2 and 3, it is clear that all the five algorithms need preprocessing to eliminate the effects of salt and pepper noise. Thus all the input patterns are initially preprocessed to remove the salt and pepper noise using the same smoothing process [17]. This smoothing process consists of eliminating the isolated dark pixels as well as the small bumps in the pattern and filling in holes in the input pattern. The input binary pattern is scanned from top to bottom and from left to right. Any dark pixel P (Figure 1.2) is set to white if the Boolean Expression given in equation (4.1) is true and P is not a break point



a) Removing Pepper Noise



b) Filling Holes

Figure 4.1: Smoothing Process

(Figure 4.1a). The white pixels are set to dark if the number of dark pixels in the 4-neighbour is at least 3 (Figure 4.1b).

$$B = (P1+P2+P3)(P5+P6+P7) + (P3+P4+P5)(P7+P8+P1) \quad (4.1)$$

<u>Quality of the Skeletons Generated</u>: The following features are generally used for judging the quality of the skeletons [7,38]:

1. Connectivity
2. Thinness and symmetry of the skeletons

3. End-point preservation

4. Presence of noisy branches

5. Sensitivity to orientation angle of the pattern

6. Visual Quality

Connectivity: This is the most important characteristic of a skeleton pertaining to the preservation of connectedness in the pattern. The skeletons can have either 4 connectedness or 8 connectedness. If we have 4 connectedness, the skeletons will have redundant pixels at the junctions and also stepped skeletons will be generated. The 8 connectivity gives a smoother skeleton and hence a better visual quality. In general we find that the CPM, DT and SPTA generate skeletons that are smooth and have perfect 8 connectedness. But the skeletons generated by MSM and MCM algorithms have imperfect 8 connectedness in that there are some redundant pixels present at the junctions of branches. In some cases, as in Figure 4.2, MSM introduces connectivity in the skeletons generated. This is introduced by the filling algorithm and is rather undesirable as the original shape of the pattern is lost.

Thinness and symmetry of the skeletons: The primary aim of all thinning algorithms is to generate thin skeletons of unit thickness. The degree to which this objective is satisfied by different algorithms is referred to as 'Thinness'. Further,

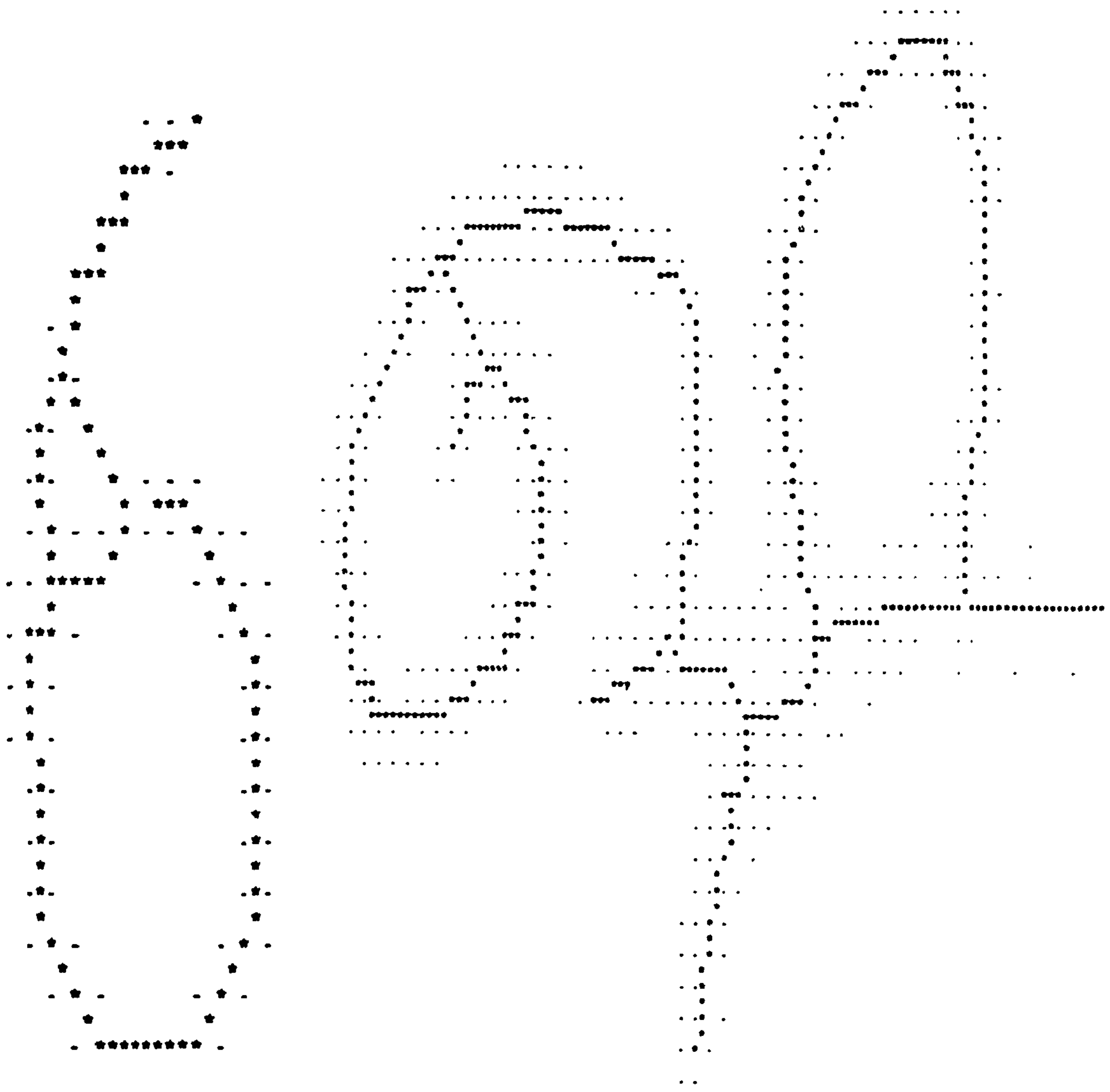


Figure 4.2: Connectivity Introduced for Disjoint Parts (MSM)

the generated skeleton should be symmetric or 'isotropic'. That is, the original pattern should be peeled off symmetrically, so that the resulting skeleton lies along the medial axis of the pattern and is not biased. All the algorithms are found to generate almost medial line skeletons but all of them fail in the case of the character 'e' (Figure 4.3) where the skeletal line is found to lean inwards at the junction of the varying width strokes, ie. more points are removed from the outer corner than the inner corner. In the case of the character ' ' the junctions of the varying width strokes are smoother and hence the skeletal lines are not biased (Figure 4.4). Furthermore the skeletons generated by MSM and MCM have redundant pixels.

<u>End-Point Erosion</u>: The end-point preservation is an important aspect relating to the shape retention by the thinning algorithms. The end point erosion is found to be the minimum in the case of SPTA which introduces a maximum shrinkage of 2 points at the ends. In the case of CPM, the end point erosion is found in patterns where the strokes are slanting and more than 3 pixels wide as illustrated in Figure 4.5. Nevertheless it does not deteriorate the topology of the original pattern. All the shape information is still retained. The MSM and MCM algorithms also introduce end point erosion when there are thick patches in the pattern as in Figures 4.6 and 4.7. The DT algorithm has better end point preservation than the MSM,

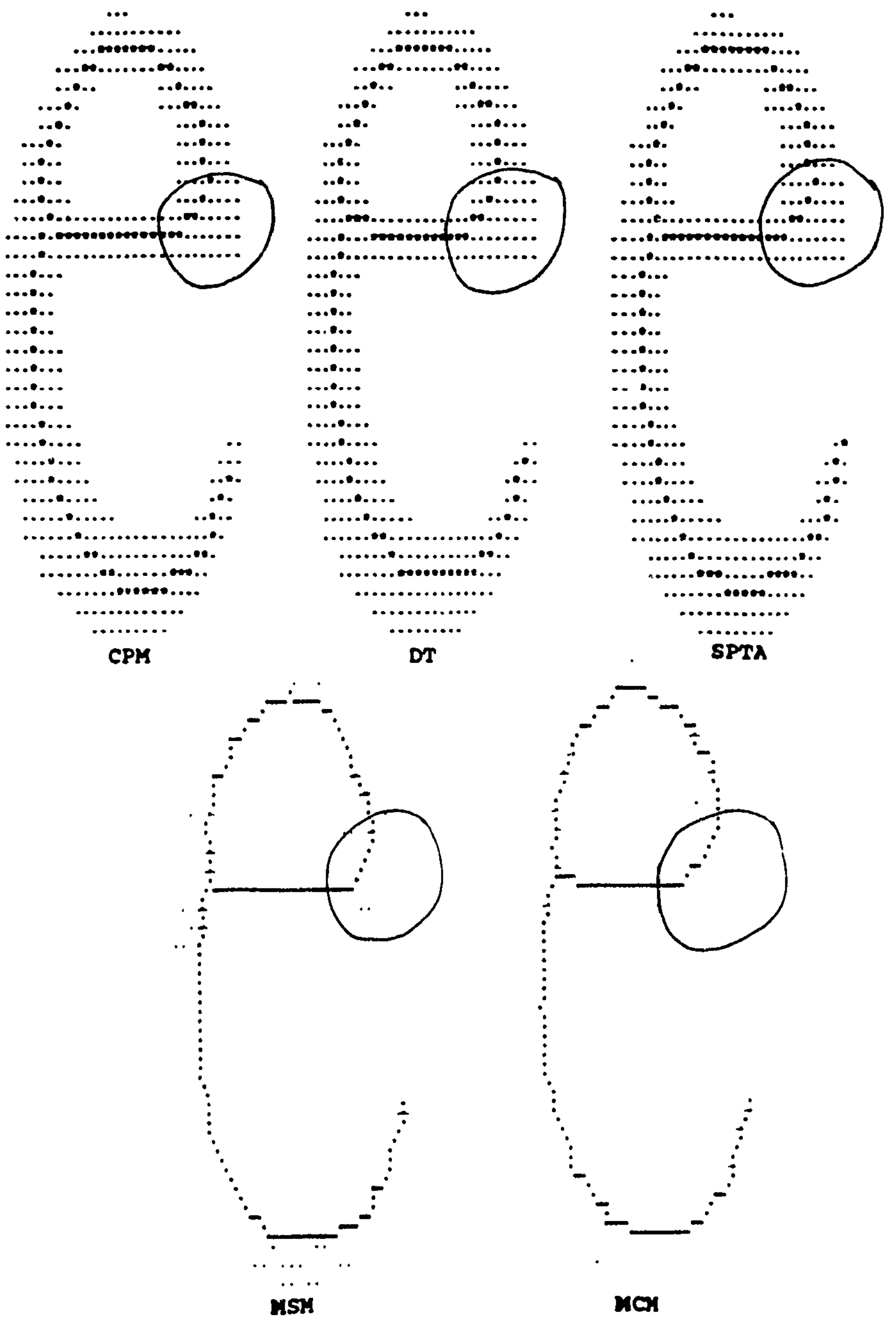


Figure 4.3: Effect at the Junction of Varying Width Strokes

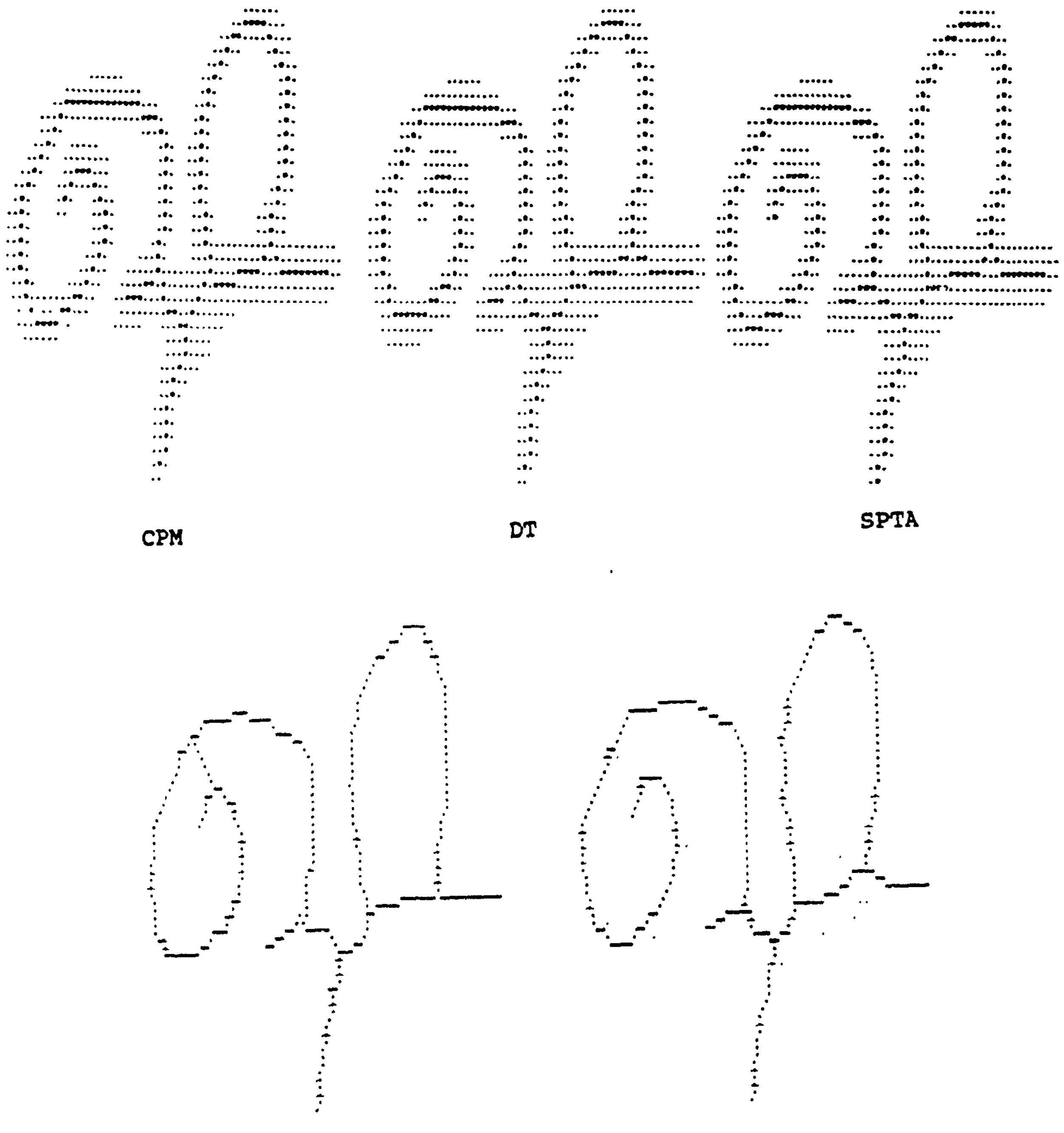


Figure 4.4: Effect at the Junction of Varying Width Strokes

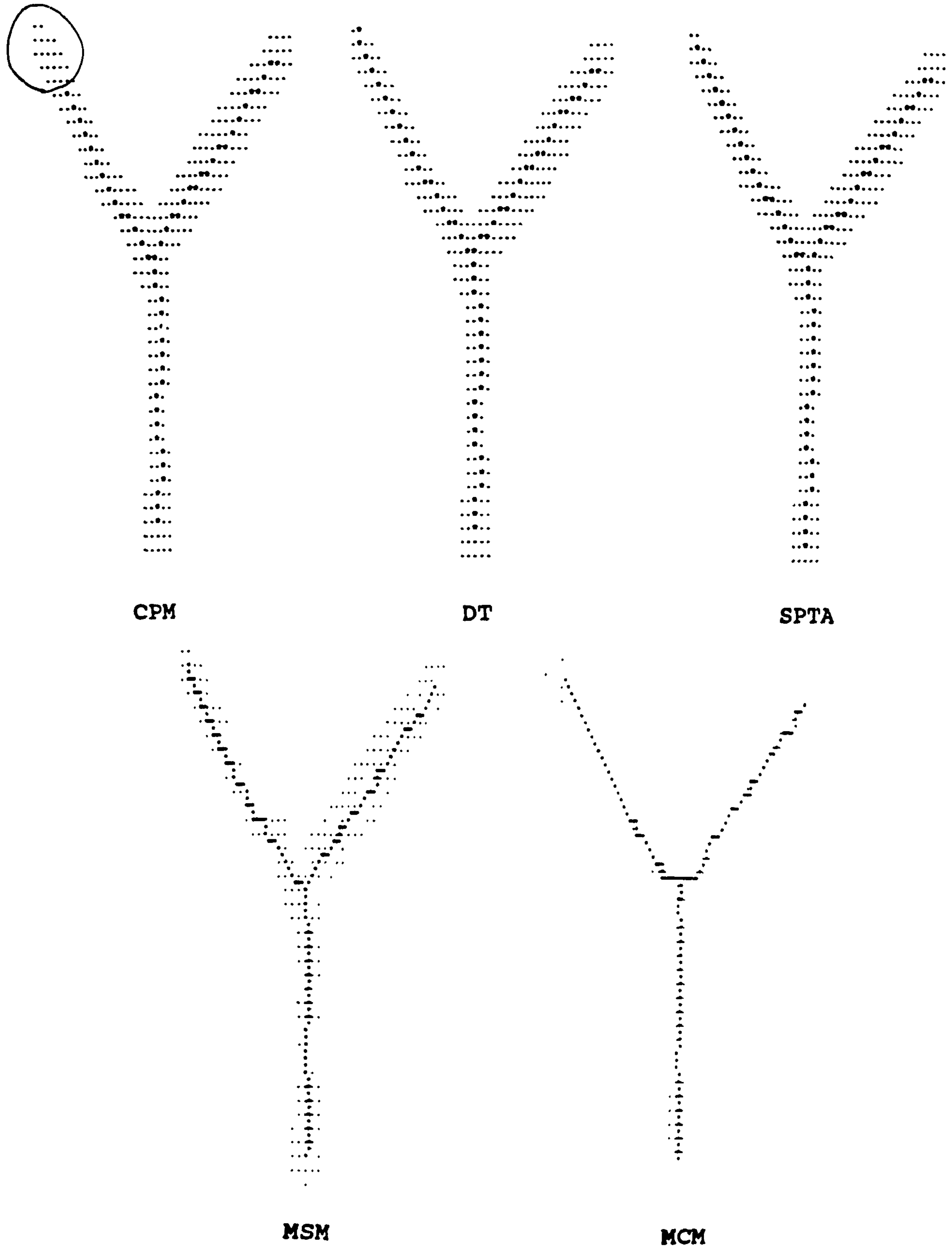
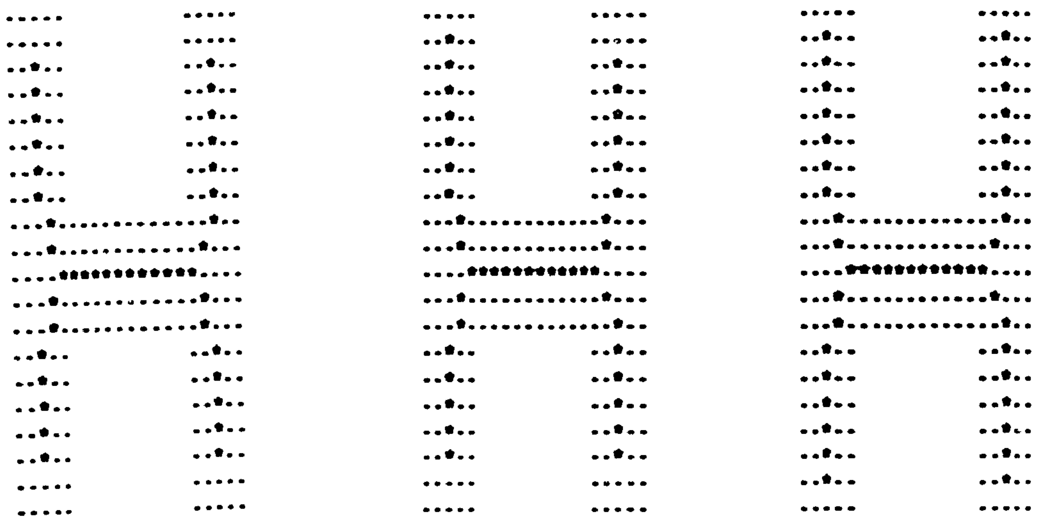


