ADAPTIVE UNIFIED THINNING ALGORITHM

GEORGE J. NASSAR

A THESIS

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE AT

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

AUGUST 1996

ISBN 0-612-18421-8

ABSTRACT

Adaptive Unified Thinning Algorithm

George J Nassar

Thinning involves removing points from the contour of a pattern until all the lines and curves are a single pixel wide The simultaneous and successive removals of contour points are accomplished by Parallel Algorithms Parallel algorithms that do one pass for each successive contour, use windows larger than 3 x 3 in order to enlarge the horizon and not break the pattern.

Table Mapping is a technique used in thinning Whether a pixel (p) is deleted or not depends on the configuration of pixels in a local neighborhood containing (p). The cells of the neighborhood, in a 3 *x* 3 window centered at (p), are numbered (1, 2, 4, 8, 16, 32, 64, 128) successively The numbers are added for those cells having a black pixel A binary table is constructed where 1 means delete the pixel, and 0 means keep it The size of the table is the number of different configurations

This thesis proposes a way to construct a one-pass parallel algorithm using only 3 *x* 3 windows and the table mapping technique, but inserting in the table any unsigned integer instead of 0 and 1 only. These numbers point to different rules that direct further simple processing, thus enlarging the limiting horizon of 3 *x* 3 windows

It turns out that all parallel algorithms can be expressed in this method and compared using the same parameters, and that the only difference between them is in the number and kind of templates used which affect the rules to enlarge the horizon

Using a new labeling technique, a systematic approach is discussed to segment templates into Inner, Break and Delete Points. From this discussion, a foundation is laid for rules to be constructed and adapted to different kinds of patterns, and we consider whether there is a possibility to find one definitive set of templates and rules for each kind of patterns.

Acknowledgments

I cannot find the words to express my gratitude and respect to my supervisor Dr. Ching Y. Suen for his knowledge and guidance throughout the preparation of this thesis. The well-equipped lab and friendly environment at CENPARMI (one of the greatest centers of research in the world, founded by Dr. Suen) helped me in carrying out my thesis.

I am indebted to Dr. Louisa Lam for providing me some of the reference papers

The proof-reading was carried out by Mr. Alan Bloch

I am very grateful to Mr. Nick Strathy for his hints while typing the manuscript of this thesis.

Thanks to all the people at CENPARMI, staff and colleagues, for the friendly atmosphere.

Dedication

This thesis is dedicated to

My mother

My wife Fida

My daughter Maya

My newly born son Mark

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Objective:

Data Reduction of Patterns

One important aim of modern technology is to turn computers into intelligent machines, so that among other things they can analyze documents and recognize their contents. The amount of information contained in the documents should be reduced to the minimum necessary for the recognition of the patterns The earliest attempts date back to the 1950's on character patterns; the thinned characters were the input to other recognition algorithms [1].

Many algorithms have been devised to compress data by thinning, and applied to a variety of patterns in different fields Thinning algorithms are applied to fingerprint recognition [38], quantitative metallography [39], and measurements of soil cracking patterns [42], and to analyze chromosomes [19] and coronary arteries [41]

The need to minimize the amount of data to process shows the usefulness of thinning algorithms; recognition is easier on line-like patterns, by extraction of critical features such as end points, junction points and connections among the components [1]

Although, there are other methods for compressing data, this discussion is limited to data compression by THINNING.

1.2 The Means:

Thinning Algorithms

1.2.1 Overview

Thinning algorithms should ideally compress data, maintain connectivity, retain significant features of the pattern, and eliminate local noise without introducing new noise, the resultant skeleton must be topologically equivalent to the original object shape A good result is the medial axis of the pattern The algorithm should be fast and efficient.

The design of thinning algorithms has been and still is a very active research area. Hundreds of algorithms have been proposed employing different methodologies, subdivided into two categories: Iterative and Non-Iterative methods

The Non-Iterative algorithms produce a certain medial line by one pass, without examining all the pixels. The center lines are determined by medial-line transforms, contour tracking , line segment extraction , *etc*

In the Iterative Methods, successive layers of pixels on the boundary of the pattern are deleted until only a skeleton remains. Whether a pixel (p) is deleted or not depends on the configuration of pixels in a local neighborhood containing (p). This neighborhood could be any k *x* m window for ($k \geq 3$, $m \geq 3$). The iterative algorithms are classified as sequential or parallel depending on the way they examine pixels A sequential algorithm works in a predetermined order, and the removal decision depends on the result obtained so far in the current iteration as well as those of the previous iterations On the other hand, a parallel algorithm processes all contour pixels simultaneously, and the removal decision depends on the result of the previous iterations only

The most popular techniques implemented by iterative thinning algorithms are:

1 Hardware (Combinational Logic):
   Template Matching

2 Software:
   2.1 Template Matching
   2.2 Formulas
   2.3 Table Mapping

1.2.2 New Method

Using only 3 x 3 windows in parallel algorithms may introduce difficulty in preserving connectivity. To overcome this problem, iterations are divided into sub-iterations where only a subset of contour pixels is examined, or information from a window larger than 3 x 3 is employed.

A technique called table mapping or table look-up was used as an alternative to template matching [2, 12].

A modified technique of table mapping will be proposed that relies on inserting integer numbers in the table, instead of 0 and 1, which point to simple rules to be subsequently processed. A one-pass parallel algorithm can then be expressed by this modified table mapping technique. In addition, all parallel algorithms can be expressed in this method using only 3 x 3 support. A comparison between them can be made using the same parameters, showing shortcomings and advantages, and indicates that the only difference between them is in the number and kind of templates used which affect the rules to enlarge the horizon. Using a new labeling technique, a systematic approach will follow which classifies configurations into inner, break, and delete points. From this discussion the foundation is laid to construct and adapt different rules to different kinds of patterns. It also points out the possibility of finding one definitive and unique set of templates and rules for each kind of patterns.

The rest of the thesis is structured as follows:

Chapter 2: The input and output of thinning are discussed. It describes what kind of documents can be thinned by the proposed method and what are the characteristics of the skeleton.

Chapter 3: It desribes how the pattern is represented in memory, the masks used to test pixels, and the method of operation of thinning algorithms.

Chapter 4: The modified table mapping technique (MTMT) is described and implemented, simulating the Wu and Tsai Template Matching Method. Justification of the modified technique is presented.

Chapter 5: Unification· Some parallel algorithms are expressed in the new method. Comparison is made using simple parameters. Improvements to existing algorithms are presented and discussed.

Chapter 6: Adaptation: Labeling is introduced to aid the discussion A systematic approach for segmenting the templates is proposed A possibility to adapt one definitive and unique set of templates and rules to each kind of patterns is visualized.

Chapter 7: Conclusion.

Chapter 2

Data Reduction

2.1 The Input Domain

What are we going to thin?

The thinning phase lies in the preprocessing level of a 4-level document analysis process (see Fig. 1).

In the first level lies the digitization phase. Most printed documents whether a picture, a drawing or simply text, are composed of tiny dots. A fine, clear picture will have more dots squeezed in an inch than a coarse one. While humans prefer looking at pictures of higher resolution, the computer prefers to minimize the storage of pixels.

A scanner or digital video will transform a document into a file of picture elements, called *pixels*, that are sampled in a grid pattern throughout the document. These strings of pixels, usually in the form of a matrix, will be the input to the thinning phase.

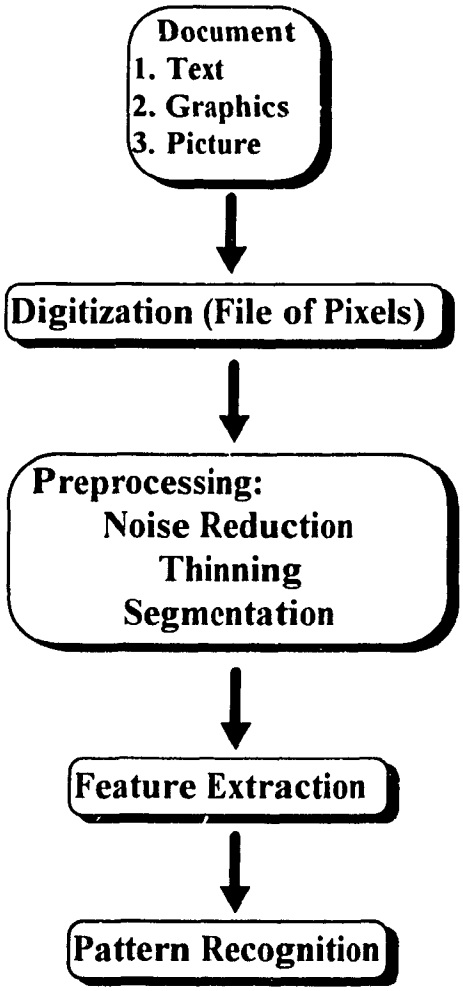


Figure 1· Steps in document analysis.

These pixels may have values:

- 0 or 1 for binary images.
- 0 to 255 for gray-scale images.
- three channels of 0 to 255 color values for color images

Printing technology advances rapidly, which means higher resolution and more pixels to process. To get an idea of the magnitude of pixels to process, an 8.5" x 11" page sampled at 300 dots per inch would yield an image of 2,550 pixels x 3,300 pixels or 8,415,000 pixels per page.

Binarization by a scanner, *i.e.*, converting a gray-scale image into a black and white one, results sometimes in disconnected lines. A gray-scale thinning algorithm can extract thin lines and curves directly from gray-scale images, by tracking along gray-level ridges whose peak intensities vary throughout the image Problems arise by following false tracks, which require computationally expensive backtracking [44].

The modified algorithm in this thesis is not intended for gray-scale thinning

Thinning discussed in this thesis will be limited to binary documents, documents consisting of black as foreground and white as background. Color documents are converted to gray-scale, and gray-scale documents are converted to binary documents. As shown in Fig. 2, the right-hand image in binary form was converted from the gray-scale image on the left by a scanner.

Thinning will be limited to text and graphics. As shown in Fig 3, the objects on the right are thinned from the objects on the left.

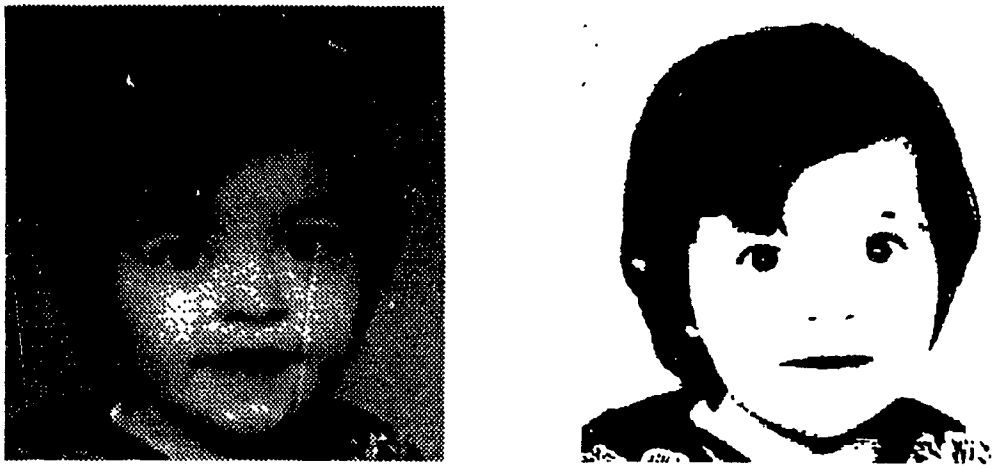


Figure 2: Gray-scale picture binarized by a scanner.

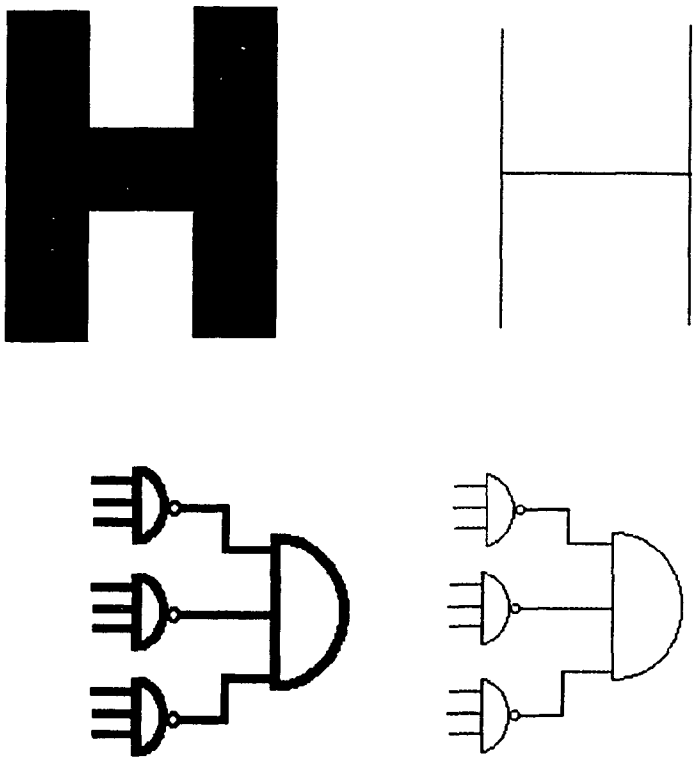


Figure 3: Text and graphics thinned by software.

2.2 The Output

What is the ***Result*** of thinning?

The object is defined as a binary document of text and graphics.

The aim is to reduce the data contents in the document by turning foreground pixels into background.

The result of thinning is called the skeleton, a one-pixel wide figure derived from a pattern.

The lack of an exact definition of skeletons is one reason for the diversity of thinning algorithms. A skeleton should have in general the following characteristics [21]:

(1) Connected pattern regions must thin to connected line structures. If the object is connected, the resulting skeleton should also be connected. If the initial background is connected, the background resulting from thinning should also be connected. If the background is not connected, the background after thinning should not be connected.

(2) Excessive erosion should be prevented; the end points of the object should be detected as soon as possible, as they represent true features to be used for further image analysis.

(3) It should be noise-insensitive. Noise, or small convexities which do not belong to the skeleton, appearing often as a tail after thinning, should be minimized.

(4) It must be topologically equivalent to the original pattern. It should approximate the medial lines.

Chapter 3

Thinning Algorithms

3.1 Overview

Thinning is a process to turn foreground points into background points, until a one-pixel wide line is formed.

The algorithm should be fast and efficient.

Speed can be measured (*ex*: milliseconds or microseconds).

There are ways to achieve this; for example:

- Limit the scanning to contour pixels only.
- Generate the inner contour from the outer contour.
- Make the tests for pixel removal as simple as possible.

Efficiency, on the other hand, has no precise definition, but can be defined by:

- Connectivity.
- Medial line.
- Retaining end points without introducing distortions.

Another technique proposed in [8, 47] to test efficiency consists of a set of alphanumeric characters selected on their main structural properties and other types of lines. These characters are thinned manually by a group of people. The resulting skeletons, according to the tests conducted, prove to be references to evaluate the efficiency of thinning algorithms. A protocol is suggested by which reference skeletons specific to the application under study should be done by hand.

3.2 Image Formats, Masks and Terminology

The thinning process may be done by special-purpose hardware (Combinational Logic) or by software. The pixels of a digitized document can be represented and scanned in the computer memory or in special hardware in different ways

The Image-format or bitmap is the format used by the computer to represent the pattern.
A Mask is a window used to view a small portion of the bitmap

3.2.1 Special-Purpose Hardware

The image-format consists of 3 scan lines stored in a special kind of memory. At each clock cycle one pixel leaves this serpentine memory structure and another one enters from scanners or files [5].

Each logic gate represents one thinning template or a group of thinning templates.

3.2.1.1 Mask

The mask is a small portion of the serpentine memory connected to logic gates.

3.2.1.2 Terminology

- Clock cycle is the hardware beat at which pixels are shifted in the design.

- Cascading stages: Each stage consists of a special kind of memory and combinational logic circuit (logic gates). We cascade different stages by connecting the output of one stage to the input of another stage. The thinning process is performed by a number of cascading stages.

3.2.2 Matrix form

A binary digitized picture is defined by a matrix M of size s *x* t, where each pixel M(i, j) for ($1 \leq i \leq s$, $1 \leq j \leq t$) is either 1 or 0.

Computer memories consist of bytes. Each byte consists of 8 bits. Each bit stores a binary digit (1 or 0). Memory chips exchange data by bytes. The image-format consists of a group of bytes forming a matrix. A byte represents either one pixel or a group of 8 pixels (8 bits). Working with bits saves memory storage. Besides that, both representations are used depending on efficiency, low-level programming, and practicality.

3.2.2.1 Mask

The Mask is a (k, m) matrix for ($3 \leq k \leq s$, $3 \leq m \leq t$). For practical purposes (§ 4.2.1) k and m are less than 5. One cell of the mask, termed (p), is the center cell of an odd-sized square mask, and could be any cell in even-sized masks. The other cells in the mask are called the neighbors of (p). (p) highlights some position i ,Mj), the other cells in the mask are then at positions relative to (p)· (i, j), (i-1, j), (i-1, j+1), (i, j+1), *etc.* (see Fig 4)

The mask moves across the bitmap. At each step it highlights a different set of pixels. This set of pixels or configuration is called a template.

Different notations are used to address the cells in the mask other than by row and column, *e.g.*, starting counterclockwise from (i, j+1), the cells are labelled: X_1 , X_2 , ..., X_8

i-1, j-1	i-1, j	i-1, j+1
i, j-1	(p) i, j	i, j+1
i+1, j-1	i+1, j	i+1, j+1

x_4	x_3	x_2
x_5	(p)	x_1
x_6	x_7	x_8

NW	N	NE
W	(p)	E
SW	S	SE

Figure 4 3 *x* 3 mask with different notations

The mask highlights binary numbers, and is denoted a BINARY mask We convert a binary mask into an INTEGER MASK by assigning some set of different integer digits to the cells in the mask. Integer digits replace the 1-pixels The 0-pixels remain the same. For example:

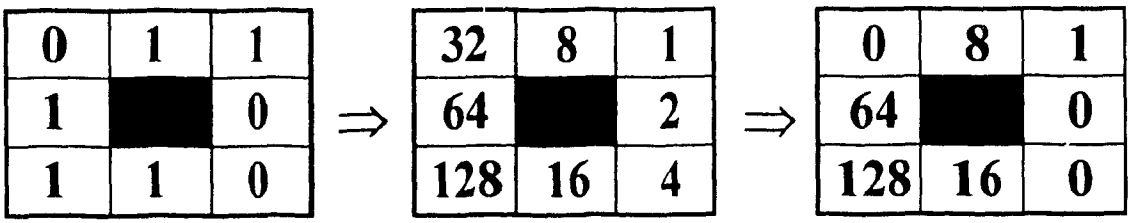


Figure 5· Binary and integer masks

3.2.2.2 Terminology

Overview:

The pixel examined for deletion is (p).