Figure 4.5: Examples of End-point Erosion


CPM DT SPTA

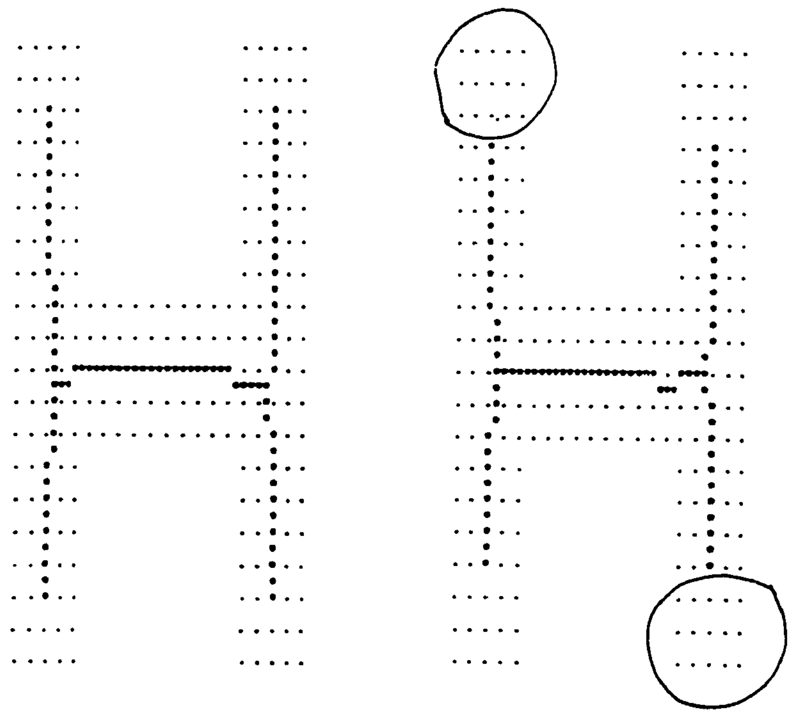
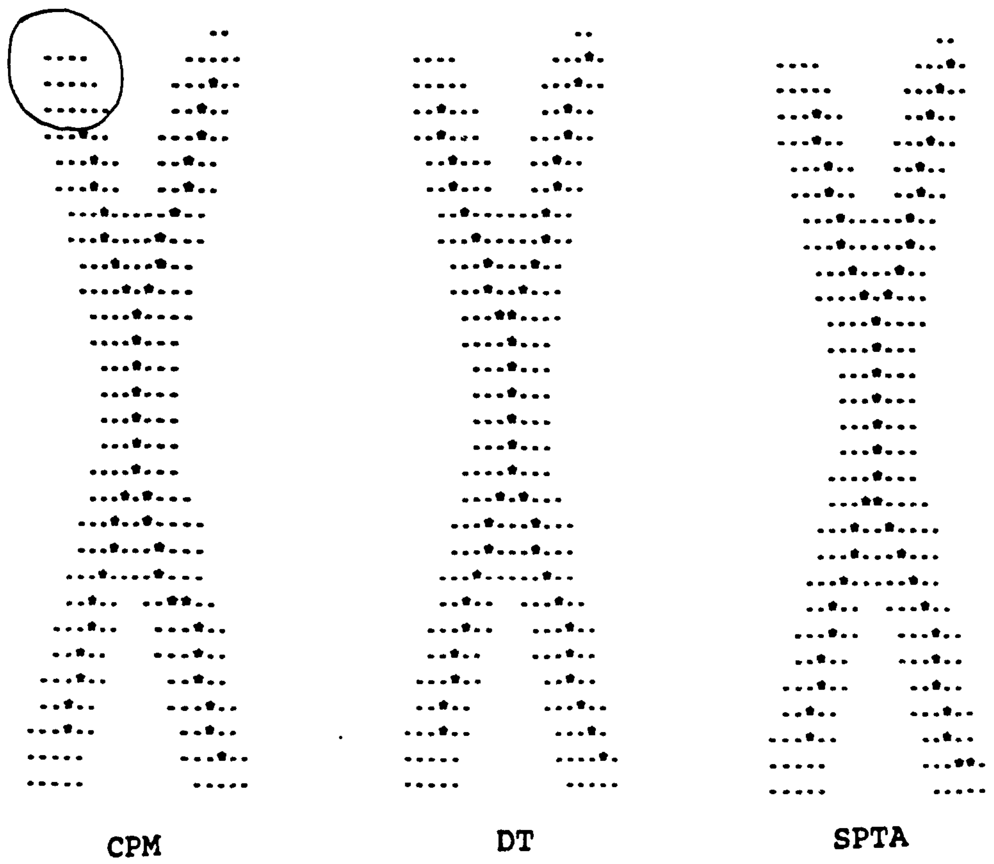

MSM MCM

Figure 4.6: Examples of End-point Erosion



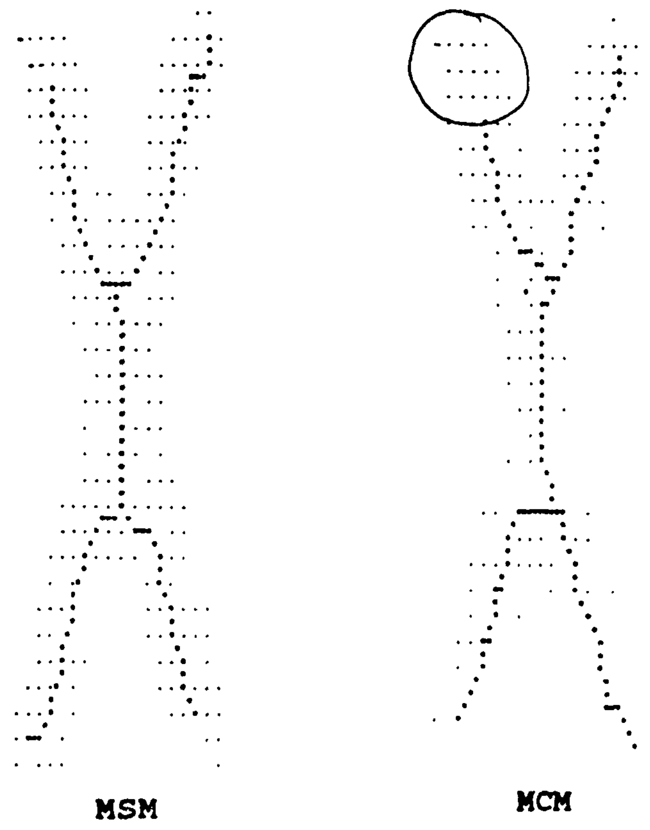


Figure 4.7: Examples of End-point Erosion

MCM and the CPM and it introduces a maximum shrinkage of 3 to 4 points at the ends.

Noisy Branches: The skeletons generated by CPM, DT and SPTA algorithms in general do not contain noisy branches. The MSM algorithm generates noisy branches (as in Figure 4.8) and in certain cases (Figure 4.9) introduces double centre lines. The double centre lines and the noisy branches are eliminated in the case of MCM algorithm as the circles cover more shapes than the squares. The changes along the contour of the input patterns also tend to result in noisy branches (as in Figure 4.10) and jaggedness of the skeleton in all the algorithms considered.

Sensitivity to Orientation Angle of the Pattern: This is another desired characteristic of any thinning algorithm. This requires that the skeletons generated for the various angles of rotation of the input pattern should preserve the geometry or the shape of the pattern. The CPM, DT and SPTA algorithms preserve the geometry of the pattern under varying angles of rotation even though there is no pixel to pixel match in the skeletons generated. The MSM and MCM algorithms also generate similar skeletons when the pattern is rotated about 90 degrees. But in the case of other angles of rotation (45, 30 degrees etc.,), the skeletons are not the same as can be seen from the Figure 4.11.

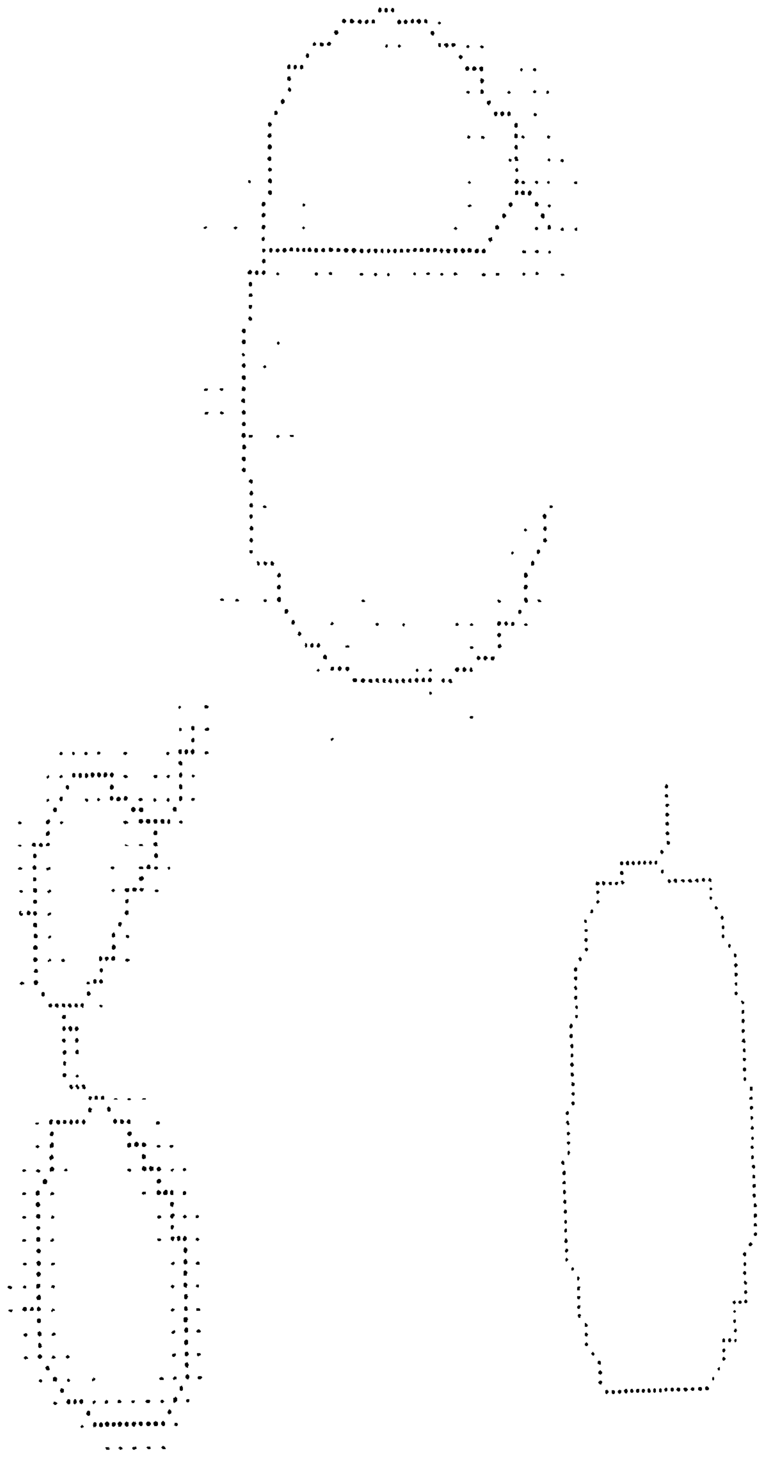


Figure 4.8: Noisy Skeletons Generated by MSM

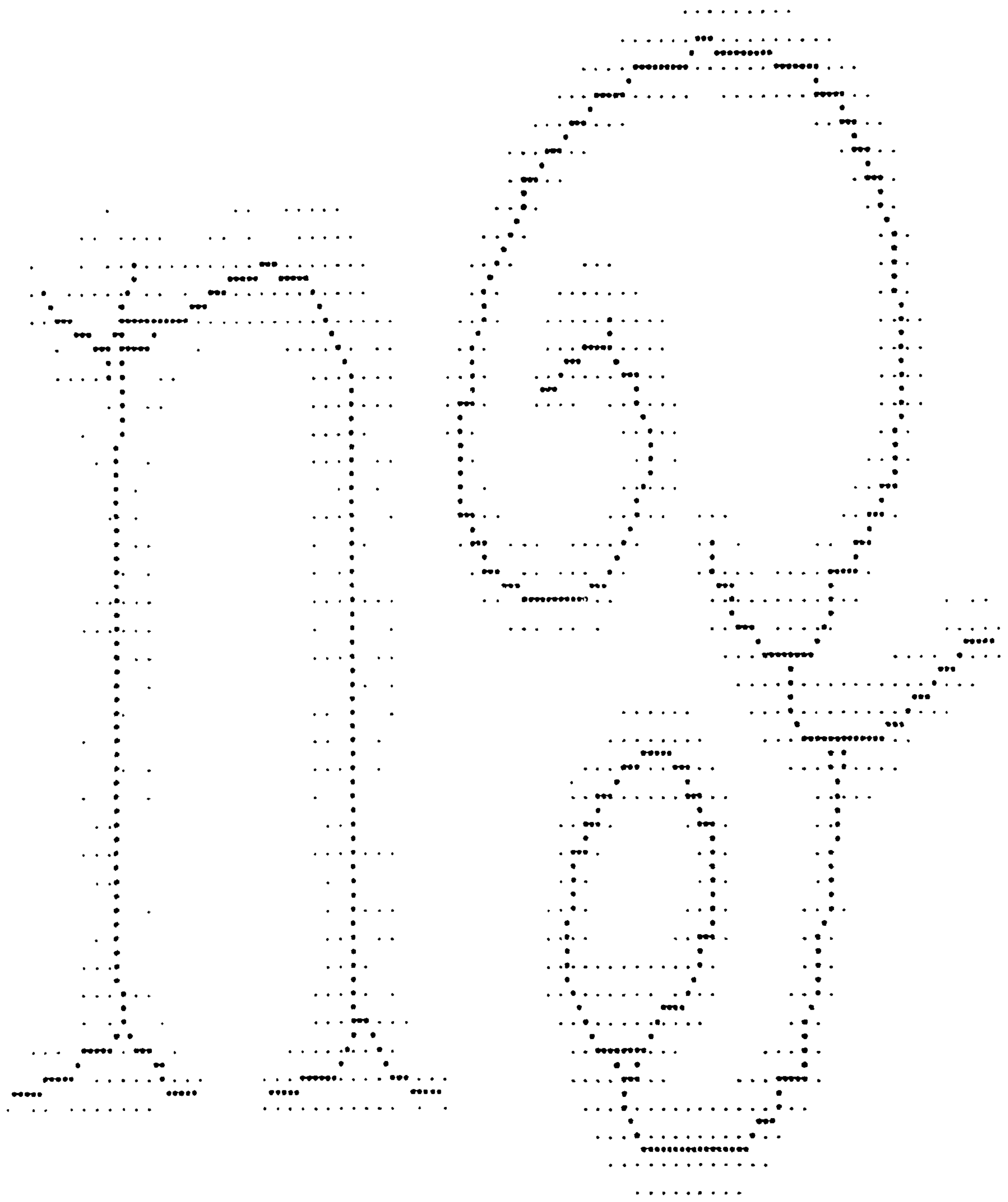


Figure 4.9: Double Centre Lines Generated by MSM

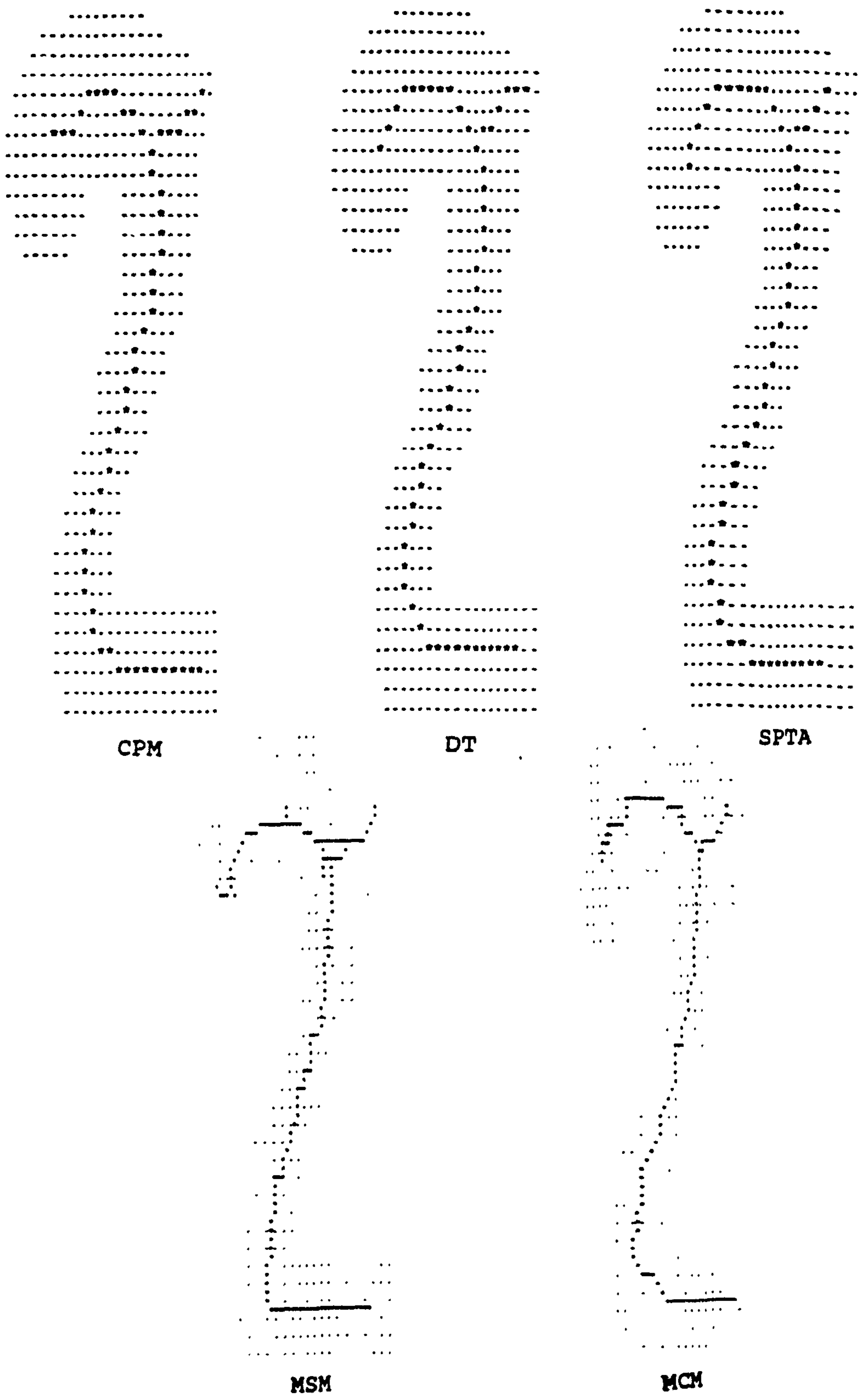
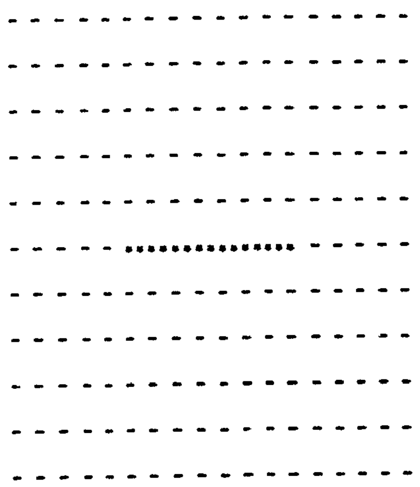


Figure 4.10: Effect of Boundary Noise


a. Skeleton generated by both MSM and MCM for a rectangle

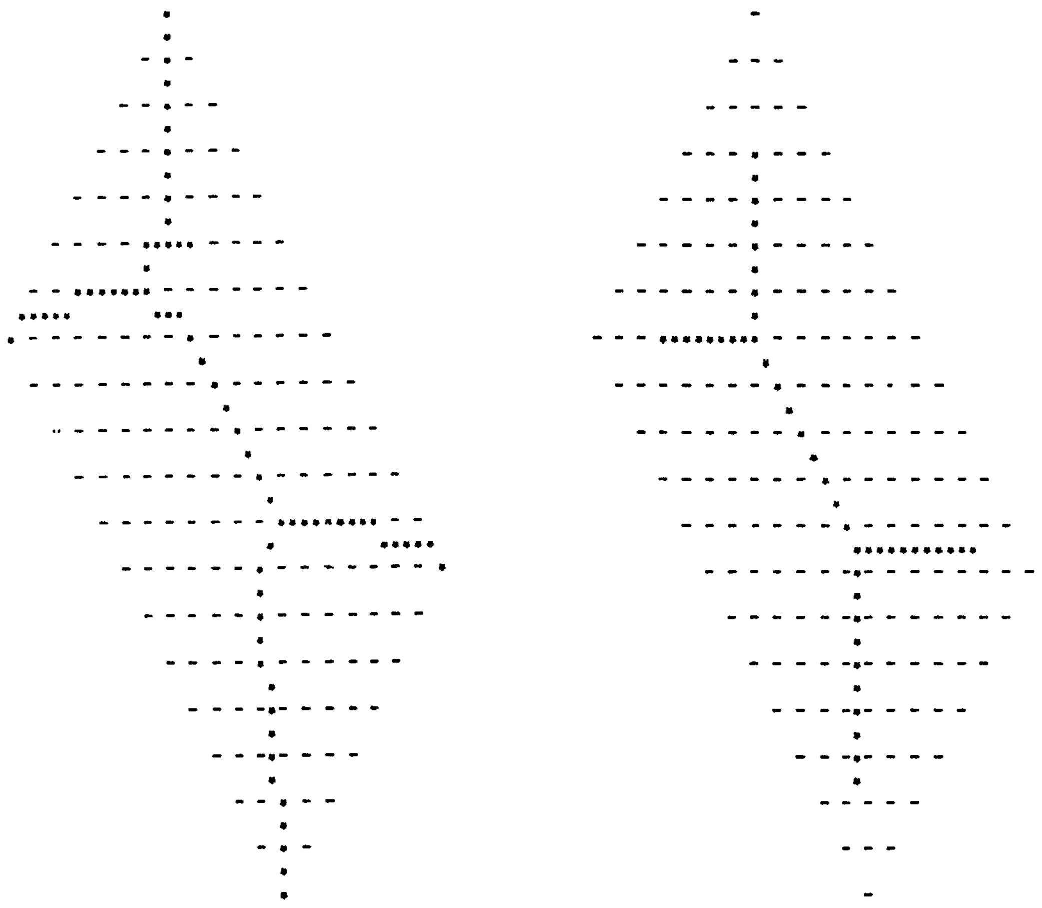

b. Skeletons generated for rectangle of (a) under rotation

Figure 4.11: Skeletons Generated by MSM and MCM for Rectangle

Visual Quality: This criterion is a subjective evaluation and would normally introduce a bias of the evaluator. As far as the visual quality is concerned the algorithms CPM, DT and SPTA provide better appearance. The other two algorithms viz., MSM and MCM are comparatively inferior.