Pixels x_1 , x_2 , . , x_8 are the 8-neighbors of (p), and are collectively denoted by N(p). Pixels x_1 , x_3 , x_5 , x_7 are the 4-neighbors of (p), or its D-neighbors. Pixels x_2, x_4 , x_6 , x_8 are the I-neighbors of (p). x_i denotes the pixel and its value 0 or 1 The number of black pixels in N(p) is denoted by b(p).

If p_o and p_m are two (dark) pixels that belong to the same pattern, there exits a path, which can be described as a series of dark points p_o, p_1, , p_m with each consecutive pair of pixels, p_i and p_{i+1} (i=0, 1,... m-1), being neighbors of each other If all eight neighbors are considered, p_o and p_m are said to be 8-connected If only the 4-neighbors are considered, p_o and p_m are said to be 4-connected.

An object is 8-connected or 4-connected if all pairs of pixels in the object are 8-connected or 4-connected respectively.

A pixel (p) is 8-deletable or 4-deletable if its removal does not change the 8-connectivity or 4-connectivity respectively of the object

The choice that the pattern is 8-connected and the background is 4-connected ensures a unit width skeleton

The pixels on the first layer of the pattern (P), *i.e.*, those adjacent to the background (B), are said to be on the contour. As shown in Fig. 6, (p) is 8-adjacent to B, and (p_1) is 4-adjacent to B.

It will suffice to test for removal only pixels on the contour that are 4-adjacent to the background to ensure 8-connectivity for the foreground and 4-connectivity to the background. In this case contour points are those 1-pixels having at least one 4-adjacent 0-pixel.

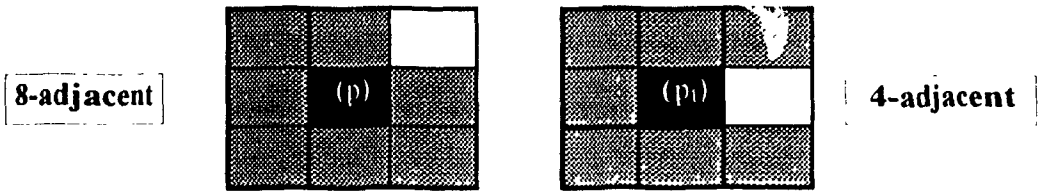


Figure 6: Different contour points.

3.2.3 Chain Codes

Chain codes are a list of 4 or 8-connected black pixels along lines and curves chained together and coded in some manner The list is derived from the bitmap by raster-scanning Some applications code the entire image into chain codes and disregard the matrix [21], while other applications keep the matrix and generate lists for successive contours [31]

Engineering drawings, for example, contain many background pixels Representing only the foreground pixels saves memory storage and increases processing speed

A list of black pixels, connected or not, is termed a queue, and is used to speed processing in some applications [23] The absolute locations of pixels in the bitmap are stored in the list

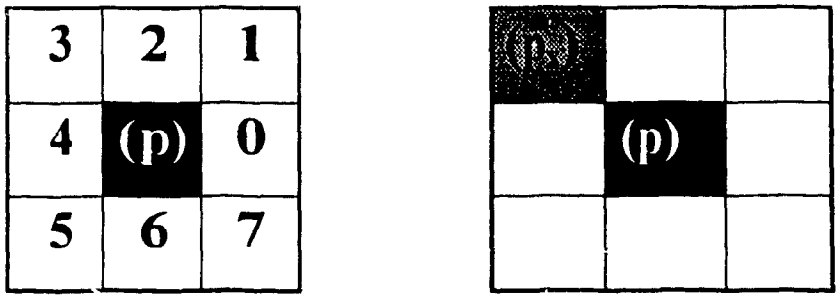
Depending on the code used to denote the pixels in the chain, information on connectedness can be derived to facilitate further image analysis

One popular code is that of Freeman's It encodes the direction to the next 8-connected pixel from a given pixel in a sequence (see Fig 7)

Chain coding will not be relevant in any way to our modified algorithm Only queuing is considered as a way to speed processing

Chain coding is used in such techniques as contour tracing and contour generating algorithms, that relies on other testing conditions than a mask for pixel removal

3.2.3.1 Mask



Freeman codes $(p_x) = 3$

Figure 7. Freeman codes.

3.2.3.2 Terminology

The pixels that are directly before and after a pixel (p) in the chain are its C-neighbors.

Most of the terminology follows that of matrix form.

3.3 Thinning Methods

Hundreds of papers already have been published on thinning. To discuss each and every one is not the purpose of this thesis. Many surveys exist on the matter; the latest comprehensive survey can be found in [1]. Here we will review the modes of operation and the techniques for pixel-removal testing to produce the skeleton.

Iterative methods process the pattern layer by layer and pixel by pixel.

Non-iterative methods produce a mid-line or a skeleton without examining all the pixels.

3.3.1 Iterative methods

Most thinning algorithms are iterative. In each iteration, each pixel in a contour is examined against a set of rules to decide on its removal. According to how they scan the bitmap, iterative algorithms are either sequential or parallel.

3.3.1.1 Sequential algorithms

The mask contains pixels already processed (*i.e.*, their state could have been changed) from the present contour, as well as from different contours and the background. These algorithms scan the bitmap in a predetermined order; the skeleton depends on this order of scanning.

The image-format could be a matrix or a list of chain codes.

3.3.1.1.1 Matrix form

The usual mask used is a 3 *x* 3 window.

In [27] a (k *x* k) mask was used. The center core of (k - 2) *x* (k - 2) pixels is deleted if the boundary pixels in the window have Hilditch [19] crossing number 1 and if they contain more than (k - 2) black pixels and more than (k - 2) 4-connected white pixels. The (k *x* k) masks are applied in decreasing order until k < 3 or the core is deleted.

Hilditch [19] and Rutovitz [9] crossing numbers are used to ensure connectivity. Many algorithms use one of these two numbers. They differ in the other tests proposed to retain end points or be noise insensitive.

Other group of algorithms use formulas to express templates [20]; the difference is in the number and kind of templates.

Techniques used to speed processing include pipeline structure [40], buffering [24], or queuing the pixels [23].

3.3.1.1.2 Chain codes

A speed-up can be achieved by considering only contour points.

The technique used is to trace the contour using chain codes. The skeleton is formed by deleting simple points, and retaining multiple points as well as some simple points to ensure connectedness [29, 31, 37], .

A pixel (p) is multiple if at least one of the following conditions is true:

- (p) is traversed more then once during tracing. (connectivity)
- (p) has no neighbors in the interior. (end points)
- (p) has at least one 4-neighbor that belongs to the contour but is not traced immediately before or after (p). (two pixel wide lines)

Contour generation [21], another technique, traces the contour once. It generates the next contours from the stored one.

3.3.1.2 Parallel algorithms

Parallel algorithms process and update all the pixels in the contour simultaneously. For this reason, these algorithms are suitable for implementation on parallel processors.

Rutovitz proposed a one-pass parallel algorithm which uses a 4 *x* 4 mask.

The Rutovitz one-pass, parallel, and 4 *x* 4 mask algorithm was coded as follows:

(1) $X_R(p) = \sum_{i=1}^{8} | X_{i+1} - X_i | = 2.$

(2) $b(p) \geq 2$

(3) $x_1 x_3 x_5 = 0$ or $X_R(x_3) \neq 2$

(4) $x_7 x_1 x_3 = 0$ or $X_R(x_1) \neq 2$

Rule (1) is to simulate the set of templates that preserve 4-connectivity.

Rule (2) is to exclude templates consisting of 1 black neighbor.

Rules (3) and (4), are to avoid 2-pixel wide erosion.

Most parallel algorithms based on formulas use the same principles as found in the Rutovitz algorithm.

These algorithms differ by adding or removing some rules, or by modifying the original rules for different purposes, *ex*:

- use a 3 *x* 3 mask only, and in this case use 2 sub-iterations [10].
- avoid excessive erosion and reduce 2-pixel diagonal lines [12].

Some one-pass parallel algorithms [3, 5] use template matching on special hardware using 3 *x* 4 and 4 *x* 3 windows.

3.3.2 Non-iterative methods

These methods do not examine all the pixels in the bitmap iteratively to produce a skeleton. The methods include:

- Distance Transforms:

The distance transform is a labeling technique where each pixel is labeled by the shortest distance from it to the boundary of the region within which it is contained. The skeleton is obtained by retaining only ridges of maximum local-distance measures [28, 43].

- Extraction of line segments

Connecting strokes having certain orientations are used to obtain approximations of skeletons. For example, vertical, horizontal, left diagonal, and right diagonal limbs are detected and then heuristically connected [32].

- Contour tracking methods

The center line is detected by tracking the two contours of each curve simultaneously by maintaining a minimum distance [34, 35].

3.4 Testing rules of iterative methods

The reason to have a section on the testing rules of iterative methods that use raster scanning is to compare the implementations of different algorithms and justify the modified algorithm.

3.4.1 Template Matching

A subset of (k *x* m) templates is used as a thinning criterion and stored in memory.

A mask moves across the bitmap. The mask is compared to the set of thinning templates, and a pixel (p) is deleted if its mask satisfies any one template in the set.

This is a low-speed process if done in software.

Template matching is implemented in special-purpose hardware by logic gates.

3.4.2 Formulas

Formulas are used to simulate a certain set of templates, *i.e.*, to code different templates into arithmetic or logical formulas.

This approach is faster than template matching.

Hilditch coded the templates that satisfy 8-connectivity by:

$$X_H(p) = \sum_{i=1}^{4} b_i = 1$$

$$\text{where} \quad b_i = \begin{cases} 1 \text{ if } x_{2i-1} = 0 \text{ and } (x_{2i} = 1 \text{ or } x_{2i+1} = 1) \\ 0 \text{ otherwise} \end{cases}$$

3.4.3 Table mapping

Table mapping is an efficient alternative to template matching.

We use an INTEGER mask (§ 3.2.2.1) to add the cells together to produce one number in the range $(0..2^{k*m-1} - 1)$. A table is constructed to accommodate the different templates. The subset of templates that are used for pixel removal will have 1 inserted in the corresponding cells in the table. Other cells of the table have value zero.

The addition of the cells and the table look-up steps replace the comparison made on the set of templates on a one-by-one basis.

Chapter 4

New Method

4.1 Motivation

Wu and Tsai [3] implemented a new one-pass parallel thinning algorithm for binary images. In the conclusion of their paper, they claim that this algorithm produces perfectly 8-connected, noise-insensitive, excessive erosion free, and isotropic skeletons

Well, at last a perfect algorithm?

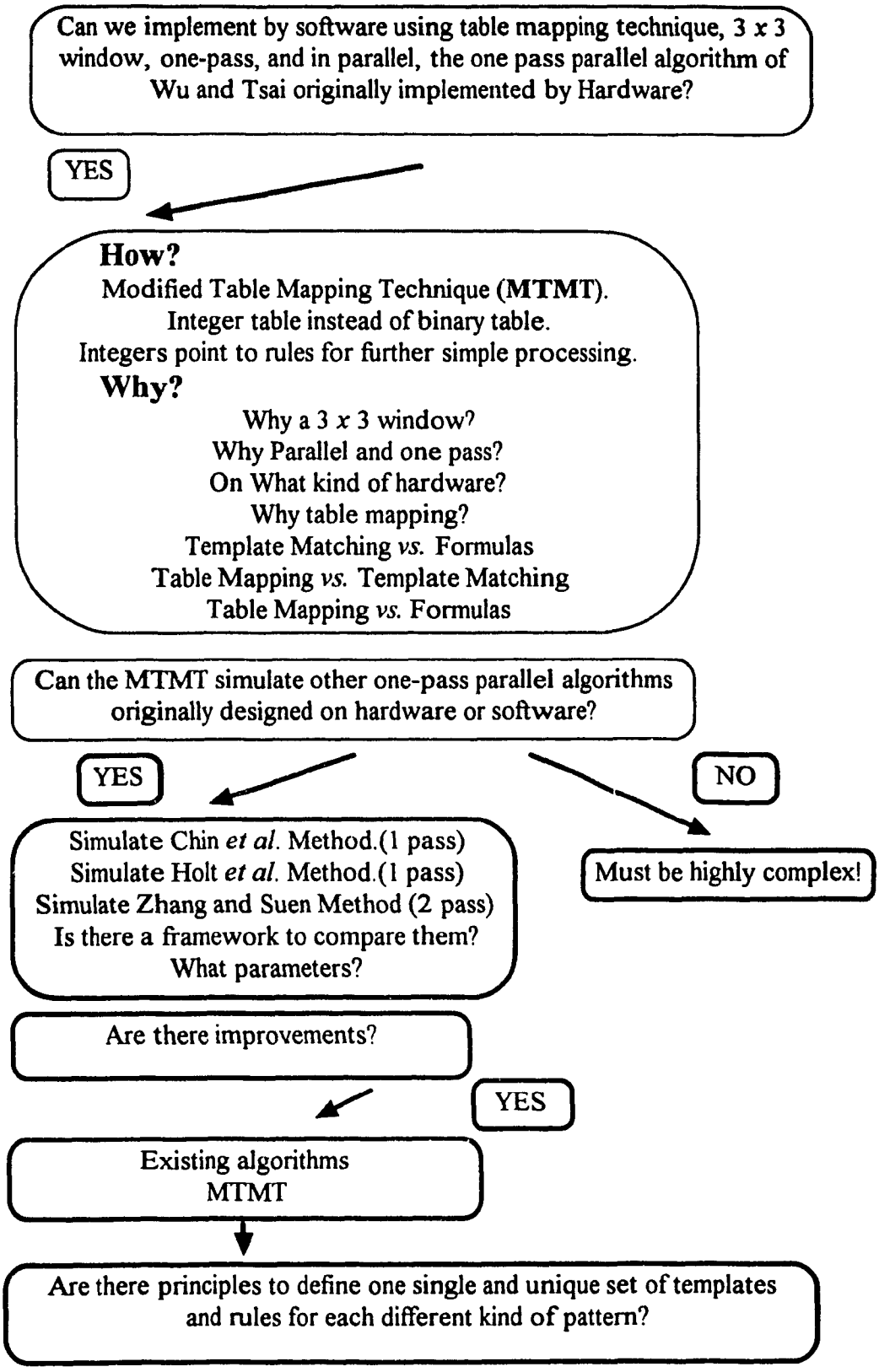
Hundreds of papers have been proposed already and most of them claim the same qualities the skeleton has, or at least some of these qualities. We are somewhat baffled by this proliferation of algorithms. Each year, new papers are published proposing new methods. Even I am proposing a new method!

The following questions will usually pop to our mind.

- Is it worth the time and pain to make improvements to thinning algorithms?
- How and in what do the algorithms differ?
- On which computer would we implement the algorithm, a special dedicated computer or on a general purpose computer? Can we implement it on both?
- Which algorithm is the best? How we can compare different algorithms: on what basis? what parameters?
- When is this flood of new algorithms going to stop?

Before I try to answer these questions, let me propose my technique as a framework for the discussion.

On the next page a flowchart summarizes what will follow:


Can we implement by software using table mapping technique, 3 x 3 window, one-pass, and in parallel, the one pass parallel algorithm of Wu and Tsai originally implemented by Hardware?
YES
How?
Modified Table Mapping Technique (MTMT).
Integer table instead of binary table.
Integers point to rules for further simple processing.
Why?
Why a 3 x 3 window?
Why Parallel and one pass?
On What kind of hardware?
Why table mapping?
Template Matching vs. Formulas
Table Mapping vs. Template Matching
Table Mapping vs. Formulas
Can the MTMT simulate other one-pass parallel algorithms originally designed on hardware or software?
YES
NO
Simulate Chin et al. Method.(1 pass)
Simulate Holt et al. Method.(1 pass)
Simulate Zhang and Suen Method (2 pass)
Is there a framework to compare them?
What parameters?
Must be highly complex!
Are there improvements?
YES
Existing algorithms
MTMT
Are there principles to define one single and unique set of templates and rules for each different kind of pattern?

4.2 Modified Table Mapping Technique (MTMT)

Trying to discover any improvement that could be made to existing thinning methods, my attention was drawn to the Wu and Tsai algorithm. It is a template matching algorithm. It is supposed to have good performance on special dedicated hardware.

- Why should we have algorithms for dedicated expensive hardware and not for multiprocessors or personal computers?
- Since the table mapping technique is a fast implementation of template matching, then how can we implement a one-pass parallel thinning algorithm using only 3 *x* 3 windows by table mapping ?

In an effort to answer the above questions, I propose an algorithm and later I discuss its usefulness (Fig. 9 shows the PseudoCode of the algorithm).

The **only** difference between the table mapping technique and the MTMT is in the *entries* of the table. In the table mapping technique, the entries are **binary**: 0 for keeping the pixel, and 1 for deleting it. In the modified technique, the entries can be any unsigned **integer**.

So, how does it really help?

To ensure connectivity using only 3 *x* 3 windows, Wu & Tsai used 3 *x* 4 and 4 *x* 3 windows to enlarge the horizon and see the pixels beyond the 3 *x* 3 window (see Fig. 8).

In the 3 *x* 4 window, they tested pixel **a** (i, j+2); **b** (i-1, j+2) and **c** (i+1, j+2) are don't-care pixels. In the 4 *x* 3 window they tested pixel **d** (i+2, j); **e** (i+2, j-1) and **f** (i+2, j+1) are also don't-care pixels.

In the 3 *x* 4 window, (p) is removed if (**a**) is black

In the 4 *x* 3 window, (p) is removed if (**d**) is black.

In all the other thinning templates, (p) is unconditionally deleted.

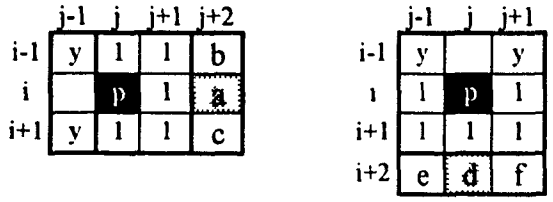


Figure 8: 3 *x* 4 and 4 *x* 3 templates used in the Wu and Tsai algorithm.

In the MTMT the unconditional templates have value 1, which means unconditionally delete the pixel; the templates testing pixel (**a**) have value 2, and the templates testing pixel (**d**) have value 3. Actually 2 and 3 are the 'case structures' in the pseudocode that simply test pixels (**a**) and (**d**) respectively.

In Fig. 9 you can find the pseudocode. The tests in the 'case structures' are simple. They test the presence of another pixel. Actually, as will be discussed later, they can test more than one pixel in order to accommodate other one-pass parallel algorithms, and can be used to define a single set of templates for each kind of pattern By testing 2 or 3 more pixels, the algorithm can stay within time constraints.

In Fig. 10 we see the original templates proposed by Wu and Tsa (1 = black pixel, void = white pixel, x = don't care, y = at least one must be white and it doesn't appear singly), and the same templates expanded by the MTMT. The templates in the last 2 rows which correspond to rules 2 and 3 are assigned values in a 3 *x* 3 window only. The extra pixels in the 3 *x* 4 and 4 *x* 3 windows will be taken care of by the rules.