With respect to quality of the skeletons generated, as summarised in Table 4.1, the algorithms can be ranked in a descending order as SPTA, DT, CPM, MCM and MSM, with the SPTA being at the top.

4.3. CPU Time Required for Processing

All the algorithms are written in Pascal language and tested on a CYBER CDC860 computer. The CPU time refers to the time required for processing each character. This is obtained as an average over 100 executions for each character processed at various times of the day. The CPU time required in seconds for all the algorithms is given in Table 4.2. The DT algorithm is found to be the fastest on the average, followed by the CPM, SPTA, MSM and MCM. The SPTA algorithm is much slower than the CPM and in most cases, but faster than the MSM and MCM algorithms.

The CPU time for rectangles with sizes varying from 40 x 10 down to 40 x 2 are measured and are given in Table 3.1. The corresponding Figure 4.12 shows the graph for CPU time versus

Table 4.1: Comparison

Characteristics	Algorithm				
	CPM	DT	SPTA	MSM	MCM
Connectivity	P-8	P-8	P-8	I-8	I-8
Skeleton Symmetry	Fair	Good	Good	Fair	Fair
Thinness of the skeleton	Unit Width	Unit Width	Unit Width	Multiple Pixels	
Presence of End-point Erosion	Yes	No	No	Yes	Yes
Presence of Noisy Branches	No	No	No	No	No
Sensitive to Orientation of the pattern	No	No	No	Yes	Yes
Visual Quality	Good	Good	Good	Fair	Fair

P-8: Perfect 8 connectedness

I-8: Imperfect 8 connectedness

Table 4.2: CPU Times in Seconds

CHARACTER	CPM	DT	SPTA	MSM	MCM
A	0.546	0.522	1.640	1.249	1.982
E	0.692	0.763	1.843	2.921	4.360
e	0.781	0.786	1.931	2.123	3.669
g	0.971	0.983	2.115	4.453	6.715
H	0.446	0.444	1.380	0.982	1.495
n	1.037	0.968	2.180	2.194	4.043
O	0.884	0.698	1.791	1.881	3.339
Q	1.091	0.963	2.407	3.230	6.188
R	1.081	0.987	2.385	3.353	6.027
u	0.915	0.632	1.740	1.489	3.047
Y	0.754	0.617	2.048	1.732	2.336
X	1.097	0.894	2.294	2.710	5.244
G	0.217	0.236	0.853	0.609	0.886
B	0.218	0.255	0.921	0.645	0.941
S	0.198	0.245	0.815	0.590	0.875
க	1.058	1.118	2.286	4.378	7.344
ச	0.662	0.675	1.74	2.083	3.315
ண	1.272	1.131	2.605	3.940	8.411
ழ	0.740	0.844	2.096	3.418	5.456
ஐ	1.708	1.249	2.850	7.240	11.88
ஷ	1.270	1.107	2.728	4.291	6.998
ஜ	1.114	0.966	2.609	5.585	8.358
ட	0.194	0.224	0.823	0.543	0.739
久	2.138	1.704	4.267	5.181	9.249
南	1.371	1.373	2.823	5.066	8.849
的	1.390	1.258	2.771	5.370	9.518
孝	0.882	0.819	2.178	3.285	4.550
2	0.929	0.713	2.216	2.083	3.207
4	0.534	0.513	1.271	1.142	1.695
5	0.551	0.571	1.736	1.711	2.708
6	0.240	0.260	1.216	0.776	1.073
8	0.234	0.262	0.856	0.876	1.237
Average	0.863	0.765	1.895	2.671	4.207

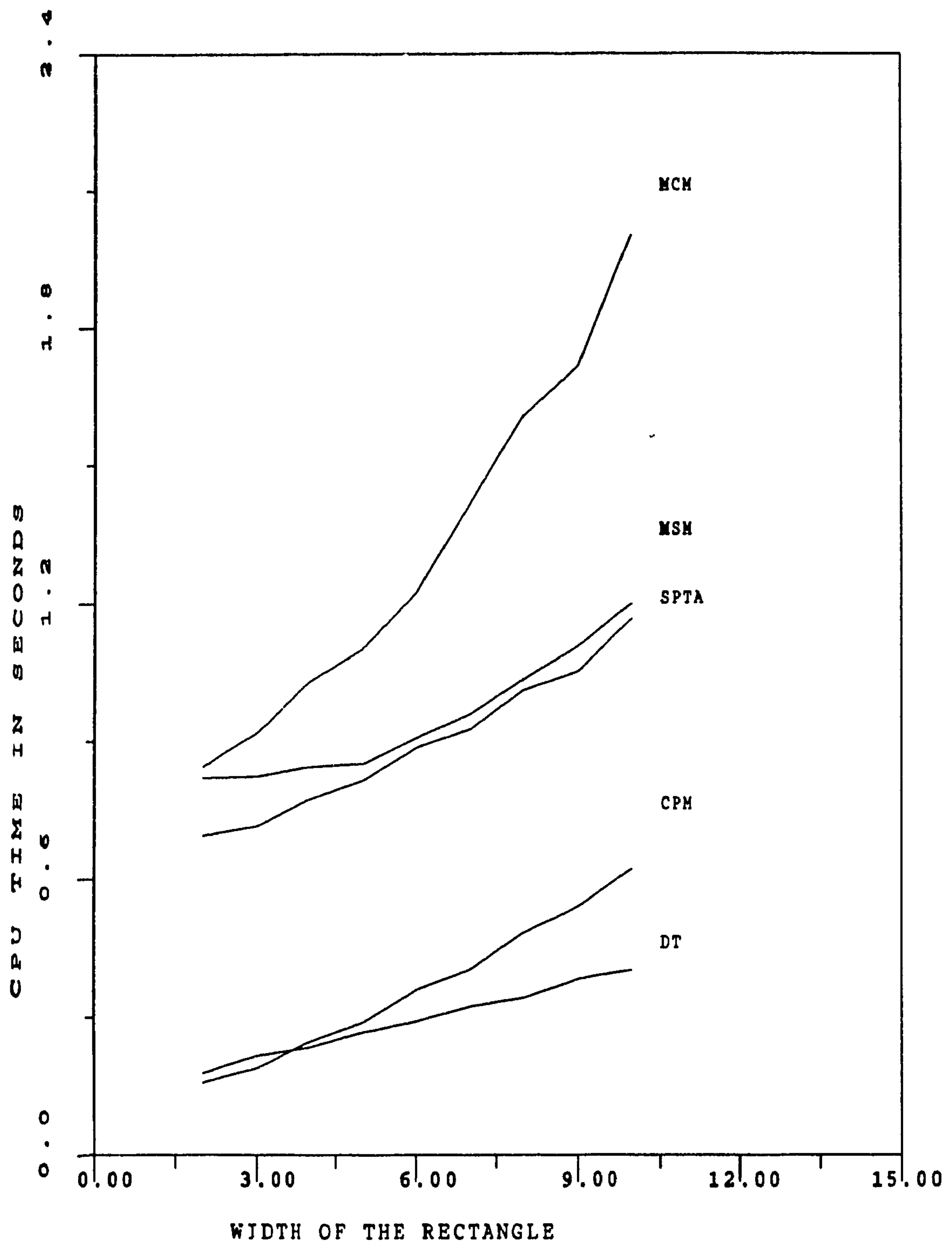


Figure 4.12: CPU Time Versus Rectangle Width

the width of the rectangle. It can be seen from the graph that the CPM and DT algorithms are much faster than the other three algorithms. The curve for the DT methodology has the smallest gradient and that of MCM has the maximum gradient. Hence if the input patterns have uneven thickness it is advantageous to use either the CPM or the DT methodologies. Although the CPM algorithm appears faster than the DT when the width of the patterns is less than or equal to 3 pixels. Table 4.3 shows the CPU time required by the CPM and the DT algorithms for thin characters (up to 3 pixel wide). It is found that the CPM algorithm is faster in general, except when the size of the object window becomes greater than 40 x 40. Hence it is preferable to use the CPM algorithm for low resolution scanner images.

The MSM and the SPTA have almost equal CPU timings for widths greater than 6. In general the large time requirements for the MSM and the MCM algorithms are due to the fact that list structures are used in the implementation and a lot of time is spent in the final filling stage needed to provide 8-connectivity to the core lines generated by the algorithm. The MCM algorithm requires more CPU time than the MSM, as additional time is required for the table look-up in deciding the size of the maximum circles.

Table 4.3: CPU Times in Seconds

CHARACTER	CPM	DT
a	0.490	0.490
b	0.521	0.549
c	0.401	0.401
d	0.602	0.593
g	0.971	0.983
B	0.218	0.255
G	0.217	0.236
S	0.198	0.245
5	0.551	0.571
6	0.240	0.260
8	0.234	0.262
க	0.710	0.767
ச	0.563	0.627
ட்	0.194	0.224
ழ	0.740	0.844
ஷ	0.897	0.879
南	1.371	1.373
Average	0.536	0.562

4.4. Reconstructability

Reconstructability is one of the desirable features expected of a thinning or skeletonising scheme. All the five algorithms discussed here have the reconstructability feature. The CPM, the DT and the SPTA methodologies make use of Pavlidis reconstruction algorithm [33]. However the labelling techniques used to identify the skeletal points are different in each case. The readily available distance transformation values of the skeletal points are used in the DT methodology, whereas the CPM and SPTA use the final iteration numbers associated with each of the skeletal points.

The MSM and MCM make use of the information in the core-line description viz., the location and the size of the maximal square or circle in reconstructing the original pattern. For each of the core-line point in the skeleton, all the white pixels in the square enclosing it are changed to dark pixels. Due to the fact that the original pattern is reconstructed in a single scan of the matrix, the time taken for reconstruction by these algorithms are comparatively less than the other three algorithms. Also the reconstructed patterns are better than the other three algorithms.

From the above discussions it is clear that the MSM and MCM algorithms are not suitable both in terms of quality and processing time. Among the other algorithms CPM and DT are the fastest and in general result in good skeletons. The CPM is preferable for low resolution scanner images than the DT methodology, as the processing time required by the CPM methodology is less than that of the DT methodology.

5. CONCLUSIONS AND SUMMARY

5.1. Conclusions

In this thesis a new thinning algorithm called Charge Particle Methodology (CPM) is presented. The CPM algorithm assumes raster scanning of the binary image. Hence a priori knowledge of the distinct contours present in the input pattern is not needed. The CPM algorithm results in a connected skeleton of unit width along the medial axis of the given binary pattern. The proposed CPM algorithm is tested using a well composed "Test Data set". The skeletons generated by this algorithm are found to preserve the connectivity well. They generally do not suffer from excessive erosion at end points except when there are slanting lines of three or more pixels wide. In addition, the topology of the object is also maintained by the skeletons. The CPM algorithm also has the ability of reconstructing the pattern with the least amount of mismatch between the original input pattern and the reconstructed pattern.

For the sake of evaluation of the CPM algorithm we have devised a "Test Data Set". This carefully composed data set can be used by other researchers to test the performance of thinning or skeletonising algorithms for connectivity and end-point preservation, behaviour at junctions of uneven thickness lines and processing time complexity for patterns with multiple branches, islands and composite characters. In addition to a number of characters and other binary patterns from various papers published so far, a set of newly scanned characters from Tamil(Indian), Chinese and English languages are also included in the data set.

The performance of the CPM algorithm is compared with four other recently published thinning algorithms, namely, the Sequential Thinning of Binary Patterns using Distance Transformation (DT), Safe Point Thinning Algorithm (SPTA), Maximal Square Moving (MSM) and Maximal Circle Moving (MCM) Methodologies. All the above mentioned algorithms are programmed in Pascal language and implemented on CYBER CDC800 machine. The results of the evaluation show that the skeletons generated by the CPM algorithm are good in quality and are comparable to those of SPTA and DT methodologies. The CPM algorithm is much faster than the three algorithms namely SPTA, MSM and the MCM methodologies and is found to be only slightly slower than the DT method (0.863 Vs 0.765 CPU seconds). Execution time of the CPM algorithm is dependent on the width of the binary pattern whereas that of the DT methodology is independent of the width of the pattern. Thus with thin characters the CPM is found to be faster than the DT method. This makes the CPM algorithm preferable even to DT method for processing of binary patterns generated by a low resolution scanner

5.2. Future Work

Firstly, the algorithm could be tested on a larger data set to create an averaging effect. Secondly the validity and comprehensiveness of the test data set to become an acceptable bench mark has to be examined. The average processing time needed for thinning a pattern by the CPM algorithm is found to be 0.863 CPU seconds. Hence thinning alone would require approximately 1600 CPU seconds for a page of about 2000 characters. On top of this, an OCR system would need additional time for the latter stages of feature extraction and classification, making the total system response too slow. Thus, parallel implementation of the proposed CPM algorithm and its integration into a VLSI chip should be further explored. Application of the proposed CPM methodology to grey level images and 3-D skeletonisation is yet another area to explore.

REFERENCES

1. C.Y. Suen, M. Berthod and S. Mori, 1980: Automatic Recognition of Handprinted Characters - The State of the Art, Proc. of the IEEE, Vol. 69, No. 4, pp. 469-483.
2. L. Lam and C.Y. Suen, 1988: Structural Classification and Relaxation Matching of Totally Unconstrained Handwritten Zip-code Numbers, Pattern Recognition, Vol. 21, No. 1, pp. 19-31.
3. J. Kittler and J. Illingworth, 1985: Relaxation Labelling Algorithms - A Review, Image Vision Computing, Vol. 3, No. 4, pp. 206-216.
4. I.D. Judd, 1979: Compression of Binary Images by Stroke Encoding, IEE Journal of Computer and Digital Techniques, Vol. 2, pp. 41-48.
5. J. Toriwaki and S. Yokoi, 1981: Distance Transformation and Skeletons of Digitised Pictures with Applications, Progress in pattern Recognition, Edited by Kanal, L.N. and A. Rosenfeld, pp. 187-274.
6. T. Pavlidis, 1982: Asynchronous Thinning Algorithm, Computer Graphics and Image Processing, Vol. 20, pp. 133-157.5.
7. C.J. Hilditch, 1969: Linear Skeletons from Square Cupboards, Machine Intelligence, Vol. 4, Edited by Meltzer, B. and D. Michie, American Elsevier, New York, pp. 403-420.

8. C.V.K. Rao, B. Prasada and K.R. Sarma, 1974: An Automatic Fingerprint Classification System, Proc. 2nd International Conference on Pattern Recognition, pp. 180-184.

9. J.F. Jarvis, 1980: A Method for Automating the Visual Inspection of Printed Wiring Boards, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, pp. 77-82.

10. J.F. O'Callaghan and J. Loveday, 1973: Quantitative Measurements of Soil Cracking Patterns, Pattern Recognition, Vol. 5, pp. 83-98.

11. D. Rutovitz, 1966: Pattern Recognition, Journal of Royal Statistics Society, Vol. 29, Series A, pp. 504-530.

12. S. Yokoi, J.I. Toriwaki and T. Fukumura, 1975: An Analysis of Topological Properties of Digital Binary Pictures Using Local features, Computer Graphics and Image Processing, Vol. 4, pp. 63-73.

13. H. Blum, 1964: A Transformation for Extracting New Descriptions of Shape, Proc. of Symposium of Models for the Speech and Vision Form, Boston, pp. 362-380.

14. J. Pfaltz and A. Rosenfeld, 1967: Computer Representation of Planar Regions by Their Skeletons, Communications of the ACM, Vol. 10, No. 2, pp. 119-125.

15. N.J. Naccache and R. Shinghal, 1984: Skeletonization of Binary Patterns: A Proposed Algorithm and a Multi-processor Network, Masters Thesis, Dept. of Computer Science, Concordia University, Montreal, Canada.

16. C. Arcelli and G. Saniti Di Baja, 1985: A Width Independent Fast Thinning Algorithm, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, No. 4, pp. 463-474.

17. Y.K. Chu and C.Y. Suen, 1986: An Alter· :e Smoothing and Stripping Algorithm for Thinning Digital Binary Patterns, Signal Processing, Vol. 11, pp. 207-222.

18. S. Suzuki and K. Abe, 1986: Sequential Thinning of Binary Pictures using Distance Transformation, Proc. of 8th International Conference on Pattern Recognition, Paris, France, pp. 289-292.

19. A.R. Dill, M.D. Levine and P.B. Noble, 1987: Multiple Resolution Skeletons, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No. 4, pp. 495-503.

20. V.K. Govindan and A.P. Prasad, 1987: A Pattern Adaptive Thinning Algorithm, Pattern Recognition, Vol. 20, No. 6, pp. 623-637.

21. P.C.K. Kwok, 1988: A Thinning Algorithm by Contour Generation, Communications of the ACM, Vol. 31, No. 11, pp. 1314-1324.

22. T.Y. Zhang, and C.Y. Suen, 1984: A Fast Parallel Algorithm for Thinning Digital Patterns, Communications of the ACM, Vol. 27, No. 3, pp. 236-239.

23. W. Xu and C. Wang, 1987: CGT: A Fast Thinning Algorithm Implemented on Sequential Computer, IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-17, No. 5, pp. 847-851.

24. Z.Y. Qin and Per E. Danielsson, 1988: Inspection of Printed Circuit Boards by Connectivity Preserving Shrinking, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-10, No. 5, pp. 737-742.

25. Y. Xia, 1988: Minimizing the Computing Complexity of Iterative Sequential Thinning Algorithm, Proc. of the 9th International Conference on Pattern Recognition, Rome, Italy, pp. 721-723.

26. P.C.K. Kwok, 1989: Connectivity Preserving Shrinking by Isotropic Erosion, The 6th Scandinavian Conference on Image Analysis, Oulu, Finland.

27. P.S.P. Wang and Y.Y. Zhang, 1989: A Fast and Flexible Thinning Algorithm, IEEE Transactions on Computers, Vol. 38, No. 5, pp. 741-745.

28. H.E. Lu and P.S.P. Wang, 1986: A Comment on 'A Fast Parallel Algorithm for Thinning Digital Patterns', Communications of the ACM, Vol. 29, pp. 239-242.

29. Y. Xia, 1989: Skeletonisation via the Realization of the Firefronts' Propagation and Extinction in Digital Binary Shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No. 10, pp. 1076-1086.

30. J.H. Sossa, 1989: An Improved Parallel Algorithm for Thinning Digital Patterns, Pattern Recognition Letters, pp. 77-80.

31. Z. Guo and R.W. Hall, 1989: Parallel Thinning with Two-subiteration Algorithms, Communications of the ACM, Vol. 32, No. 3, pp. 359-373.

32. E.R. Davies and A.P.N. Plummer, 1981: Thinning Algorithms: A Critique and a New Methodology, Pattern Recognition, Vol. 14, No. 1, pp. 53-63.

33. T. Pavlidis, 1982: An Asynchronous Thinning Algorithm, Computer Graphics and Image Processing, Vol. 27, pp. 133-157.

34. M.R. Abairid, G.Martin and C.Y. Suen, 1990: Tracing Centre-Lines of Digital Patterns using Maximal-Square and Maximal-Circle Algorithms, Proc. of Vision Interface'90, Halifax, Canada, pp. 67-79.

35. M.R. Abairid, 1987: Tracing Centre Lines of Digital Patterns using Maximal Square and Maximal Circle Algorithms, Major Report, Dept. of Computer Science, Concordia University, Montreal, Canada.

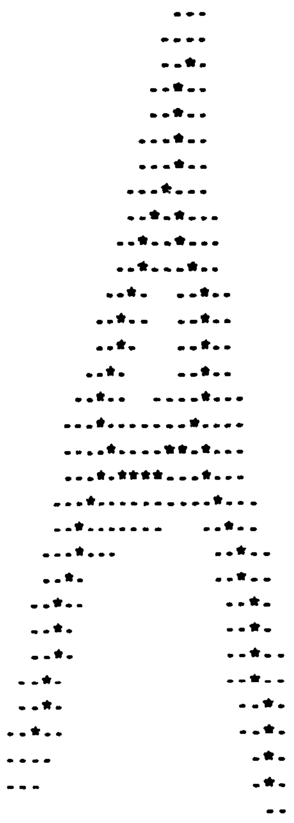
36. G. Borgefors, 1984: Distance Transformations in Arbitrary Dimensions, Computer Graphics and Image Processing, Vol. 27, pp. 321-345.

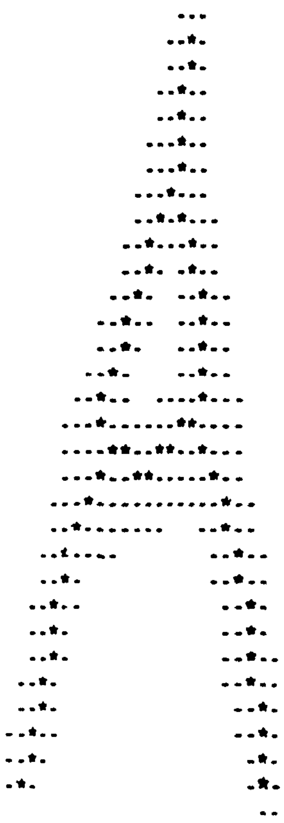
37. T. Wakayama, 1982: A Core-line Tracing Algorithm based on Maximal Square Moving, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-4, No. 1, pp. 68-74.

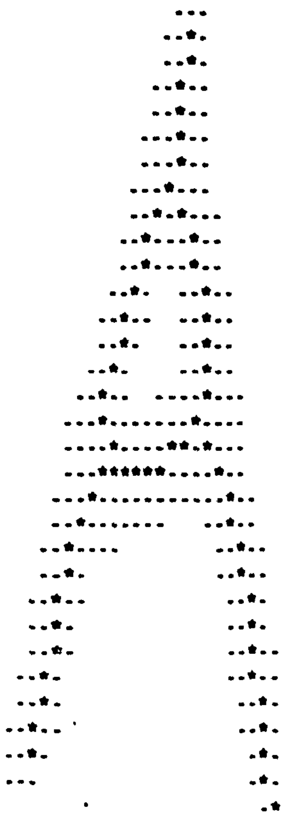
38. H. Tamura, 1978: A Comparison of Line Thinning Algorithms from Digital Geometry Viewpoint, Proc. of 4th International Conference on Pattern Recognition, Kyoto, Japan, pp. 715-719.

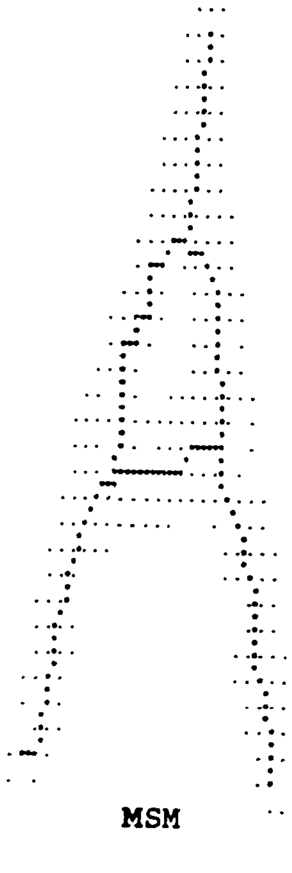
APPENDIX

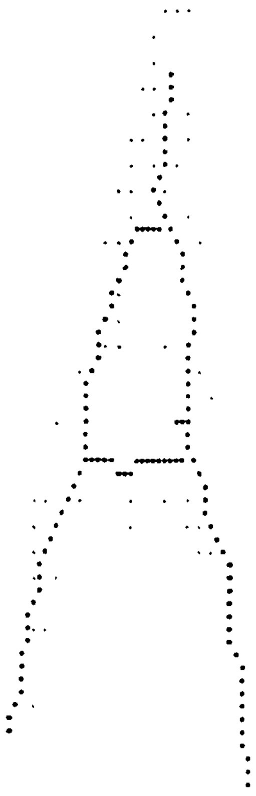
List of data set and the skeletons generated by the five algorithms (CPM, SPTA, DT, MSM and MCM).