In Fig. 11 we find how the table is structured. It is not feasible to put on one page 256 entries consecutively. To find the entry of a template, take its number (found beneath

it); go to the row that is less or equal to it, then subtract it from the row heading, and go that number of columns to reach the cell.

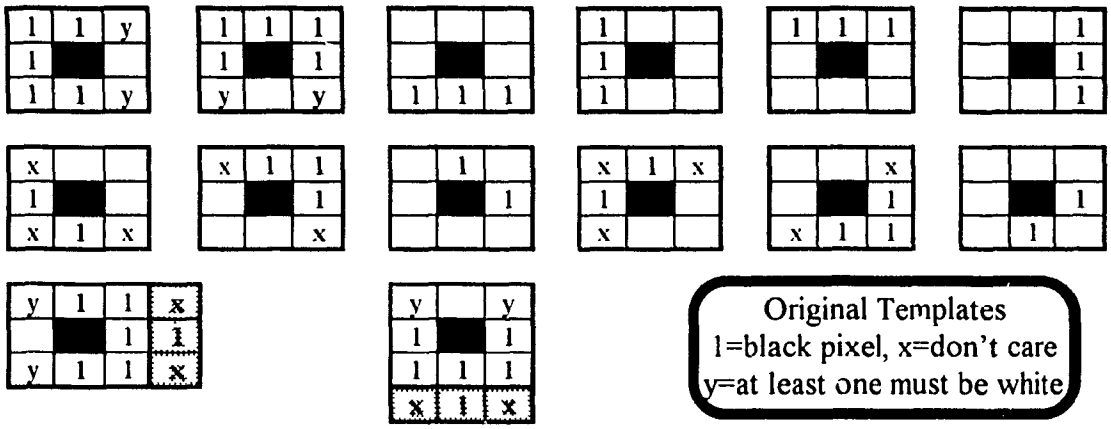
In Fig. 12 the skeletons correspond to those in the original paper of Wu and Tsai, but were produced by simulating their method using the modified table mapping algorithm.

At this stage I have shown with the modified table mapping technique that a thinning algorithm running in parallel, using only 3 x 3 windows, in one-pass per contour, is feasible, and as a test, simulated the Wu and Tsai algorithm.

In order to justify the MTMT we must consider the following issues: why a 3 x 3 window? why in parallel? on what kind of hardware? why table mapping?

```
procedure thin
  for all pixels in the bitmap do
    if black pixel do
      compute the mask number
      look up the mask number in a table and for each entry number do
        case 0  do nothing
        case 1  remove the pixel
        case 2  check if pixel (i, j+2) is black
                if black remove the pixel
        case 3  check if pixel (i+2, j) is black
                if black remove the pixel
        case 4  check if pixel (i, j+2) and pixel (i+2, j) are both black
                if both black remove the pixel
        etc. for other rules
      endtable
    endif
  endfor
endprocedure
```

Figure 9: Pseudocode of the MTMT.



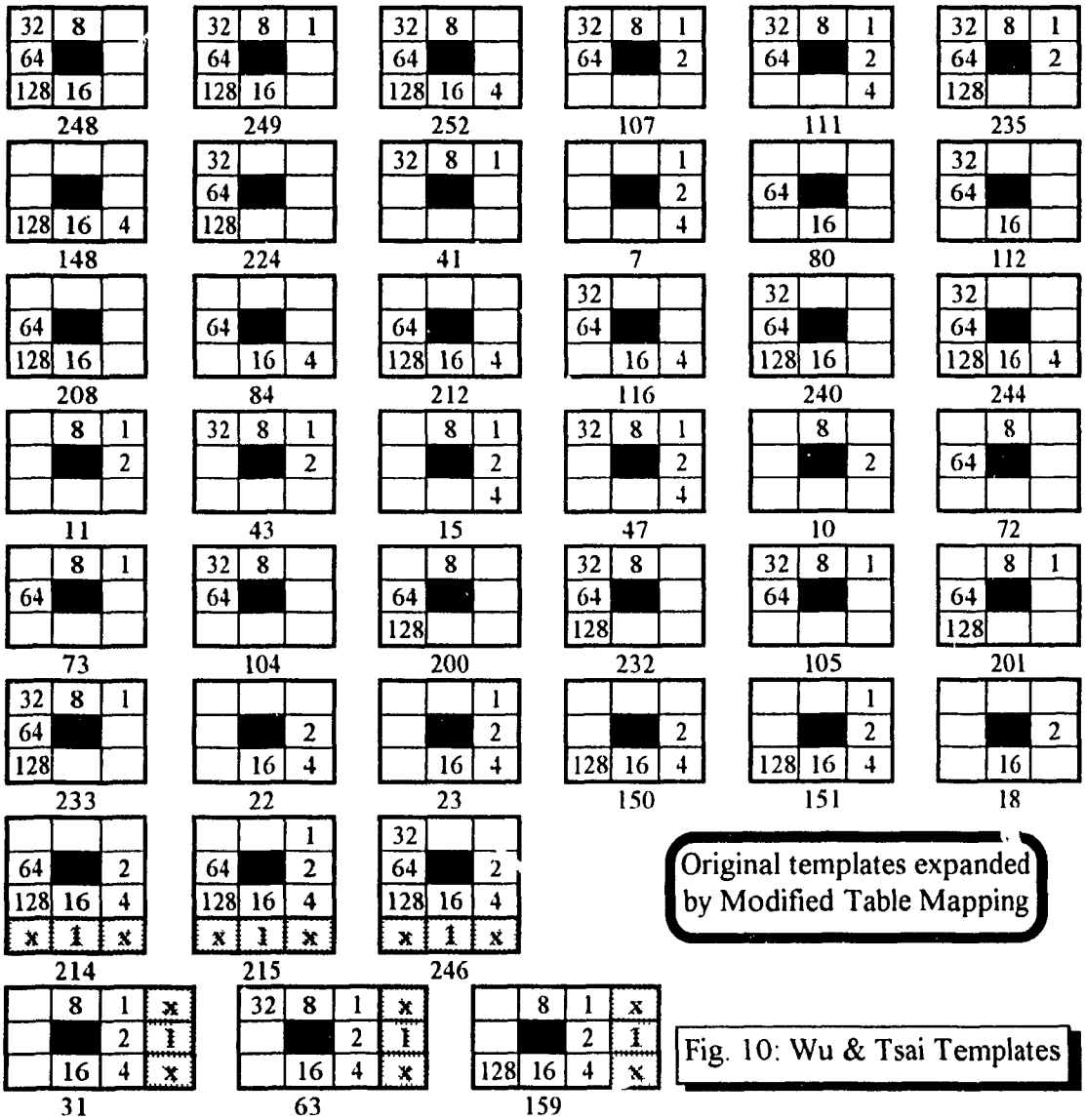


Fig. 10: Wu & Tsai Templates

32	8	1
64	p	2
128	16	4

8 - 4 - 2 - 1

16
32
64
128

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0								1			1	1				1
16			1				1	1								2
32										1		1				1
48																2
64									1	1						
80	1				1											
96									1	1		1				1
112	1				1											
128																
144					1		1	1								2
160																
176																
192									1	1						
208	1				1		3	3								
224	1								1	1		1				
240	1				1		3		1	1			1			

Figure 11: Table look-up of Wu and Tsai templates.