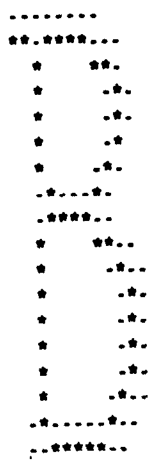

CPM

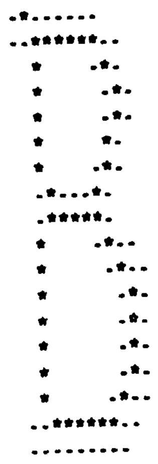

DT

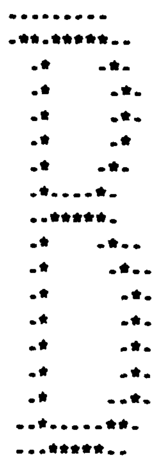

SPTA

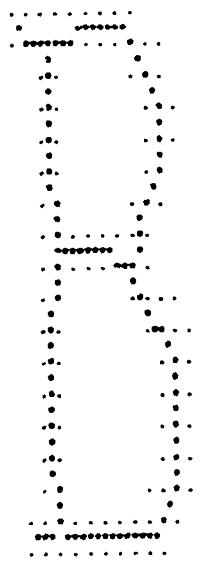

MSM

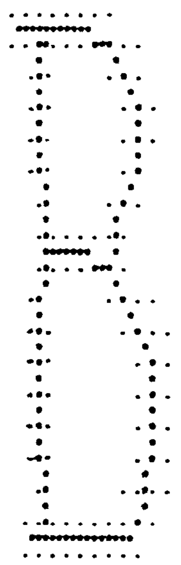

MCM

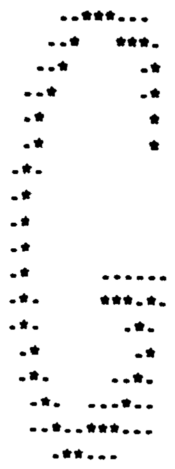

CPM

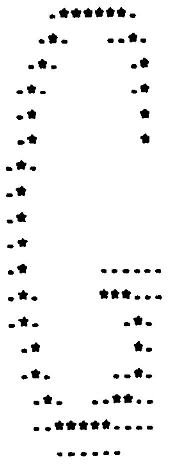

DT

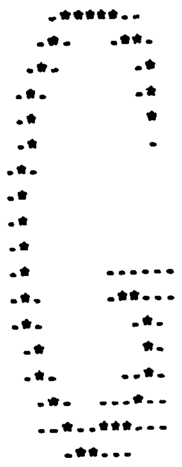

SPTA

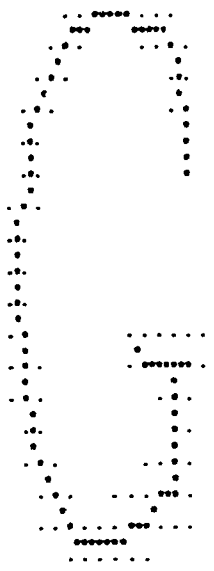

MSM

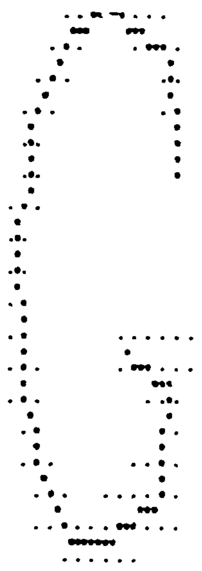

MCM

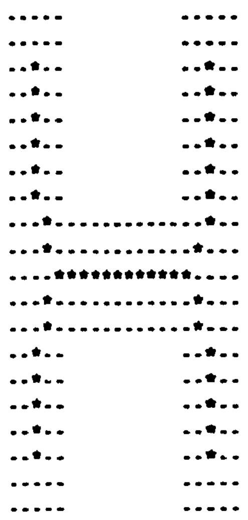

CPM


DT

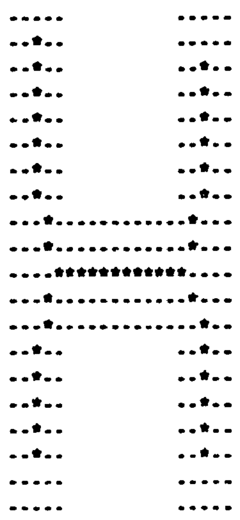

SPTA


MSM

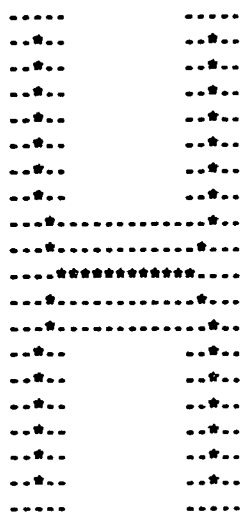

MCM



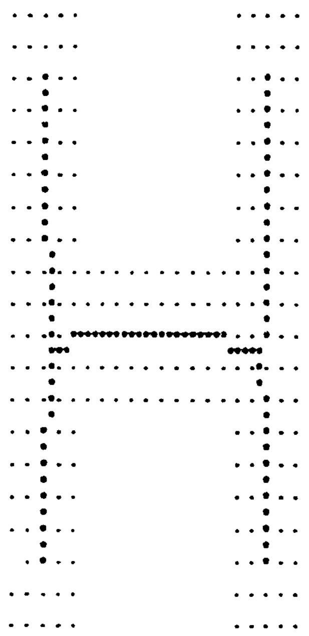
CPM



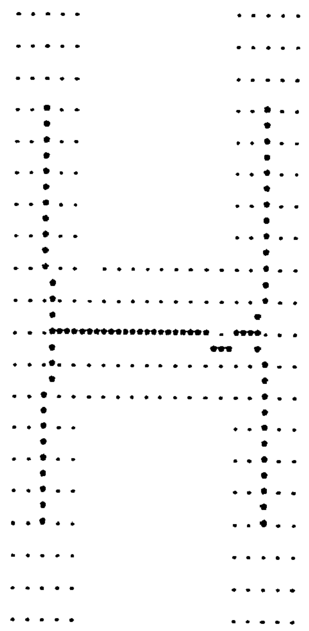
DT



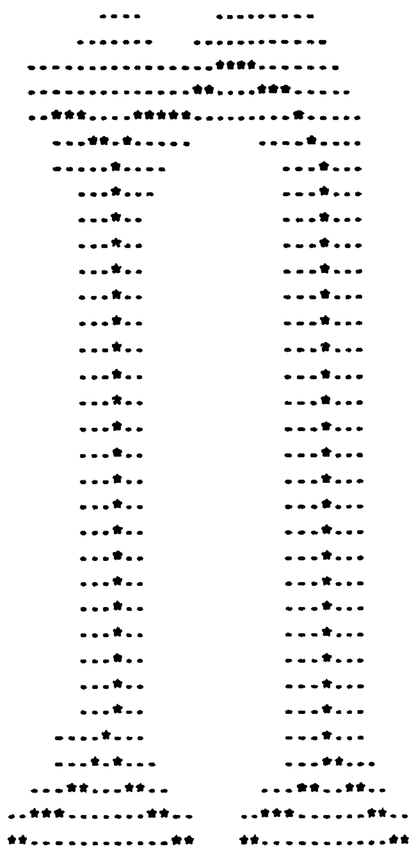
SPTA

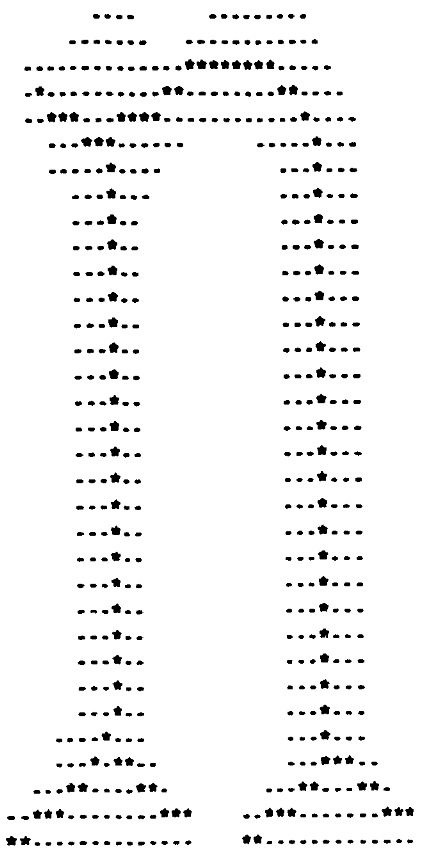


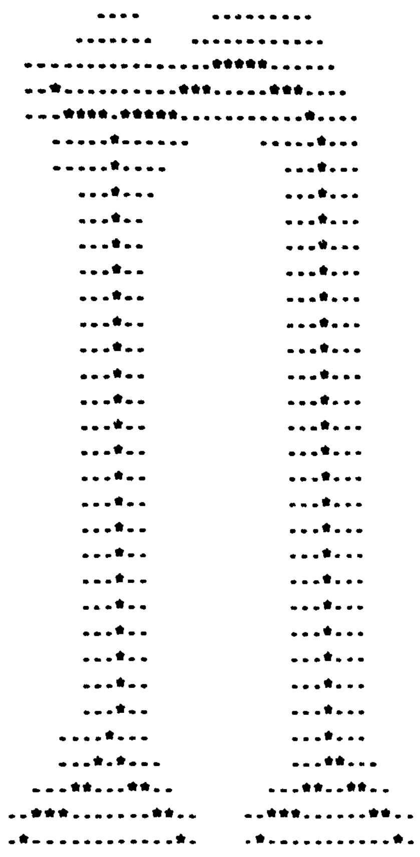
MSM

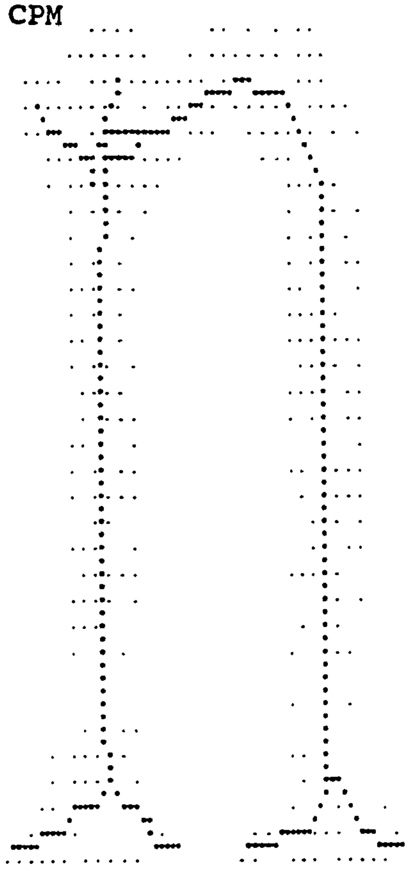


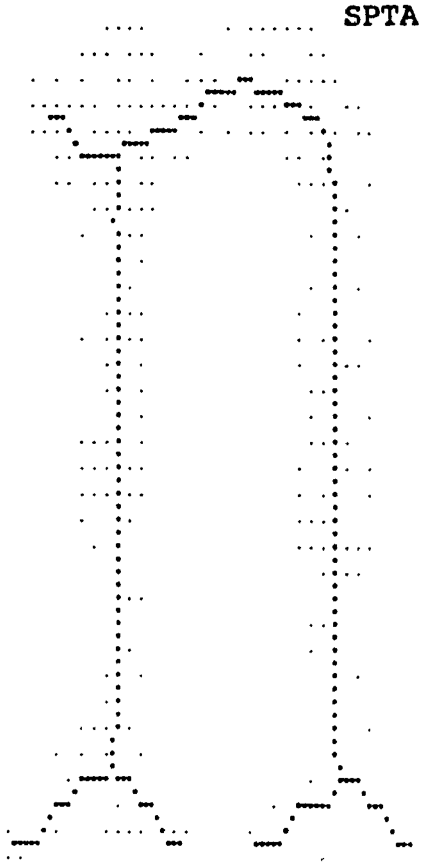
MCM

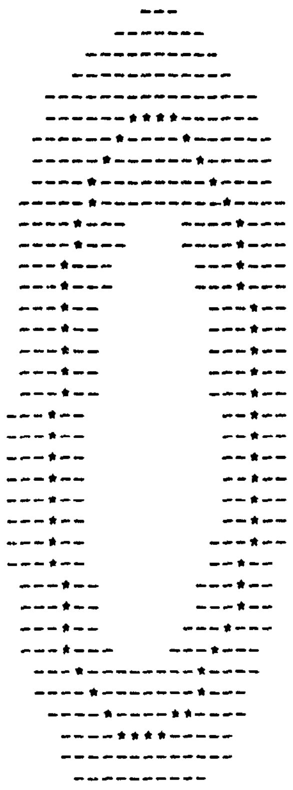

CPM

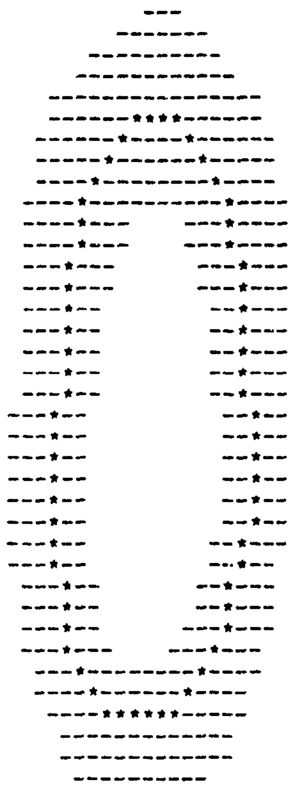

DT

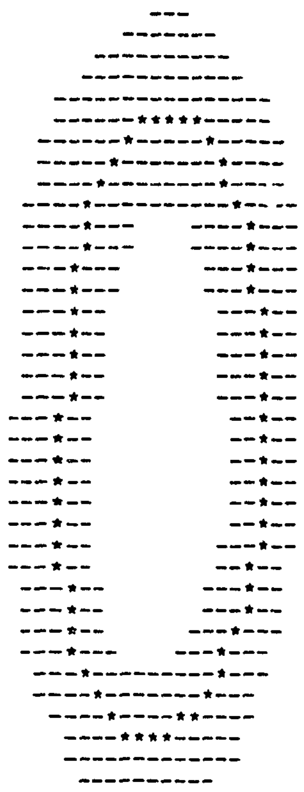

SPTA

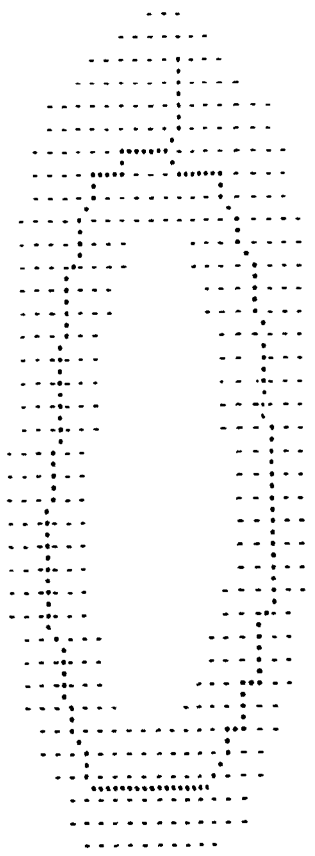

MSM

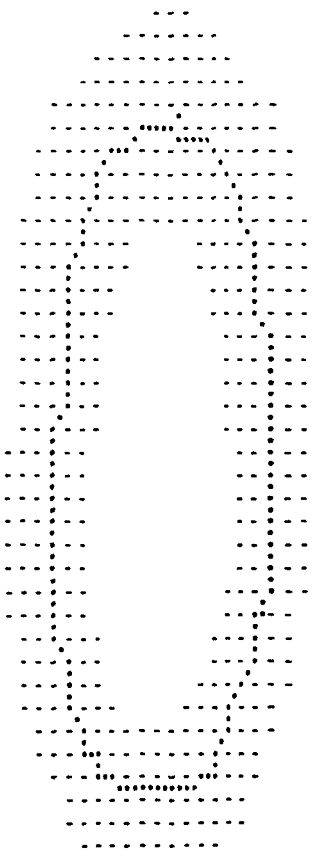

MCM

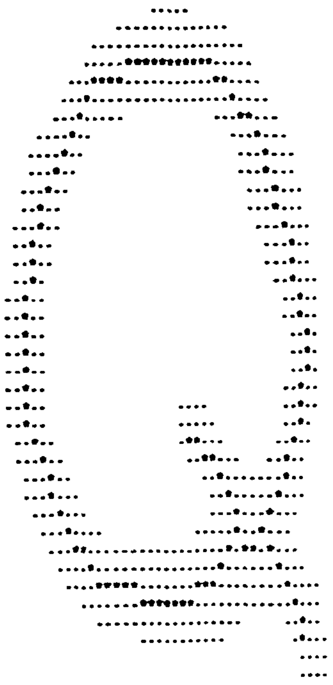

CPM

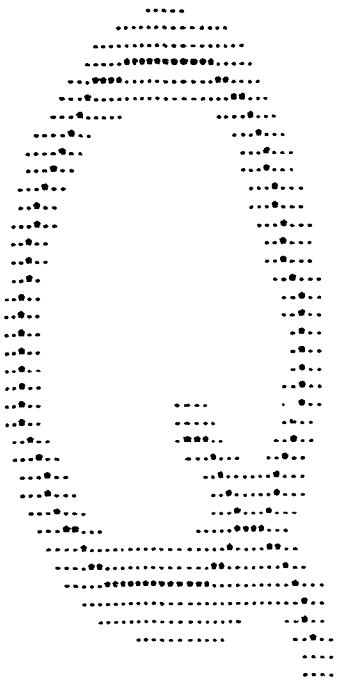

DT

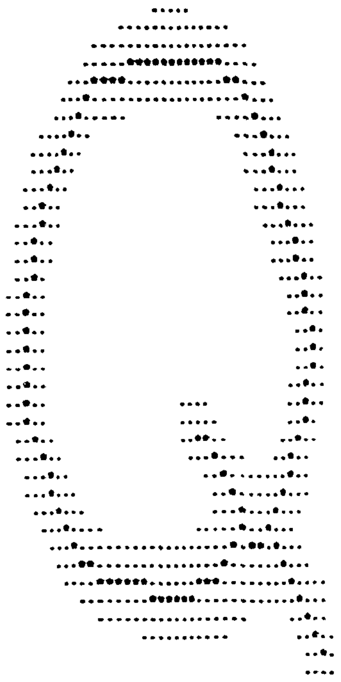

SPTA

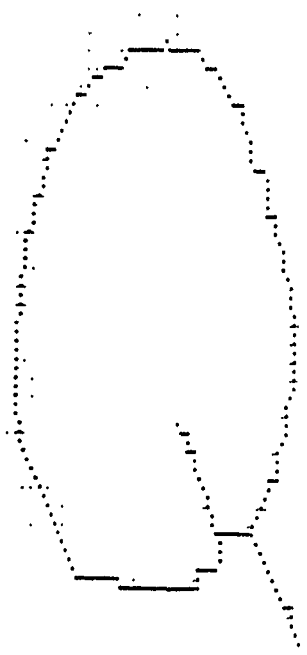

MSM

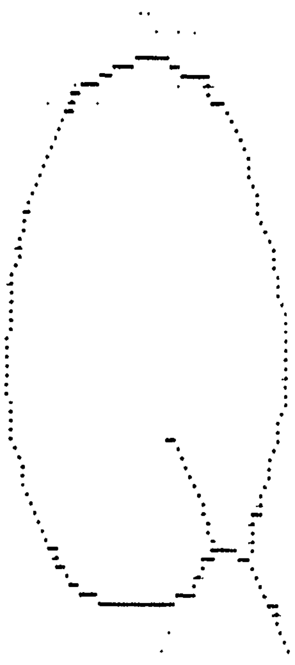

MCM

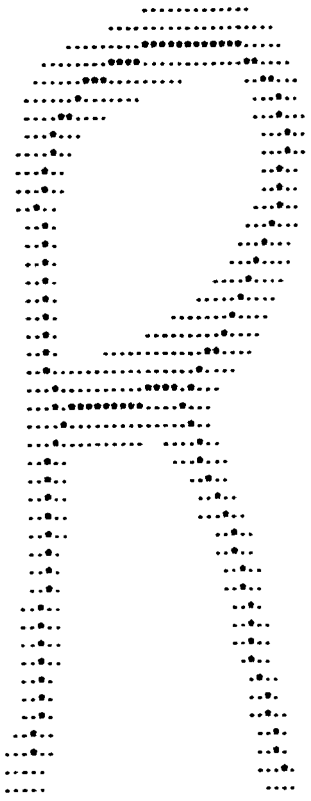

CPM

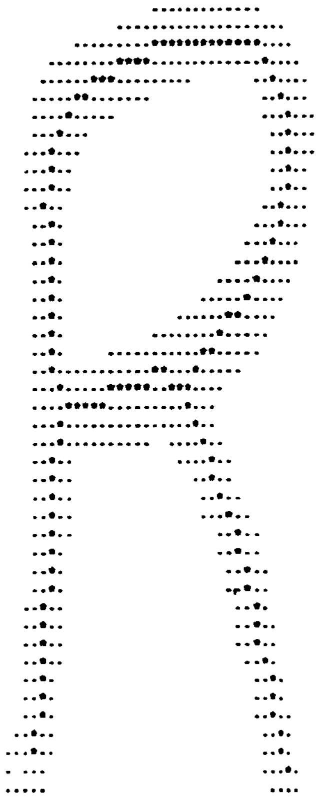

DT

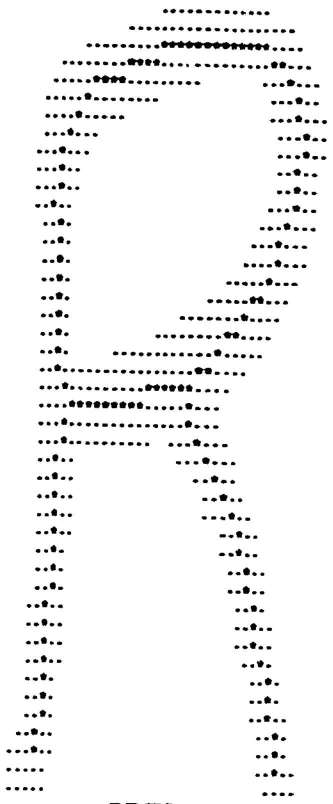

SPTA

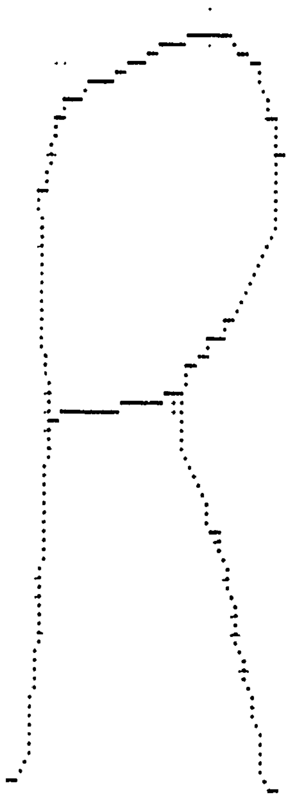

MSM

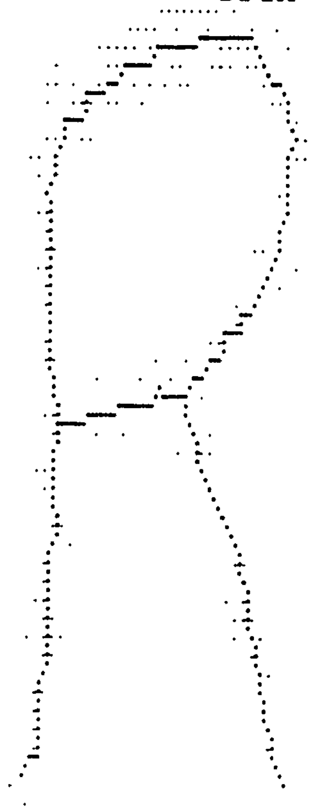

MCM

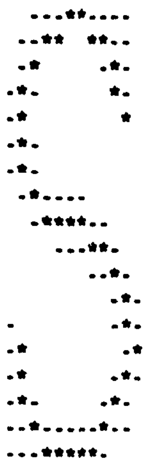

CPM

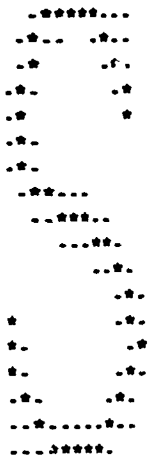

DT

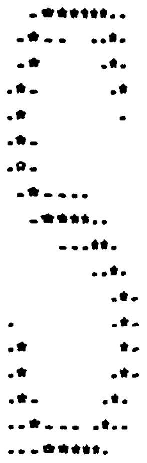

SPTA

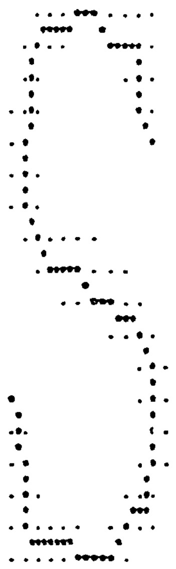

MSM

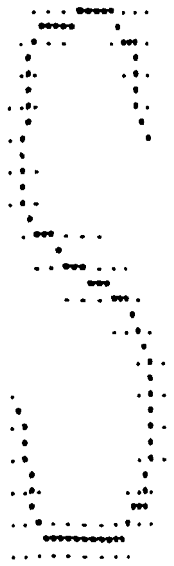

MCM

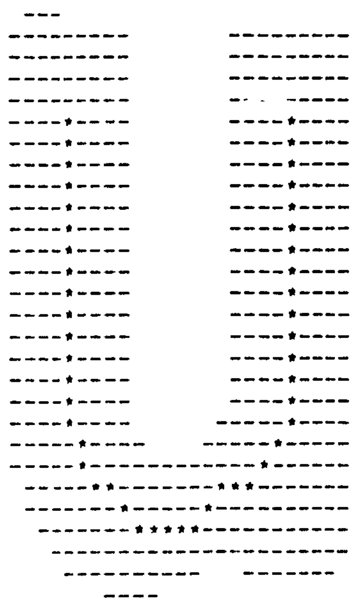

CPM

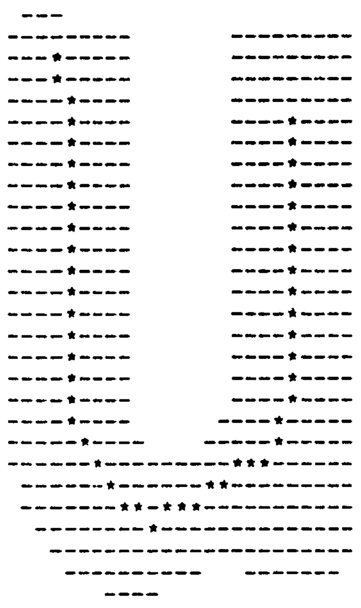

DT

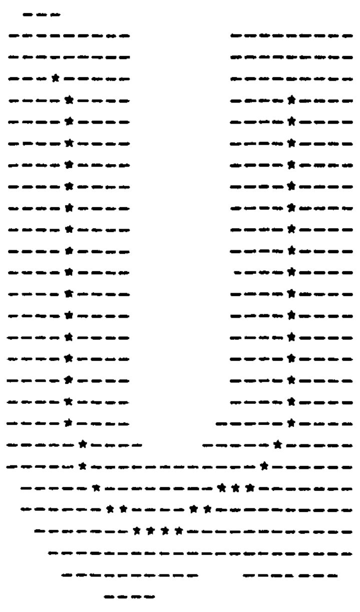

SPTA

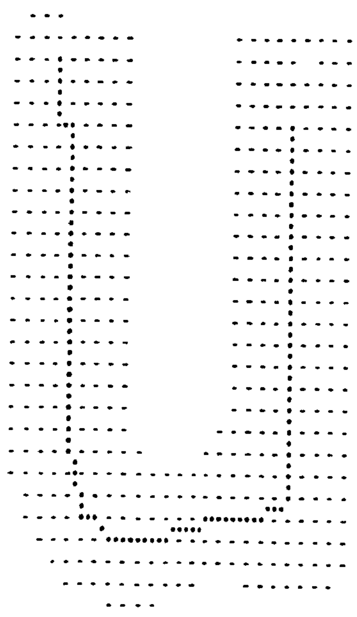

MSM

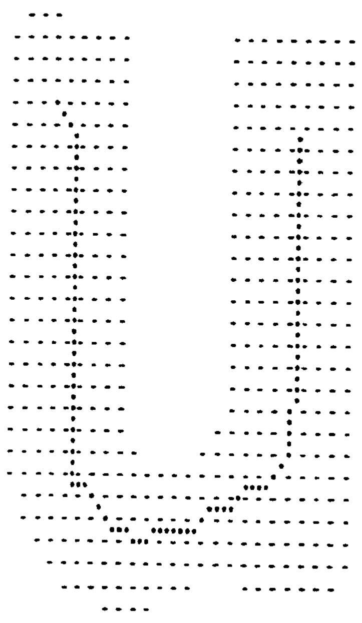

MCM

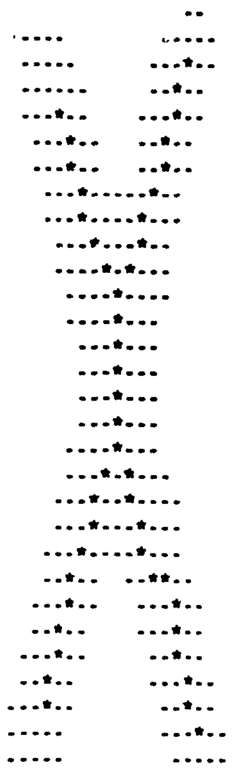

CPM


DT

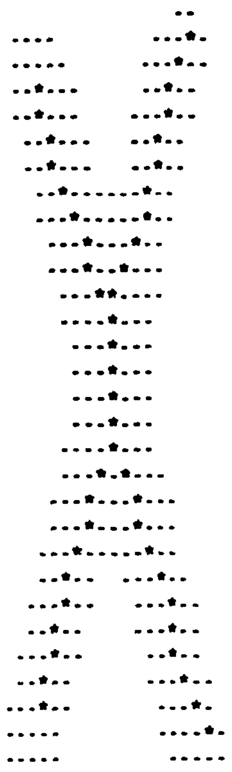

SPTA


MSM

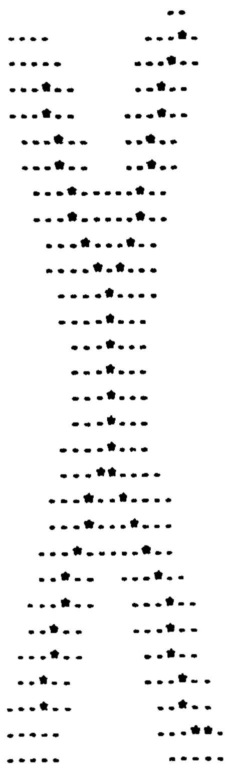

MCM



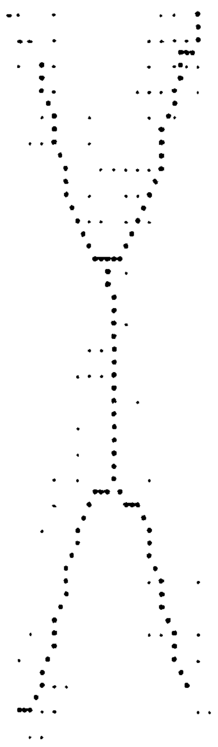
CPM



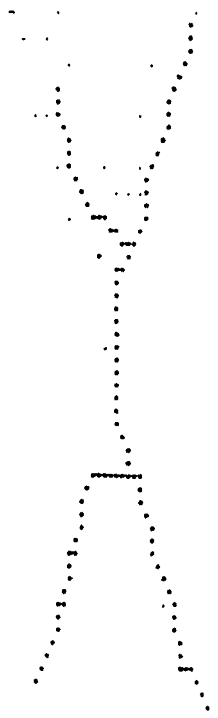
DT



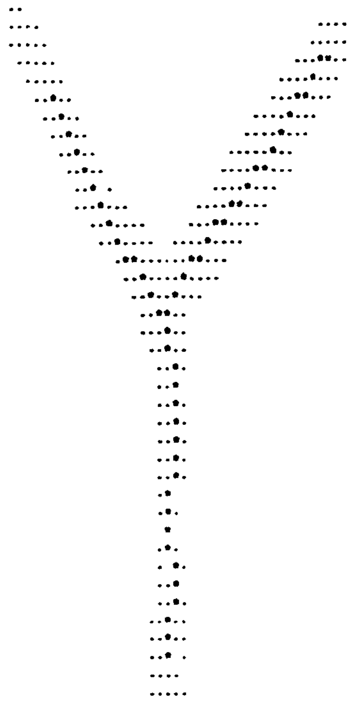
SPTA

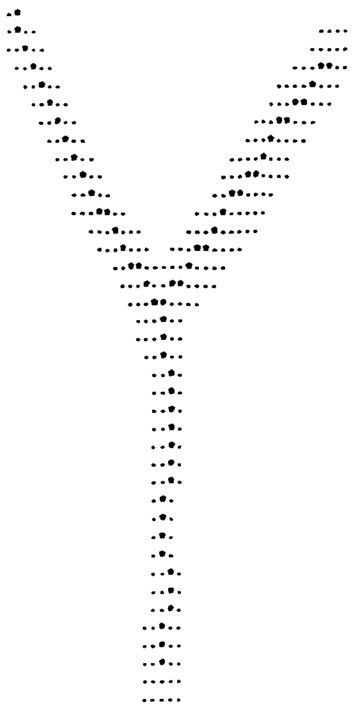


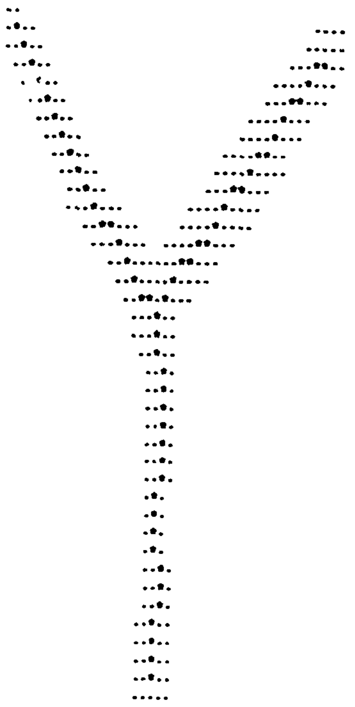
MSM

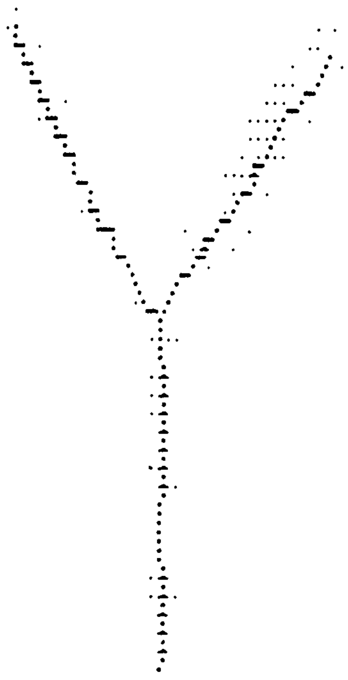


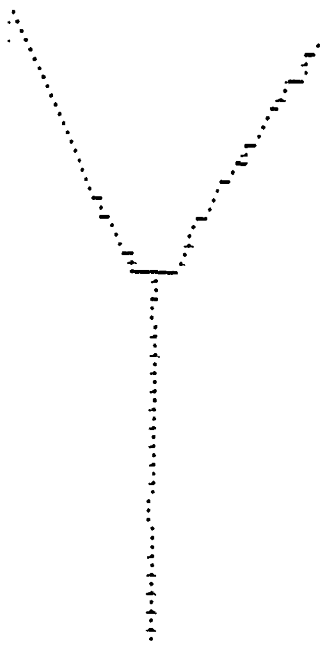
MCM

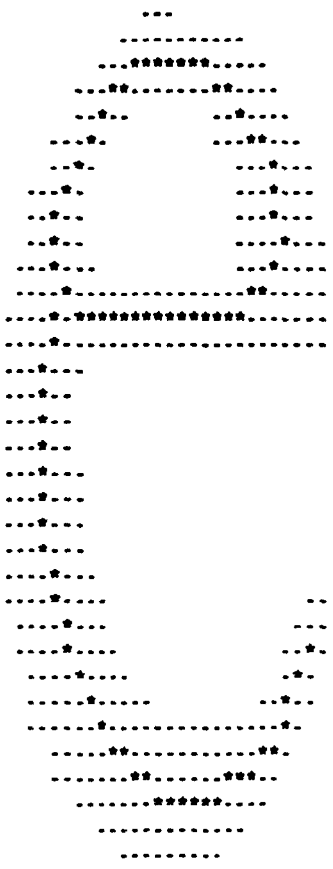

CPM

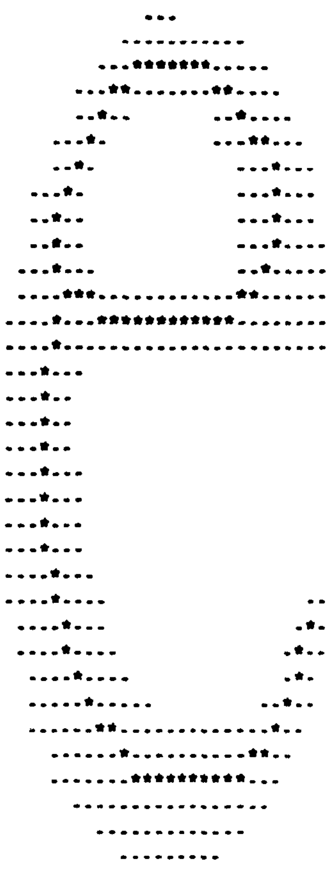

DT

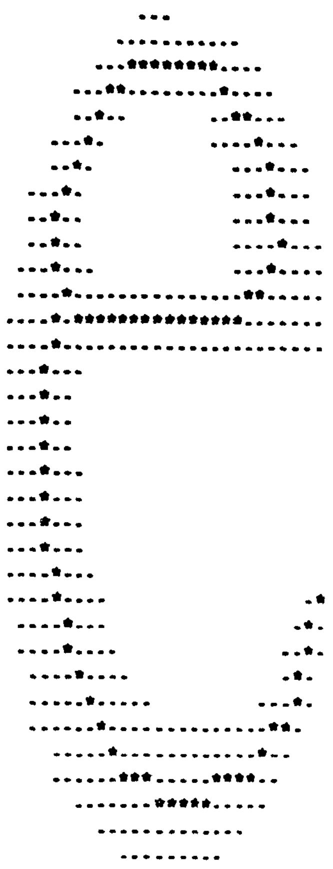

SPTA

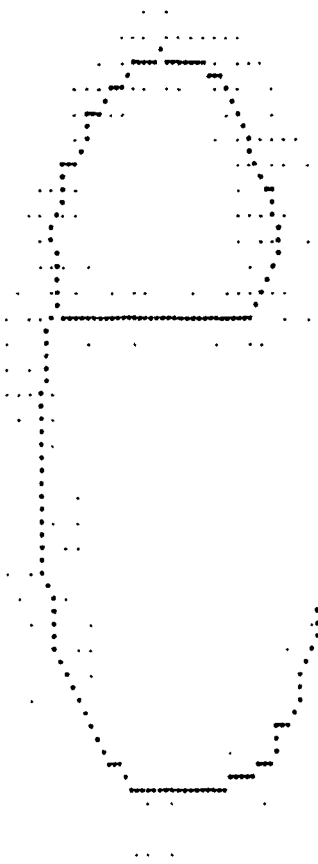

MSM

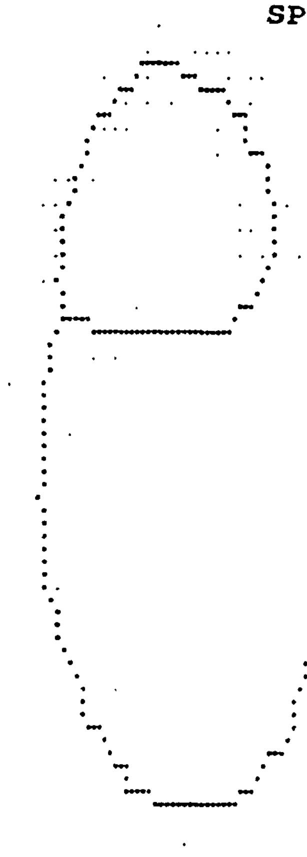

MCM

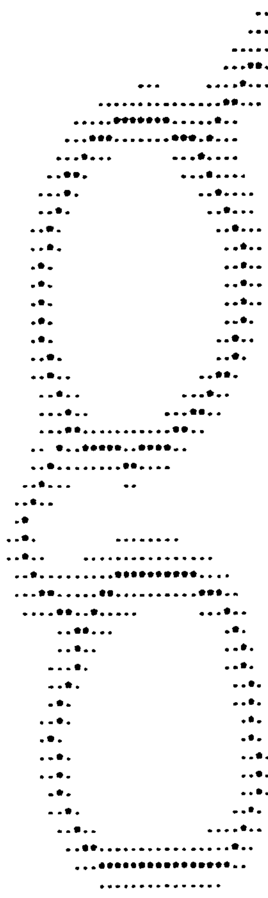

CPM

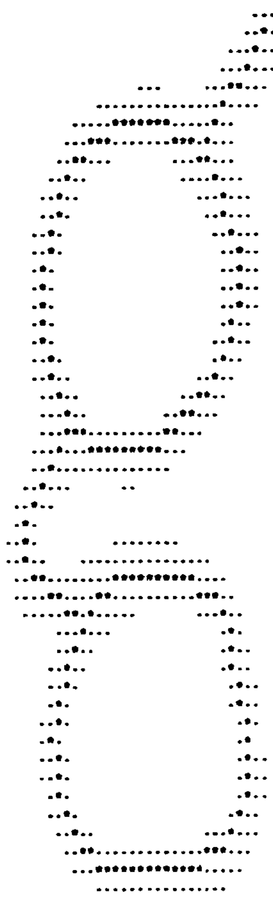

DT

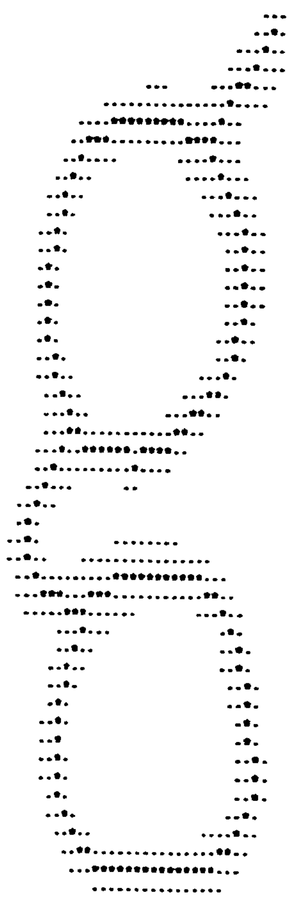

SPTA

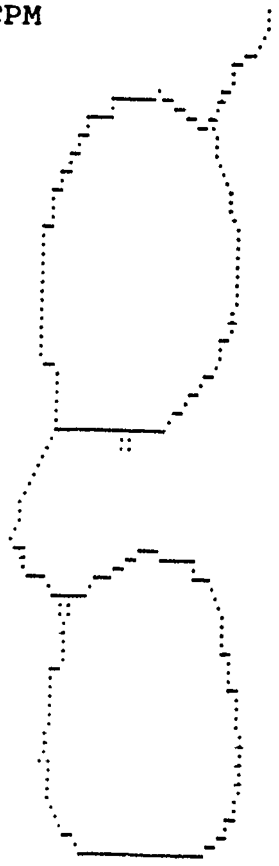

MSM

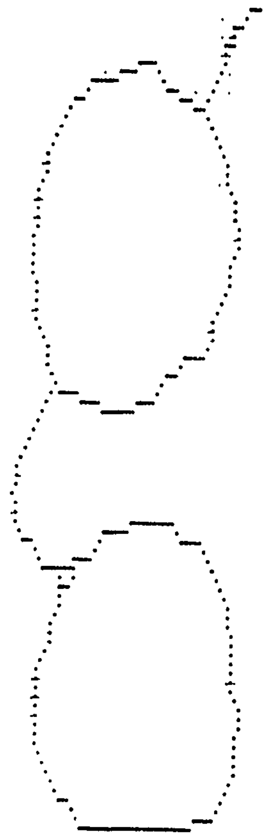

MCM

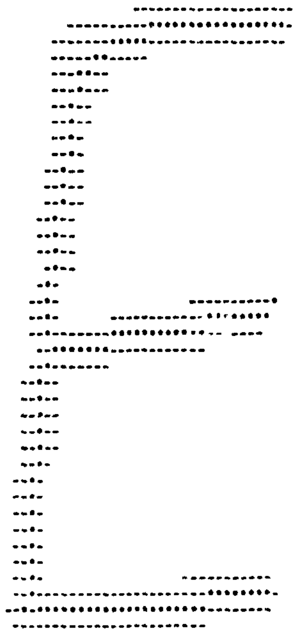

CPM

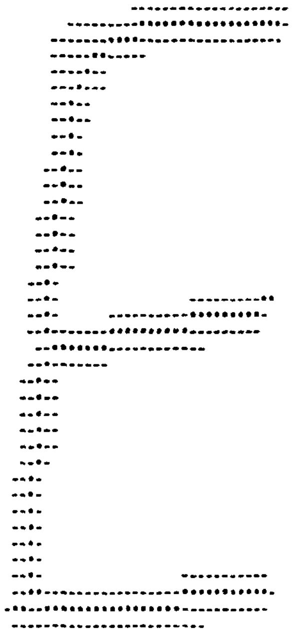

DT

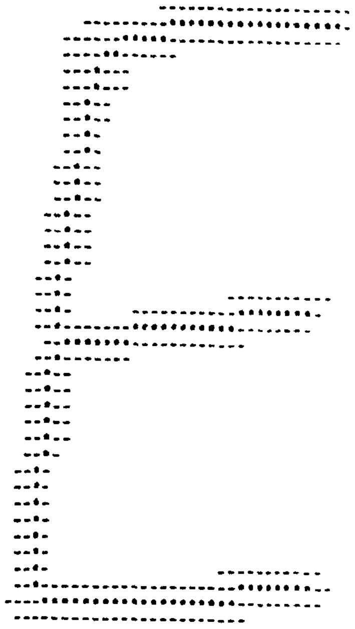

SPTA

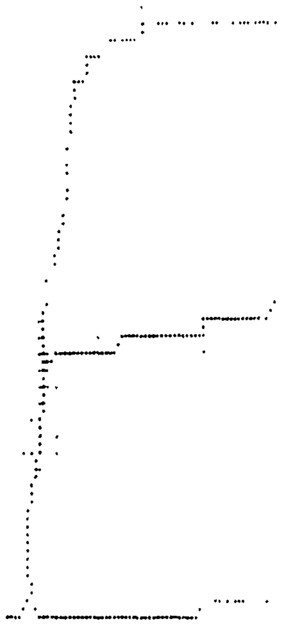

MSM

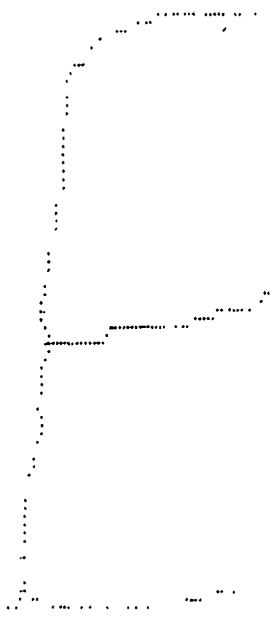

MCM

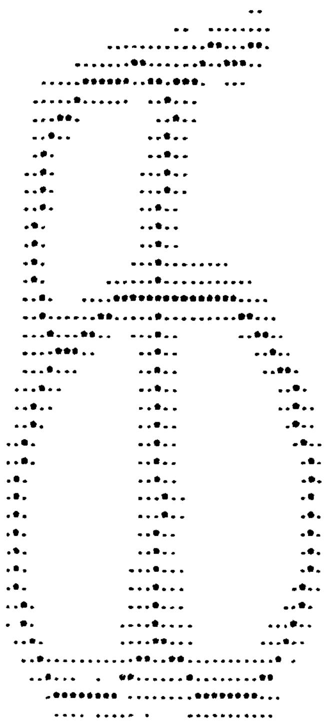

CPM

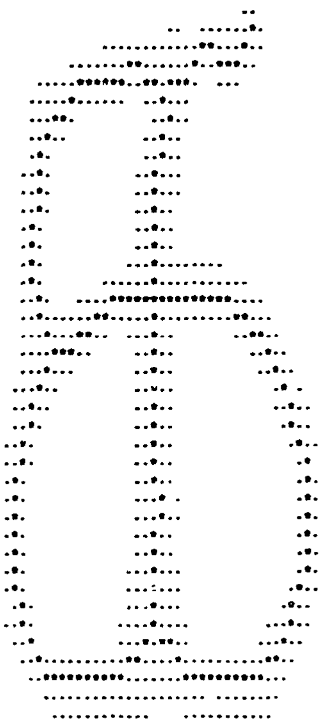

DT

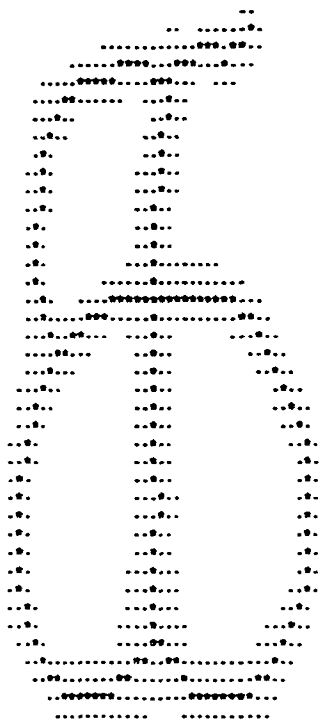

SPTA

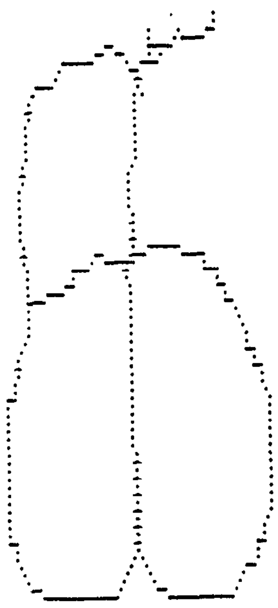

MSM

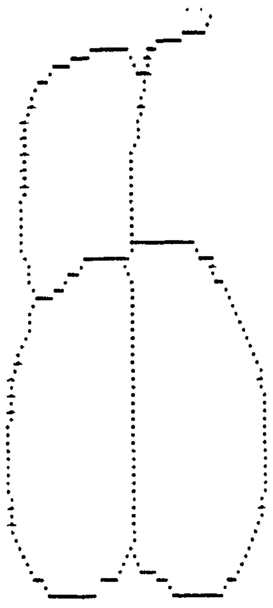

MCM

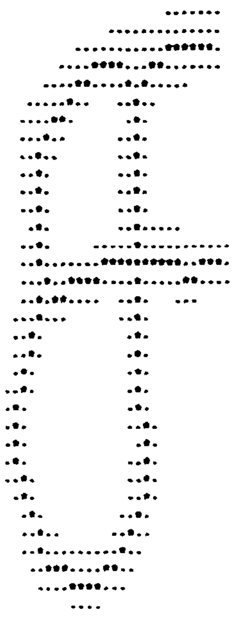