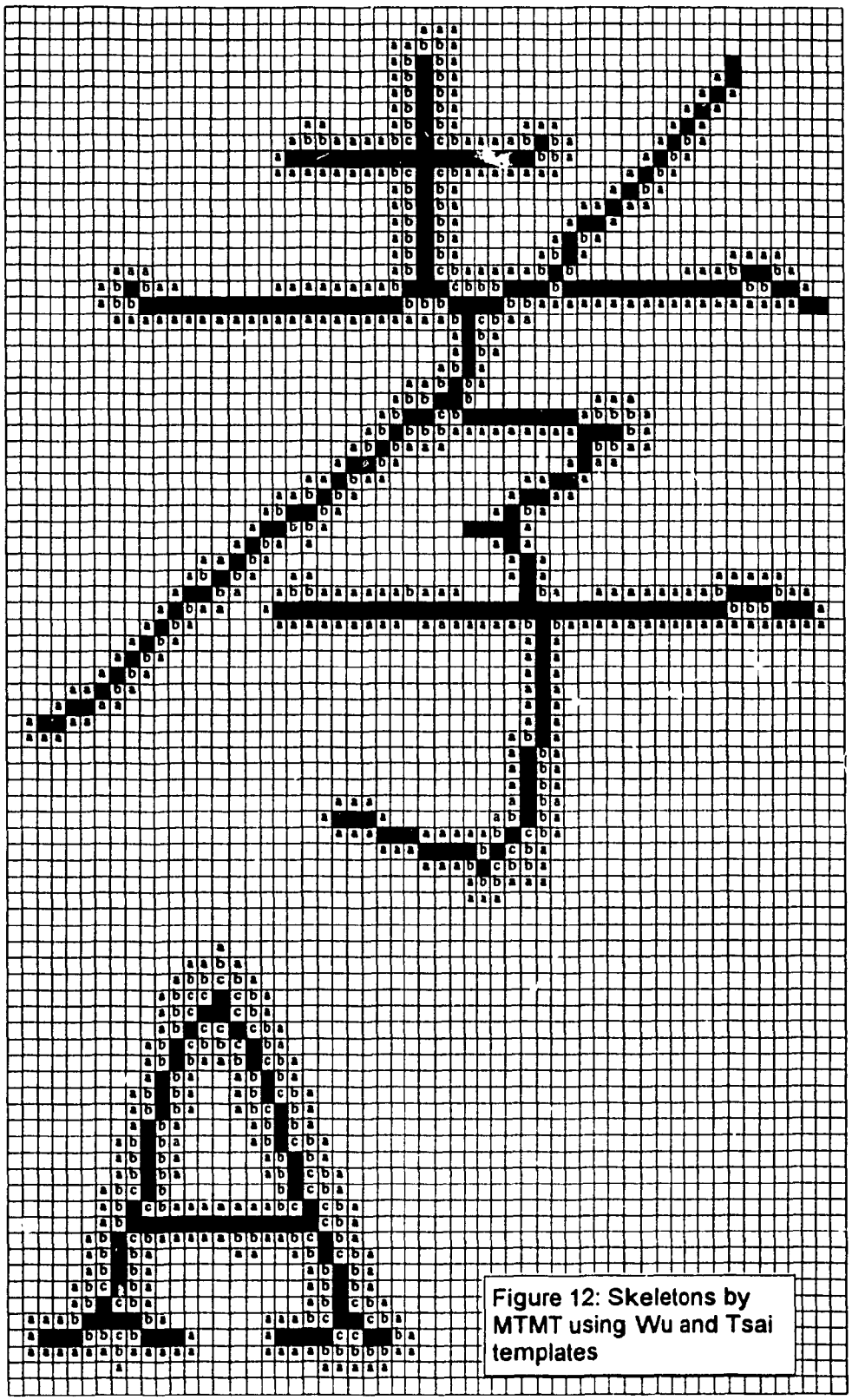


Figure 12: Skeletons by MTMT using Wu and Tsai templates

4.2.1 Why a 3x3 Window?

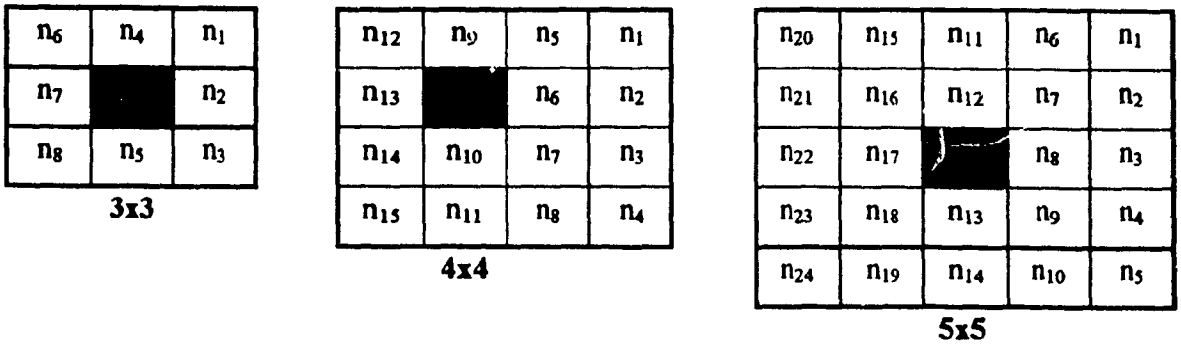


Figure 13: Different sizes of masks.

The possible combinations of a binary (k *x* k) mask are 2^{k*k-1}.

window	format	Templates
3 x 3	2^8	256
4 x 4	2^{15}	32,768
5 x 5	2^{24}	16,777,216

Table 1: Mask combinations.

Advantage of larger windows

The advantage is the enlarged horizon; more pixels can be seen in a template.

Disadvantages of larger windows

(1) In even-sized windows, *e.g.*, 4 *x* 4, the window is biased to one side, *i.e.*, you see more pixels in some directions than in other directions. You have to shift the pixel (p) to n_7 in Fig. 13 to consider the other directions. Actually it could suffice in those applications

of thinning to rely on exploiting the *bias* in only two directions, as the case of the Wu and Tsai algorithm.

(2) Memory storage: The cost of memory was and still is an expensive part of a computer. Even though the cost of memory chips has dropped drastically, they are still expensive.

(3) The number of combinations to consider: It is feasible, though time consuming, to study and segment 16,777,216 templates, but this is not the issue. In specially designed hardware the number of logic gates will increase dramatically and their cost too, while on general-purpose computers the running time will increase as we are calculating 24 cells (5 x 5 window) instead of 8 cells.

4.2.2 Why Parallel?

Sequential algorithms are implemented on one processor only. On the other hand, more processors can cooperate to perform the work faster by using parallel algorithms. Parallel algorithms are implemented on different kind of computers.

Sequential algorithms can use contour tracing and contour generating to process only contour pixels. Multi-processors and personal computers can use the technique of queuing the contour, and generating the inner contour from the actual one.

The research now is to find one-pass parallel algorithms, for the obvious reason of doing less iterations over the bitmap.

4.2.3 On what kind of hardware?

Parallel algorithms can run on any kind of computer with different speed performances.

One of the objectives of this thesis is to run parallel algorithms, that were originally designed to run on a specially designed computers, on: personal, multiprocessors, or array computers, with acceptable performance

Specially designed hardware can run in the range of nanoseconds. They simulate only one set of templates. They are not computers in the real sense. We design different logic gates for different set of templates. They are rigid structures.

Array computers are expensive, with one processor for each pixel. A document page of 2,000 x 3,000 pixels will need 6,000,000 processing units!

Multi-processors using moderate numbers of processing units are not very expensive. Pixels are put in a queue; each processor works on one pixel from the queue. In another scheme, each processor works on one line of pixels at a time.

Distributed systems act like multi-processors. For practical implementation, not to degrade the whole system by tying it down to one application, they can do the thinning in the background while they are waiting for some input.

Nowadays, personal computers are in every sector of life, put to multiple uses. They cannot be ignored. Computers in the 50's were the size of a warehouse with the capability of that of a tiny wrist calculator in the 90's. Technology advances rapidly; what seemed impossible in the processing power of personal computers a few years ago has now become a reality. Computers work with digits. With the trend of current technology, it is not surprising that digital video cameras will replace scanners in their functions and we can get instant, cheap data to process. As a video camera will be used for other things than just scanning documents, why should we have special designed hardware, or expensive processors, to do the thinning?

4.2.4 Why Table mapping?

To justify the use of table mapping in the thinning process, this technique is compared with others performing the same job.

4.2.4.1 Template Matching *vs.* Formulas

Template matching is a fast technique on special-purpose hardware, and a slow technique on general-purpose hardware.

Formulas are used to implement thinning in software. They express in formulas or computations a subset of templates. For example: Hilditch crossing number $X_H(p) = 1$ simulates all the templates that are 8-connected (see Fig. 14).

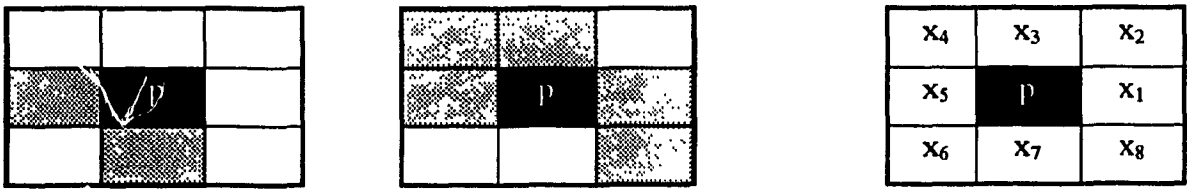


Figure 14: Examples of 8-connected templates.

Most algorithms use formulas in varying degrees of sophistication to express the subset of the thinning templates that is used for pixel removal. This accounts for the vast majority of proposed algorithms. SPTA [20] used a preprocessing step in order to simplify its formulas, and excluded some pixel-removal templates. For example:

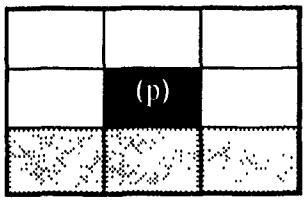


Figure 15: A noise-removal template.

The template in Fig. 15 and its rotations are excluded from the removal-pixel templates, just to simplify the formula. Its removal is deferred to a preprocessing noise-cleaning step.

Formulas are a good alternative to template matching, though they are not the perfect one.

4.2.4.2 Table Mapping *vs.* Template Matching

Table Mapping is a fast implementation of template matching on general-purpose computers. The choice here is hardware-dependent.

4.2.4.3 Table Mapping *vs.* Formulas

When Chen and Hsu [12] were modifying the Zhang and Suen algorithm [10], the formulas became computationally involved, which slowed down the modified algorithm. Table mapping was a good remedy in terms of speed and computation.

By table mapping: the number of templates is not a factor in the performance of the algorithm.

By formulas: to express more templates and have a reasonable performance was and is the challenge facing many researchers.

Chapter 5

Unification

5.1 UNIFIED ALGORITHM

Wu and Tsai's one-pass parallel algorithm for thinning binary patterns was described and implemented on a general-purpose computer by modified table mapping technique. A discussion of the usefulness of the method was also presented. Another justification of the MTMT will be presented in the course of the present section.

Can the MTMT simulate other one-pass parallel algorithms?

Two other one-pass parallel algorithms were proposed by Chin *et al.* and Holt *et al.*, respectively. Chin's algorithm implements template matching on specially designed hardware. Holt's algorithm implements formulas.

A 2-pass parallel algorithm, implemented using table mapping by Chen and Hsu [12] will be included, in order to compare it with the one-pass algorithms.