CPM

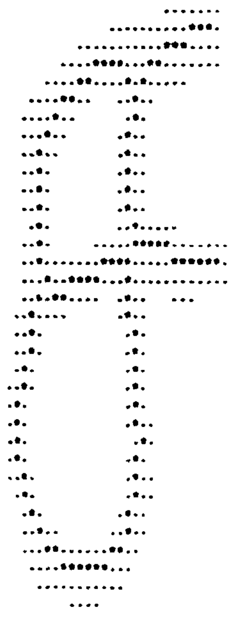

DT

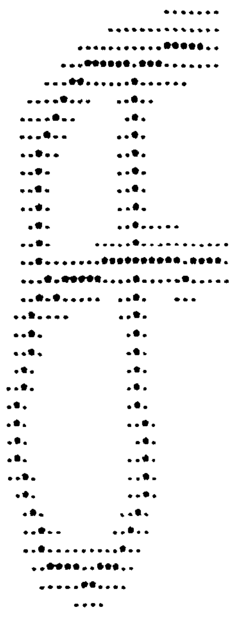

SPTA

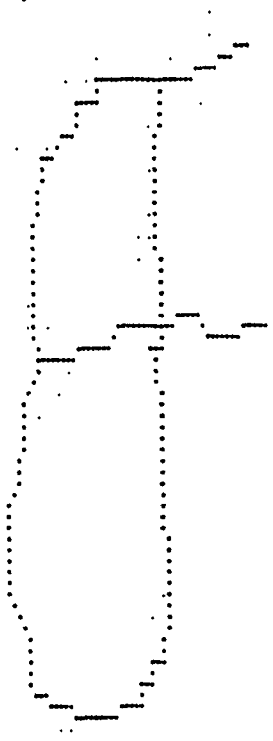

MSM

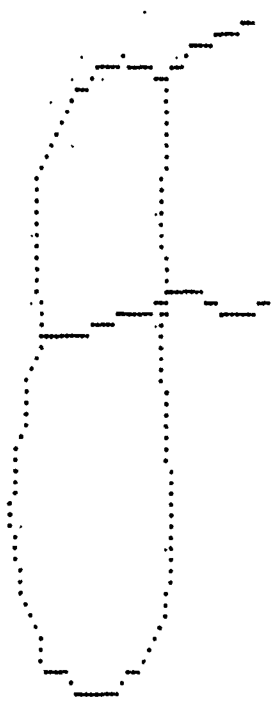

MCM

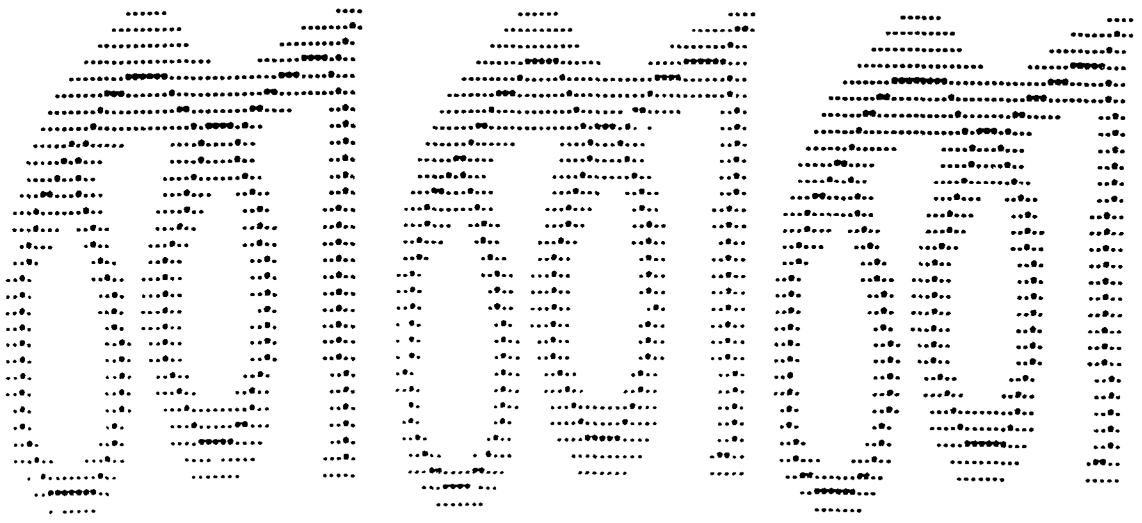

CPM


DT

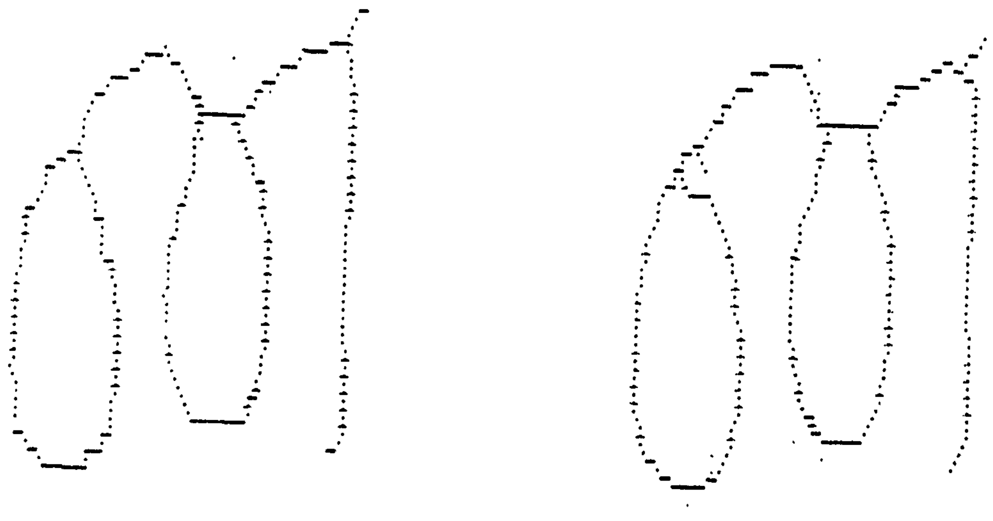

SPTA


MSM

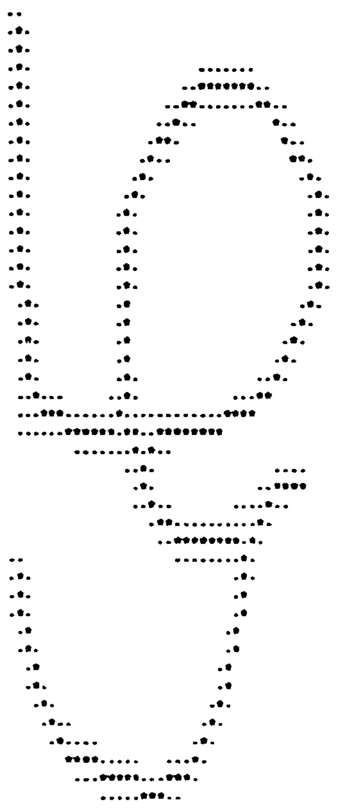

MCM

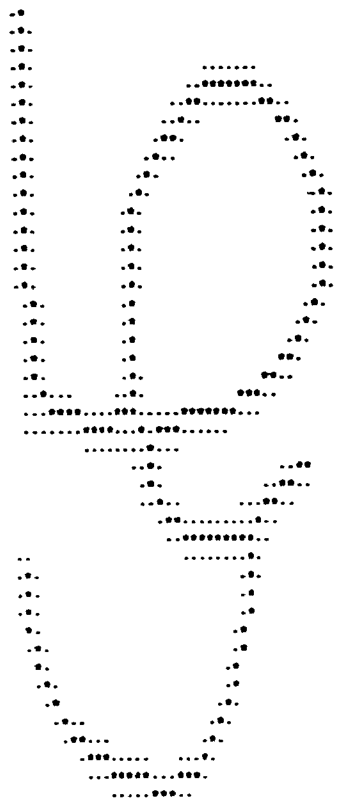


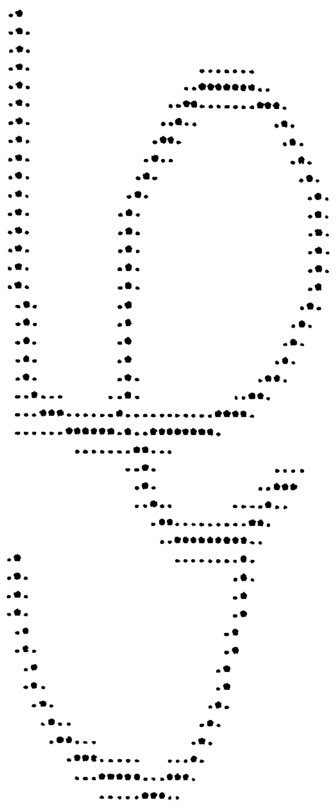
CPM DT SPTA

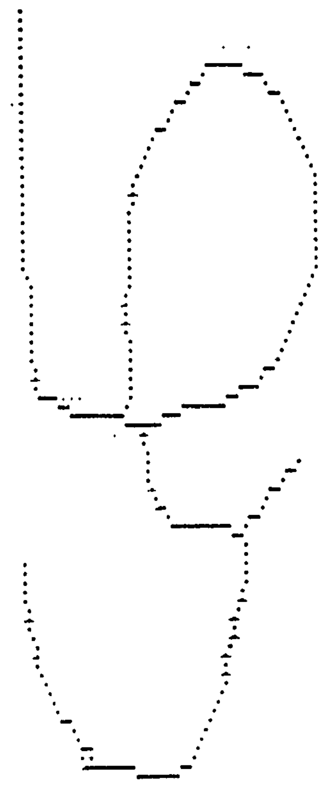


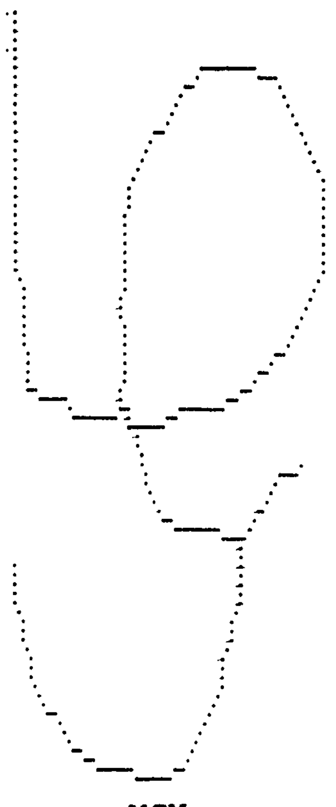
MSM MCM

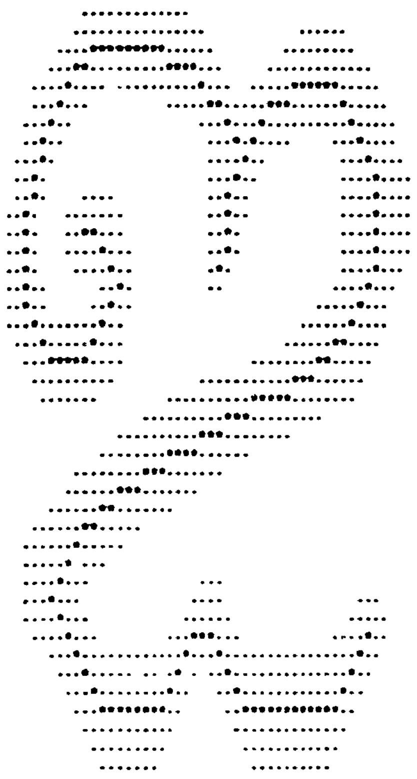

CPM

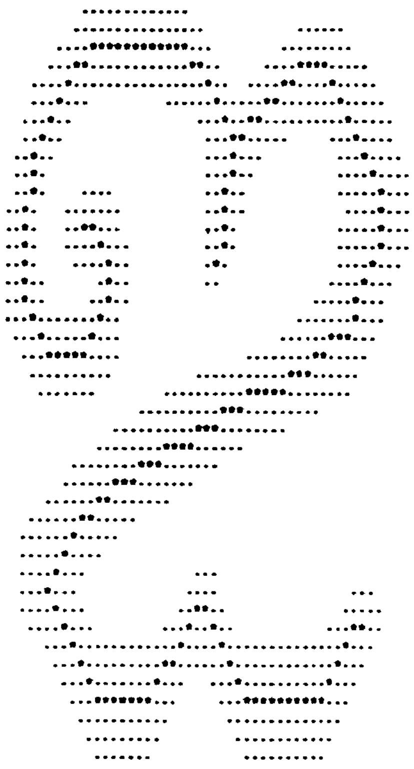

DT

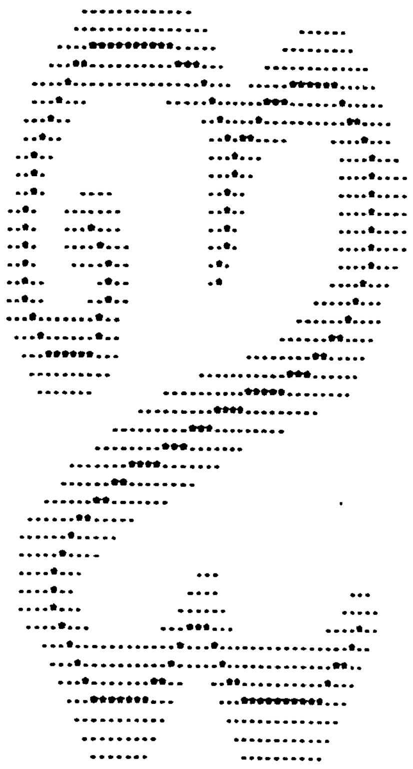

SPTA

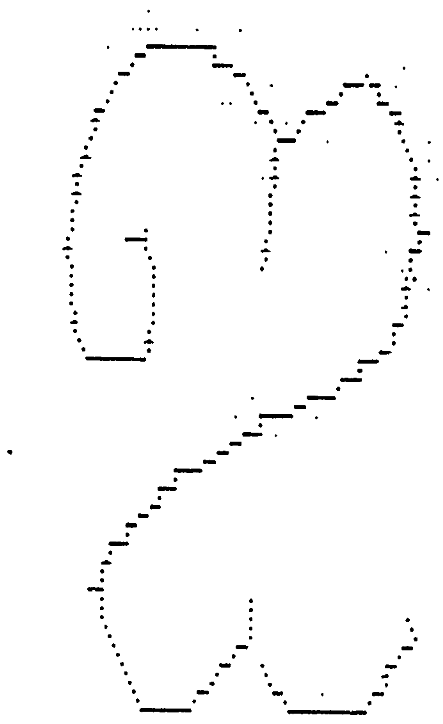

MSM

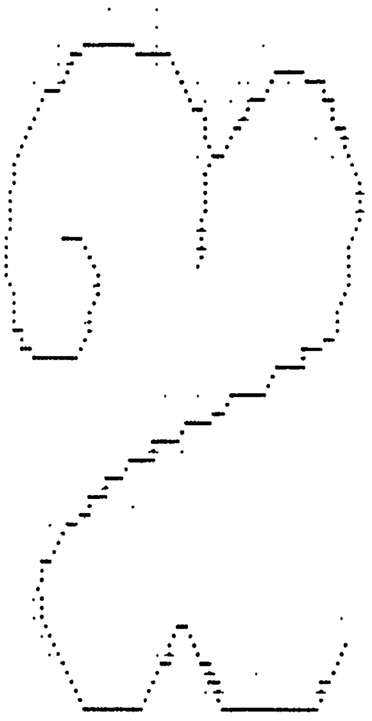

MCM

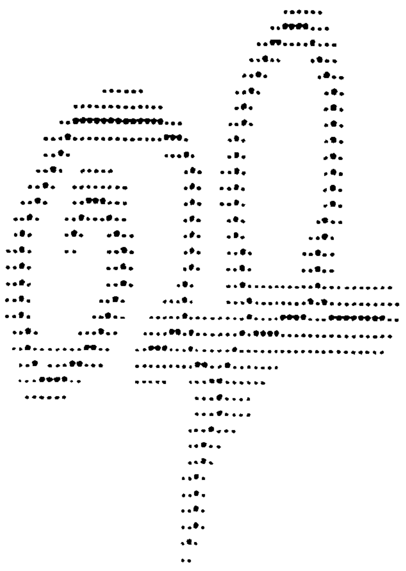

CPM

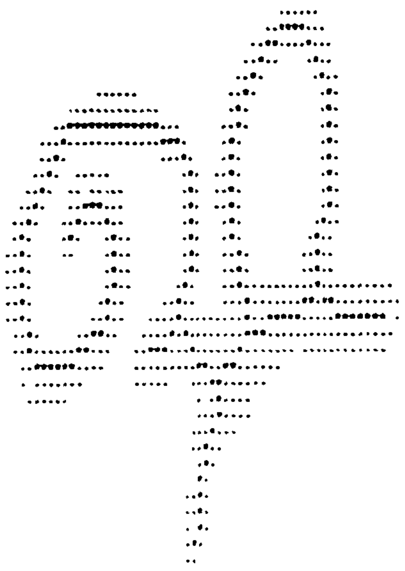

DT

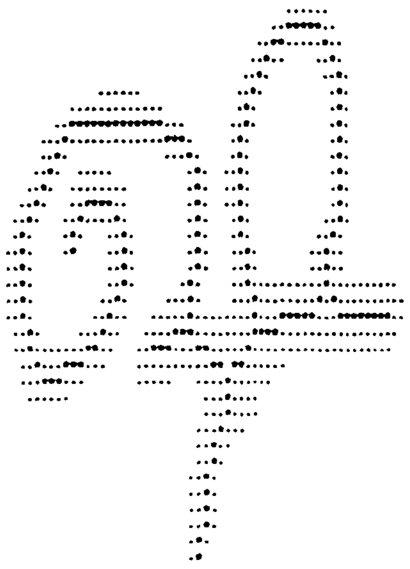

SPTA

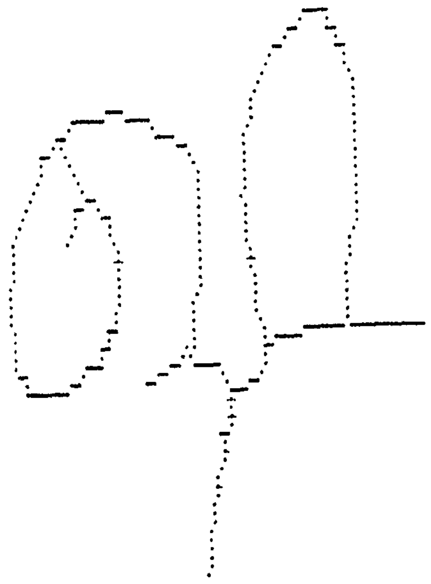

MSM

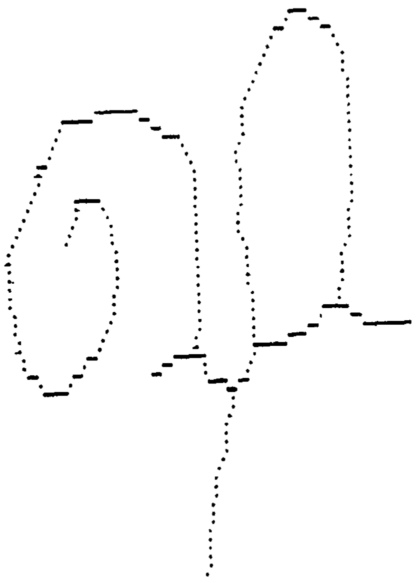

MCM

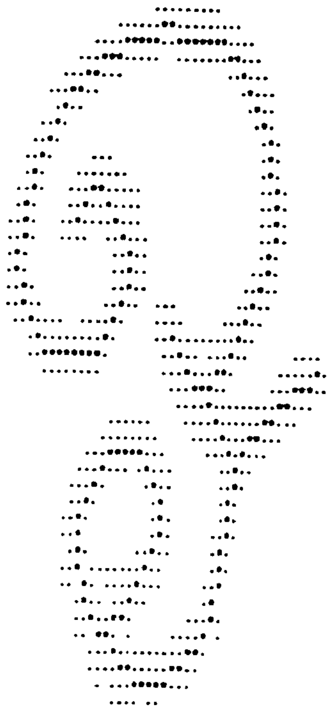

CPM

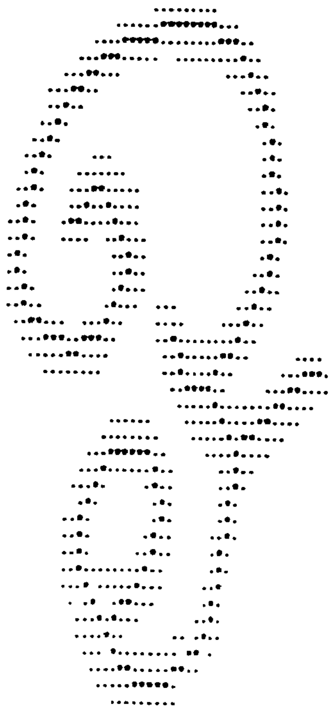

DT

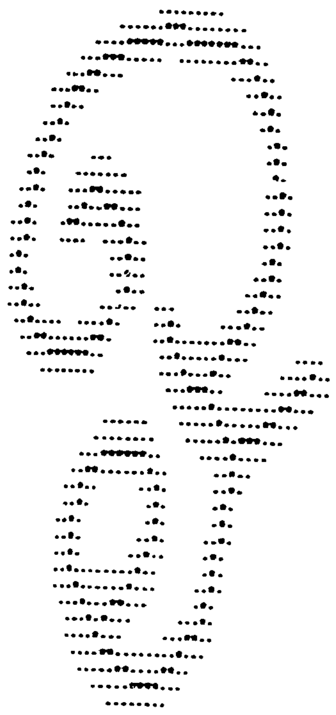

SPTA

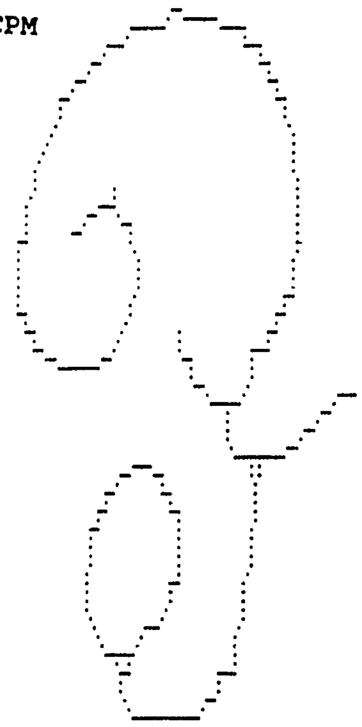

MSM

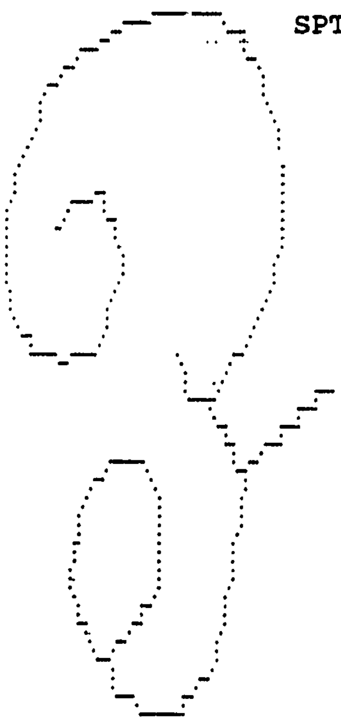

MCM

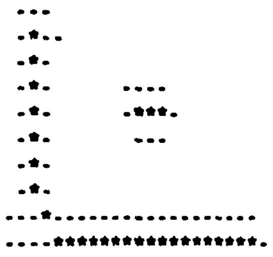

CPM

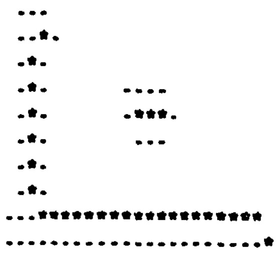

DT

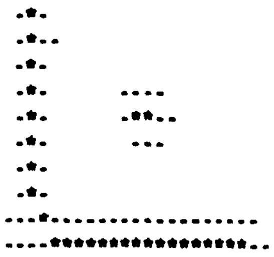

SPTA

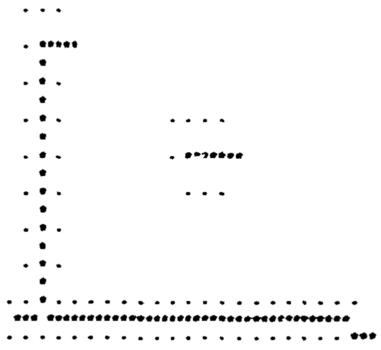

MSM

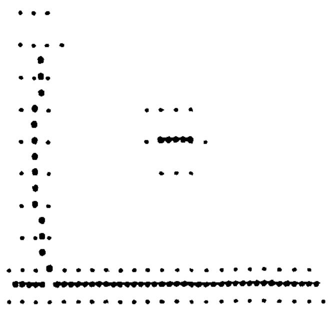

MCM

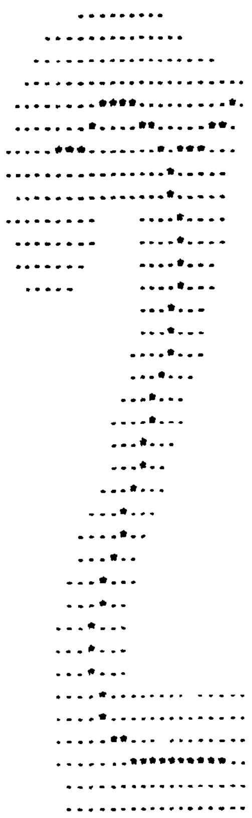

CPM

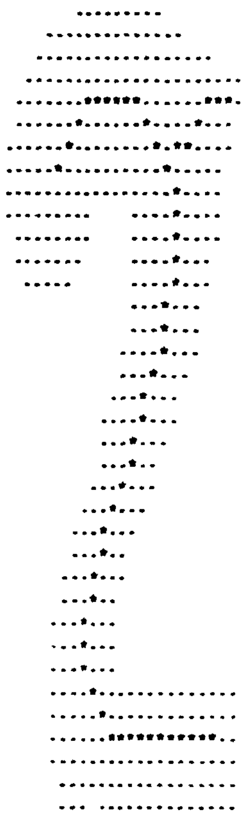

DT

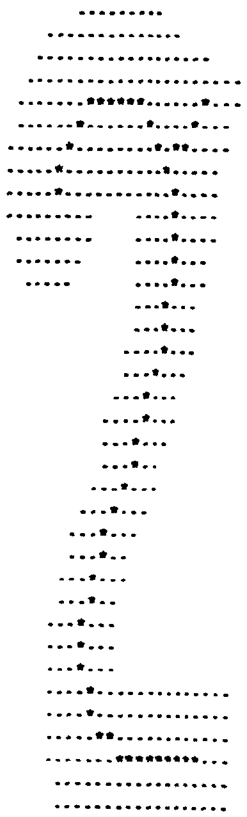

SPTA

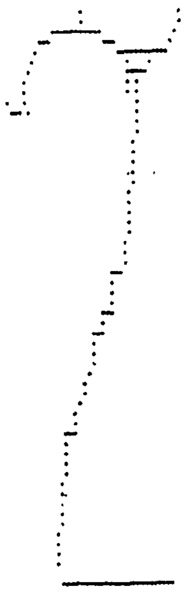

MSM

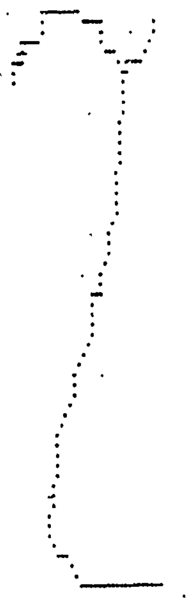

MCM

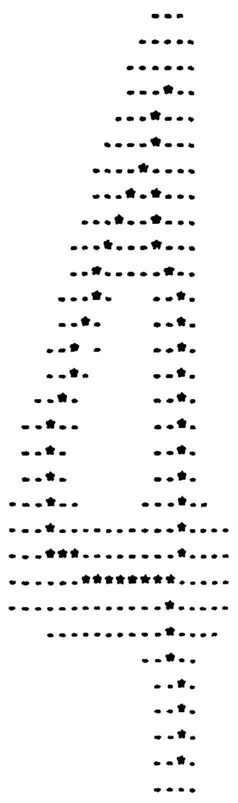

CPM

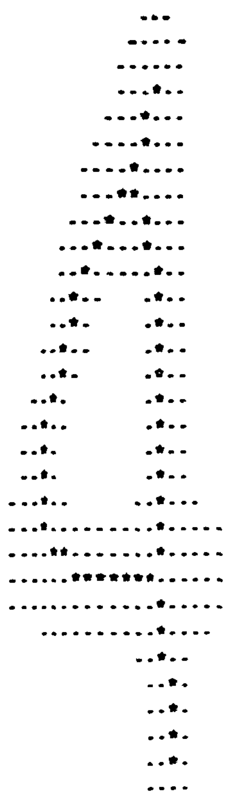

DT

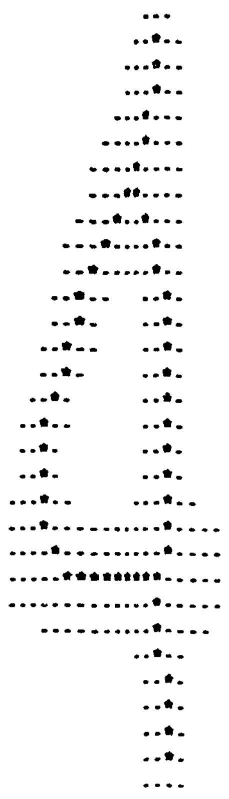

SPTA

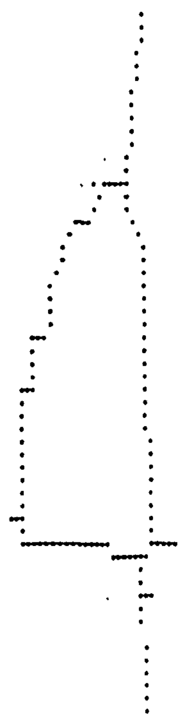

MSM

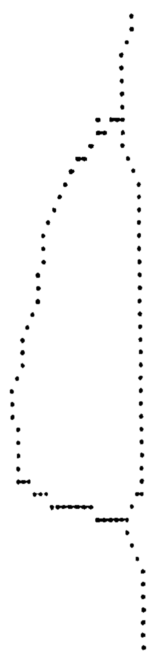

MCM

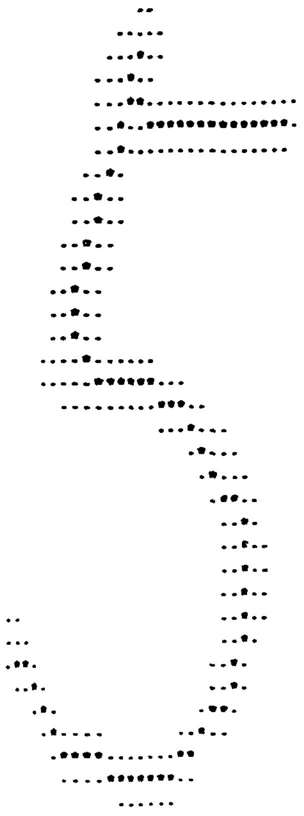

CPM

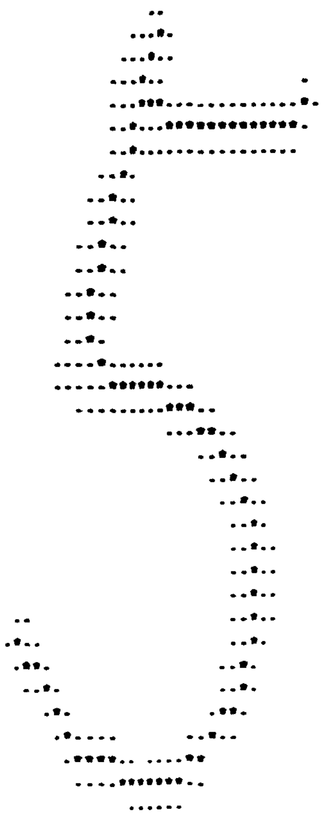

DT

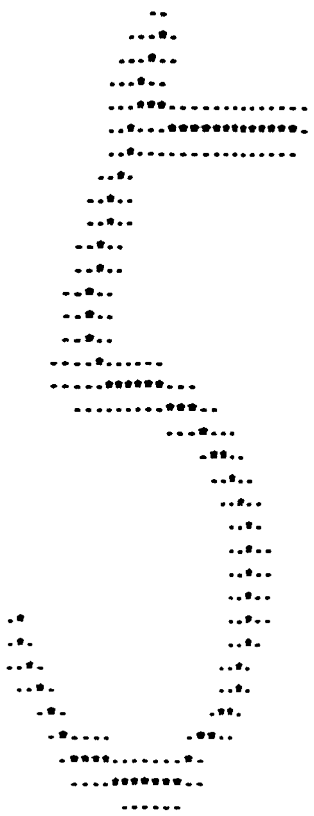

SPTA

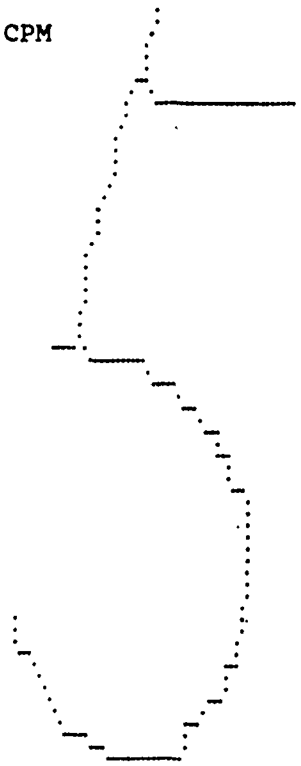

MSM

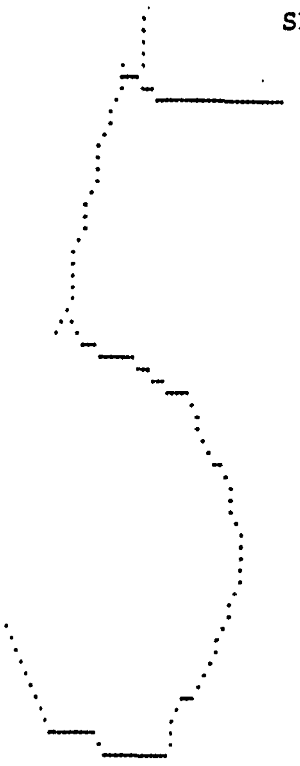

MCM

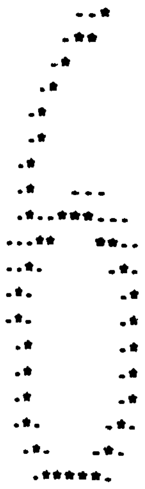

CPM

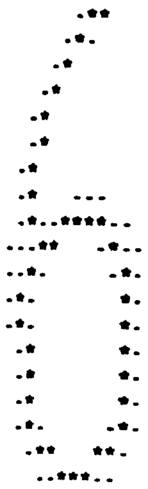

DT

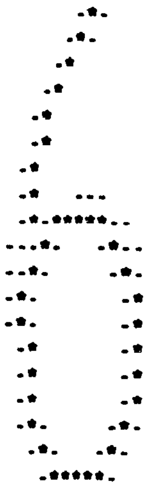

SPTA

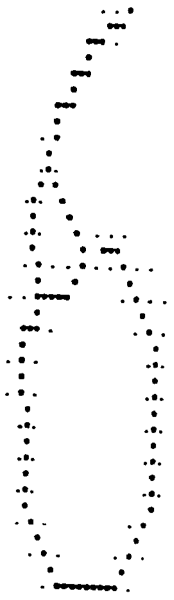

MSM

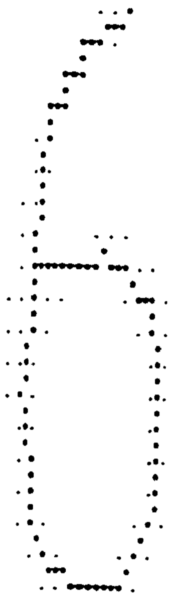

MCM

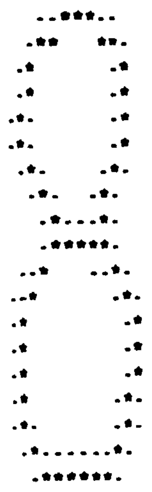

CPM

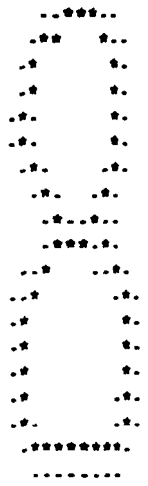

DT

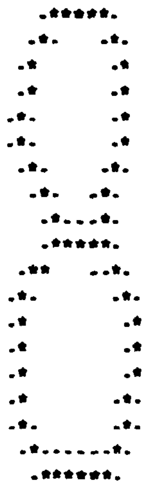

SPTA

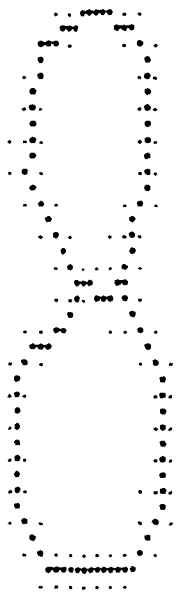

MSM

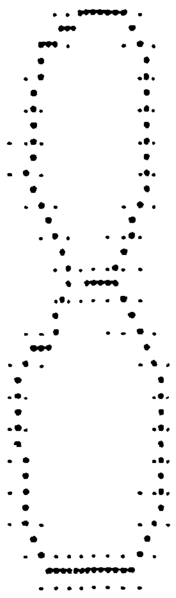

MCM

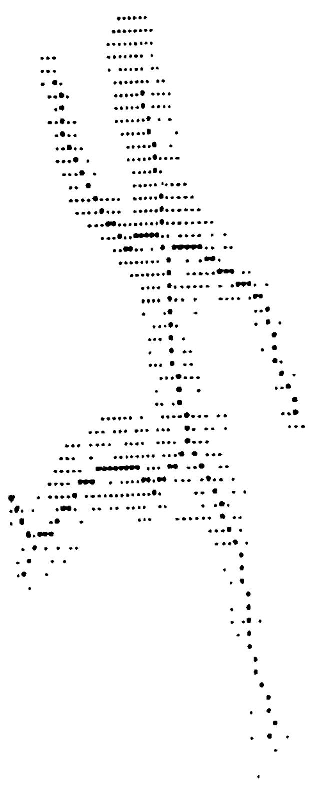

CPM

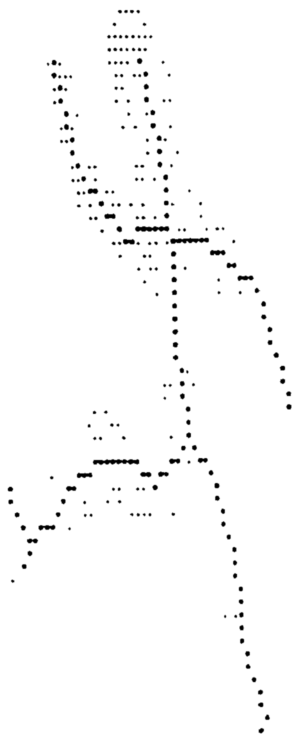

DT

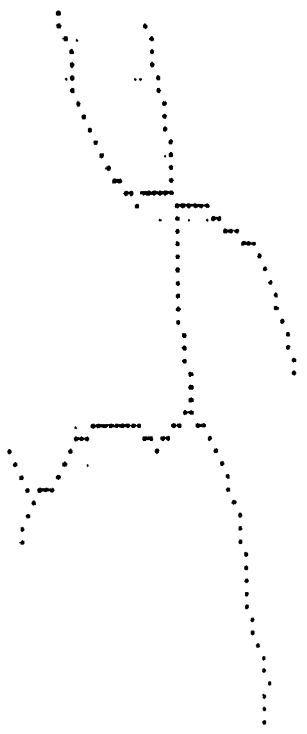

SPTA

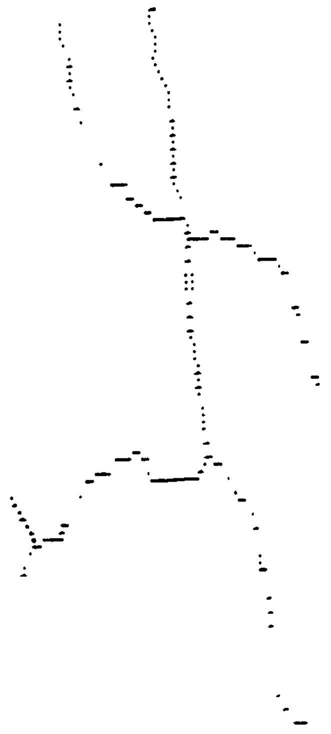

MSM

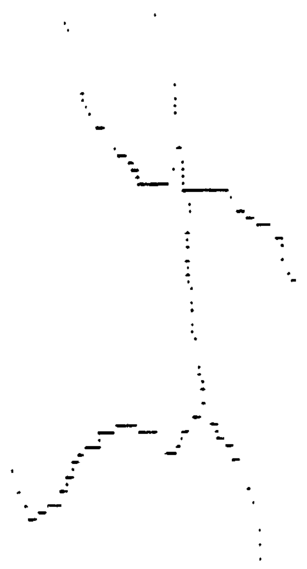

MCM

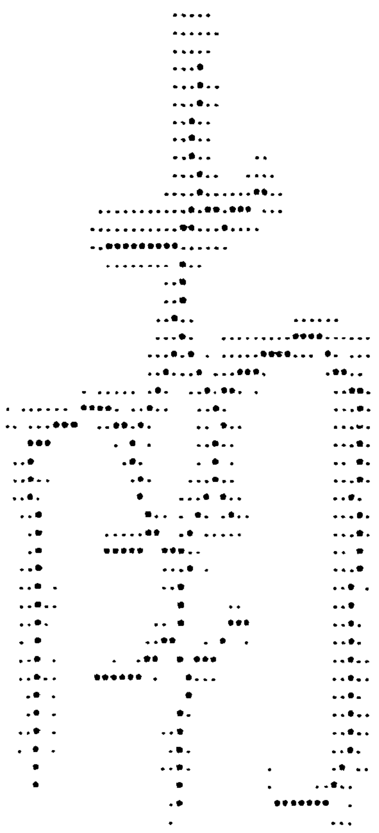

CPM

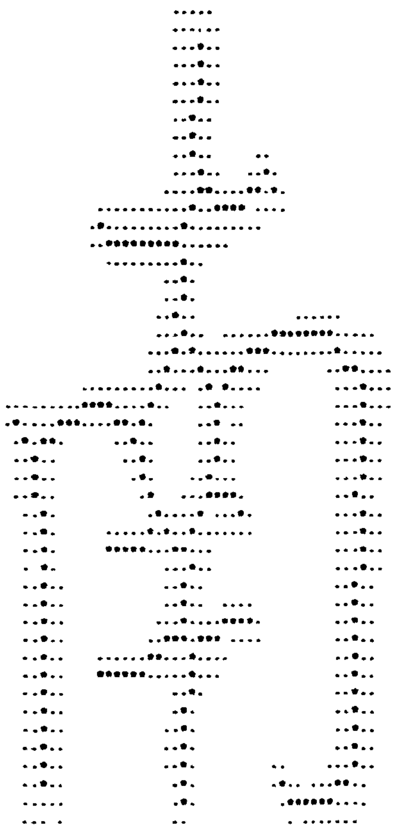

DT

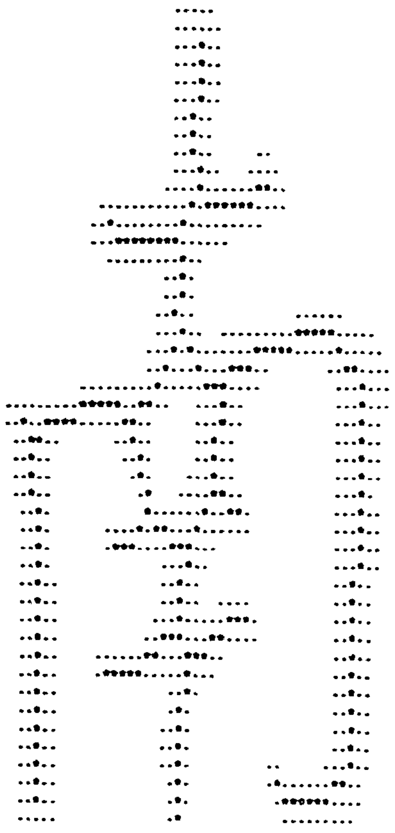

SPTA

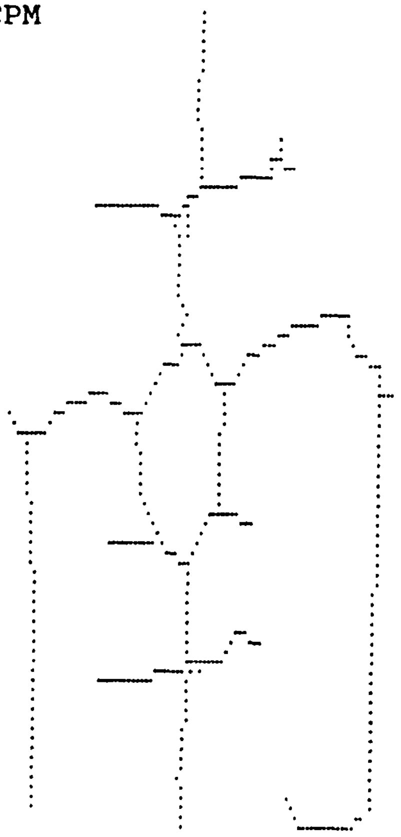

MSM

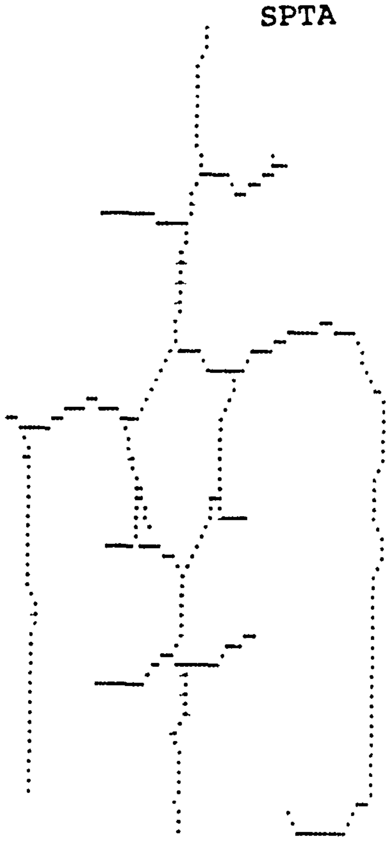

MCM

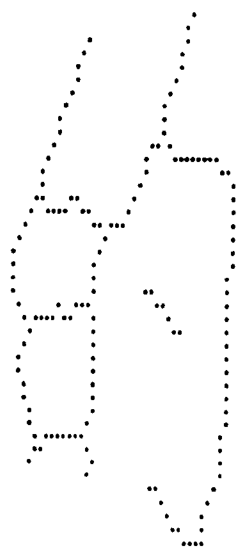

CPM

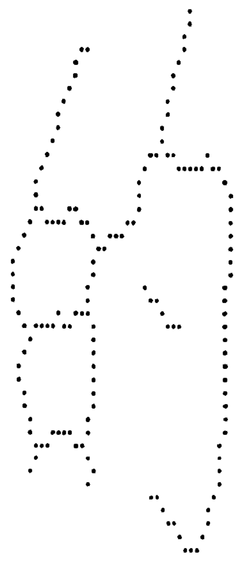

DT

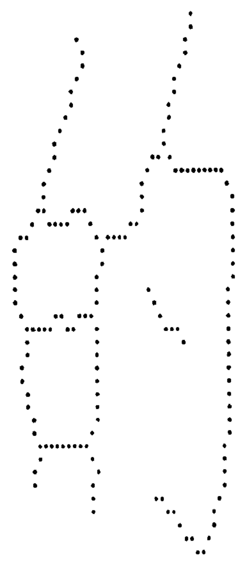

SPTA

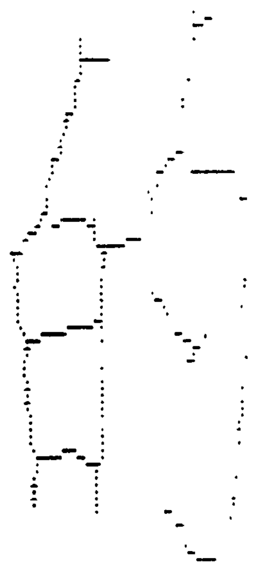

MSM

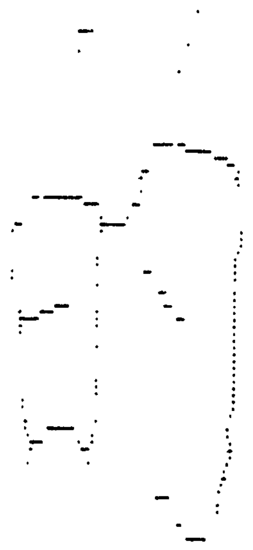

MCM

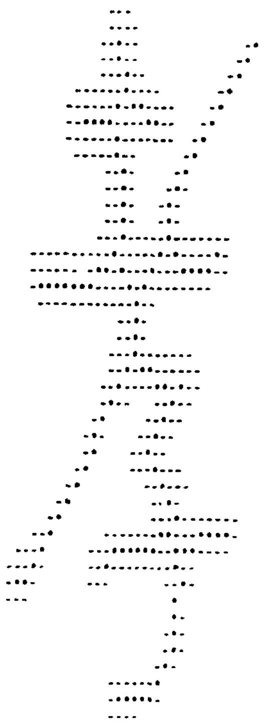

CPM


DT

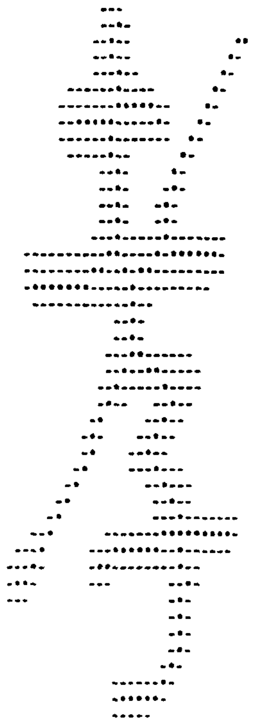

SPTA


MSM

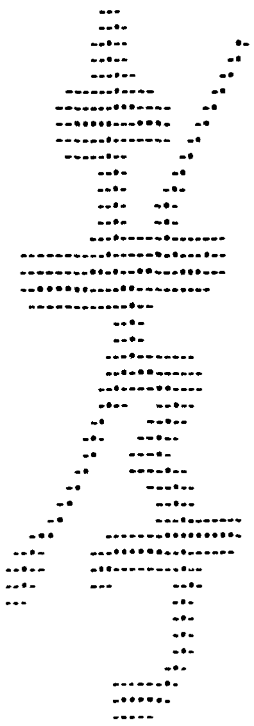

MCM



CPM

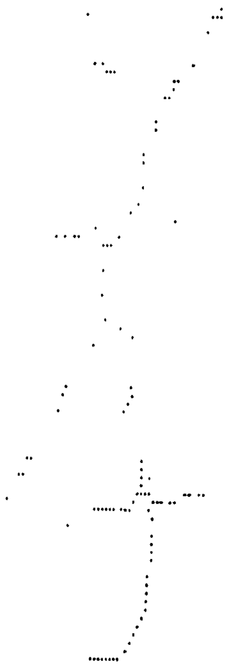


DT



SPTA

MSM



MCM