5.1.1 Chin *et al.*'s Algorithm

Chin *et al.* used two sets of thinning and trimming templates of 3 x 3 mask, and a set of restoring templates of 1 x 4 and 4 x 1 masks. Two versions exist: one version with thinning templates alone, and the second version with thinning and trimming templates. The thinning and trimming templates are used to delete contour pixels. The restoring templates are used to disable the deletion of pixels if certain conditions occur (see Fig. 18); all the templates are applied simultaneously.

The simplest way to implement this algorithm by the MTMT is to give all the templates the same rule. To keep a pixel, after applying the thinning and restoring templates, the rule tests between 2 and 4 pixels to see if the mask matches the restoring templates. The 4 pixels are **a** = (i, j-1), **b** = (i, j+2), **c** = (i-1, j), and **d** = (i+2, j) (Fig. 16).

Another alternative to implement the algorithm is the segmentation of the thinning and trimming templates into 4 groups and the negation of the restoring templates.

To keep a pixel: (a=0 and b=0) or (c=0 and d=0) should be true (Fig. 18).
To delete a pixel: (a=1 or b=1) and (c=1 or d=1) should be true (by negation).

In the first group (see Fig. 17), (p) is deleted unconditionally (**a** and **c** are both black). In the second group, we test pixel **b** = (i, j+2) (as a is white and **c** is black). In the third group, we test pixel **d** = (i+2, j). In the fourth group, we test pixels **b** and **d** (see Fig. 19 for these rules).

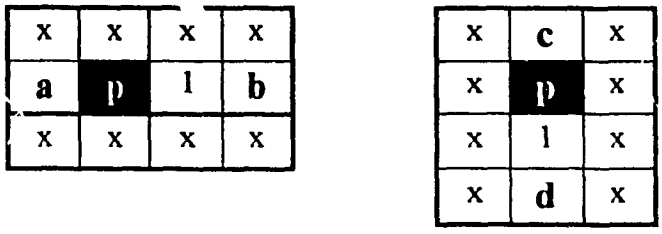


Figure 16: Chin *et al* 1 *x* 4 and 4 *x* 1 restoring templates.

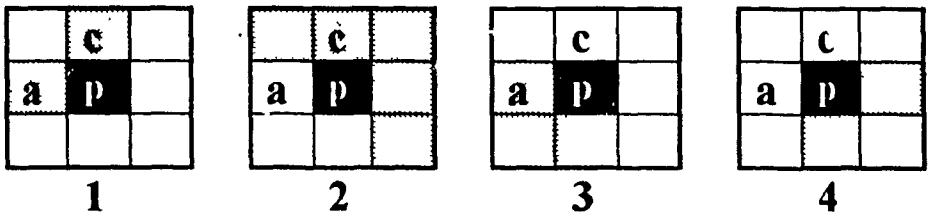


Figure 17: Examples of the 4 groups to segment Chin's templates.

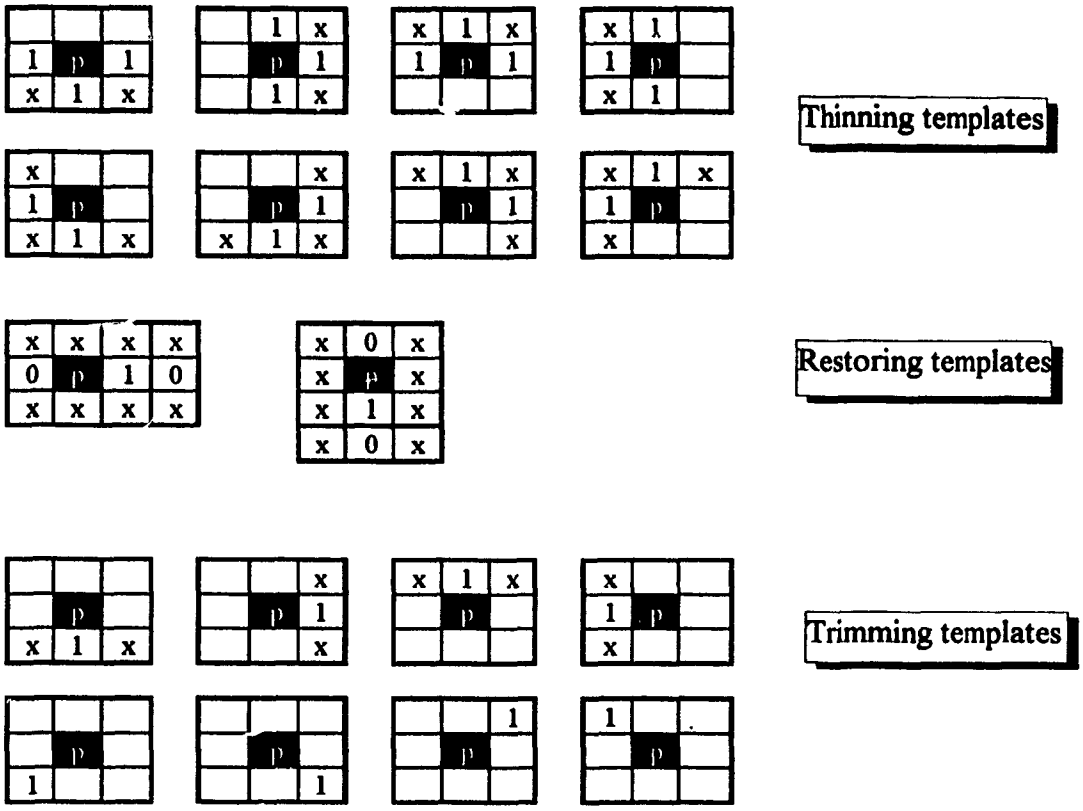


Figure 18: Chin *et al*'s templates.

32	8	1
64	p	2
128	16	4

8 - 4 - 2 - 1

16
32
64
128

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0											2	2			2	2
16			4	4			4	4			2	2			2	2
32											2	2			2	2
48																
64									1	1	1	1				
80	3		3		3		3		1							
96									1	1	1	1				
112	3				3				1							
128																
144			4	4			4	4								
160																
176																
192									1	1						
208	3		3		3		3		1							
224									1	1						
240	3				3				1							

Figure 19: Table look-up for Chin *et al*'s templates.

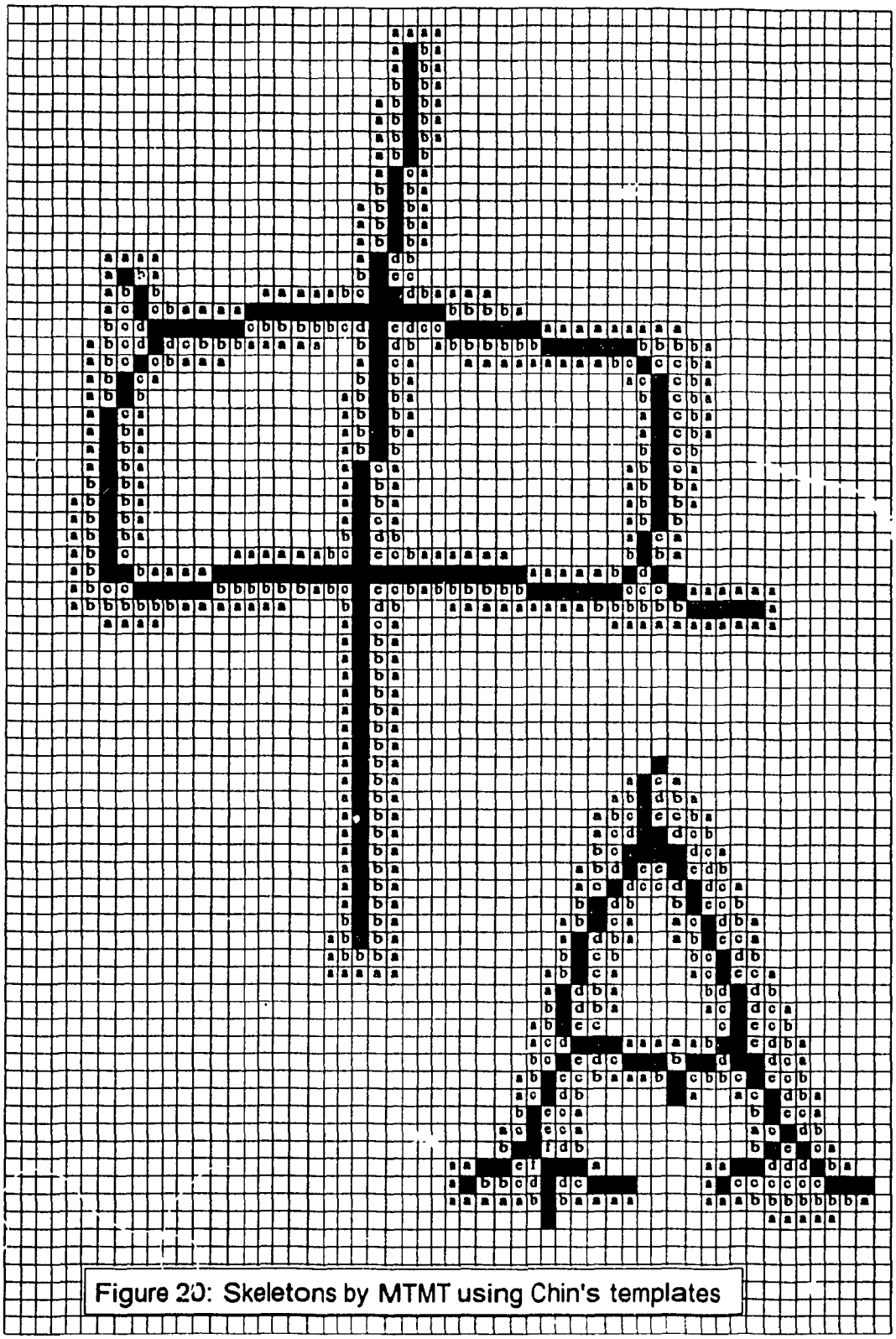


Figure 20: Skeletons by MTMT using Chin's templates

5.1.2 Holt *et al's* Algorithm

Holt *et al* used the following formula to decide on pixel removal:

νC & (~edgeC |
(edgeE & νN & νS) |
(edgeS & νW & νE) |
(edgeE & edgeS & edgeSE))

ν = the value of a pixel. C = (p) in a compass mask, & = AND, | = OR, ~ = NOT.

The edge function [4] formulated a set of connected-templates to make sure that (p) is a 4-contour point in a 4-connected neighborhood. The number of neighbors could be between 2 and 6 or between 2 and 7 (two versions).

The edge function is applied to some neighbors of (p) to enlarge the horizon.

A pixel survives an iteration if the above formula is true. A special condition is included to save an isolated 2 x 2 square by checking the east, south, and south-east neighbors, *i.e.*, if the expression (edgeE & edgeS & edgeSE) is true.

The above formula was used to save a pixel from deletion if it is true.

To delete a pixel, we use the negation of the above formula:

(edgeC &
(~edgeE | ~νN | ~νS) &
(~edgeS | ~νW | ~νE) &
(~edgeE | ~edgeS | ~edgeSE))

To express the Holt algorithm in MTMT, we expand the above formula into its constituent parts, and we simplify:

(edgeC & ~edgeE & ~edgeS)
(edgeC & ~edgeE & ~edgeS & ~edgeSE)
(edgeC & ~edgeE & ~νW)

.

etc.

Since edgeC is in every sub-expression, we generate all the templates corresponding to the expression edgeC.

Expressions like ~vW are included in the generated templates of edgeC.

The expression edgeE means: (E) is checked in its neighborhood (see Fig. 21).

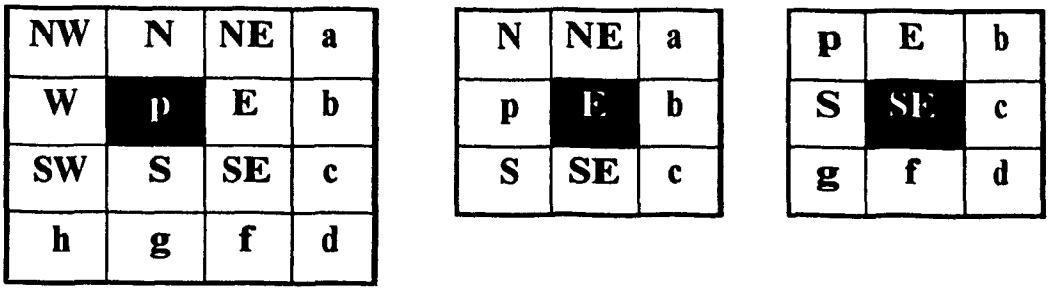


Figure 21: (p)'s 4 *x* 4 window, (E)'s 3 *x* 3 window, and (SE)'s 3 *x* 3 window.

5 neighbors can be known from (p)'s neighborhood (N, p, S, NE, SE). 3 neighbors remain to be known from a 3 *x* 4 window, *viz.* (a, b, c). We have then to test 3 pixels, but since the neighborhood of (E) must be connected, and (E) must be a 4-contour point, then 3 pixels of (E)'s 4-neighbors can be known from (p) neighborhood, *viz.* (p, NE, SE). NE and SE are black as (p) is on an edge. The only way (E) is on an edge is to have (b) white. (c) and (d) are don't-care pixels since (NE), (SE), (N), and (S) are black pixels. If (N) or (S) is white then (E) is not on an edge.

So the edgeE function is simulated by testing the (b) pixel in the 3 *x* 4 window.

The edgeS function, by similar reasoning, is simulated by testing the (g) pixel in the 4 *x* 3 window.

edgeSE happens only in one template, template 22 in table mapping, where the pixels at positions: p, E, S, and SE are black (see Fig. 21). For (p) to be on an isolated square, pixels b, c, d, f, and g should be tested. Since this template occurs often when scanning the bitmap, and the computations are involved, it could be excluded from the thinning set.

32	8	1
64	p	2
128	16	4

8 - 4 - 2 - 1

16
32
64
128

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				1			1	1		1		1				1
16					1		5	1								2
32									1	1		1				1
48																2
64																
80																
96	1								1	1		1				1
112																
128																
144	1				1		1	1								2
160																
176																
192	1															
208	1				1		3	3								
224	1								1	1		1				
240	1				1		3		1	1			1			

Figure 22: Table look-up for Holt *et al*'s templates.

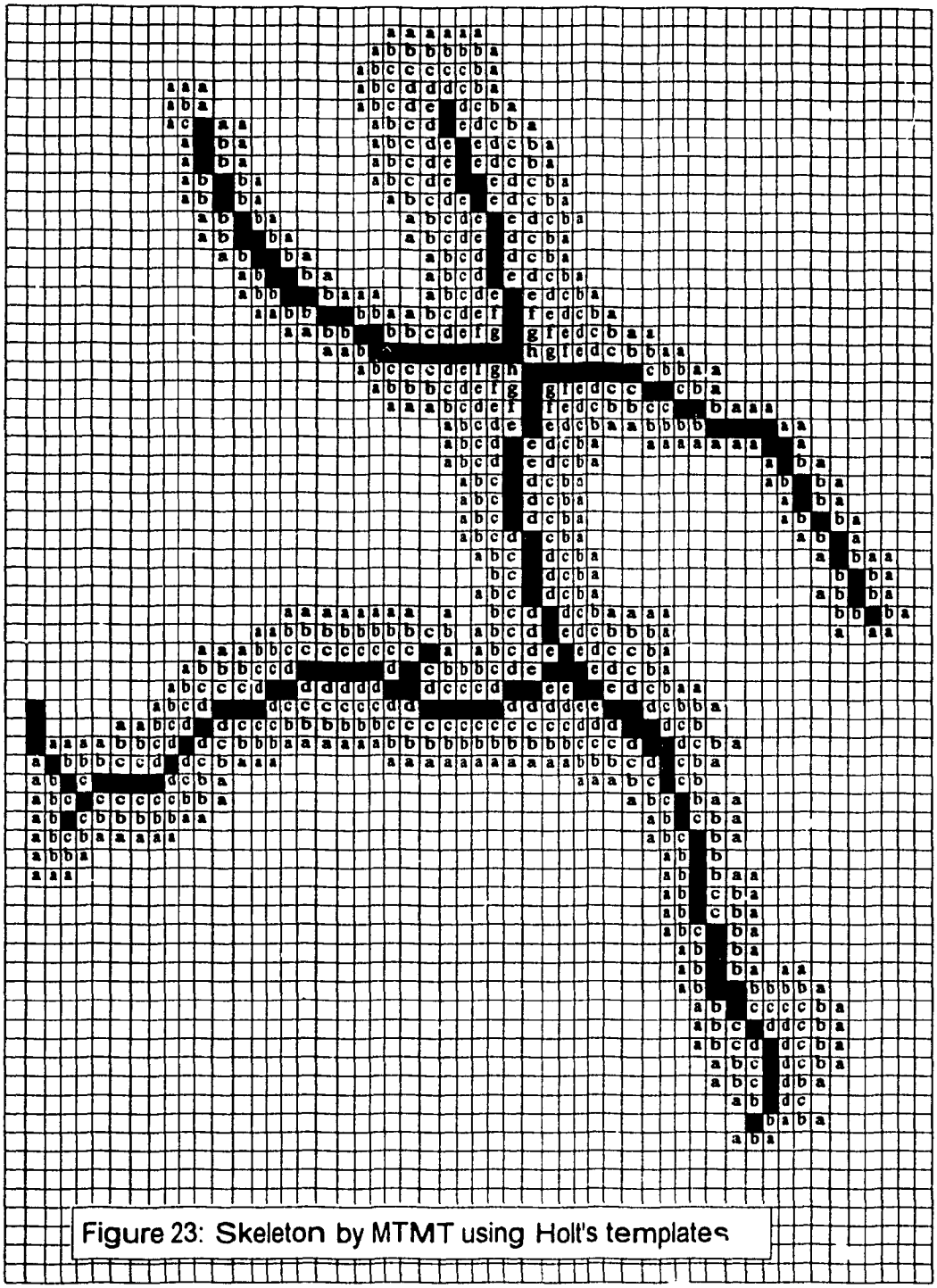


Figure 23: Skeleton by MTMT using Holt's templates

Template 22 tests 5 pixels, some templates do not test any pixel, and the others test only 1 pixel (see Fig. 22).

5.1.3 Unification:

How to express other parallel algorithms?

The Wu and Tsai algorithm was the easiest of the three to simulate by table-mapping; the other two were somehow difficult to simulate. It doesn't matter how complex the algorithms are to decode, the result for the three algorithms is the same: one integer table having entries pointing to simple rules.

Table mapping expresses any (k x m) window. Large windows slow the thinning process and generate large tables, increasing dramatically the number of templates to consider.

How does MTMT unify the implementation of parallel algorithms?

5.1.3.1 2 or 4 sub-iterations

Each sub-iteration has a separate table, to simulate the subset of templates [12]. 2 or 4-sub-iteration algorithms use information from a (3 x 3) window to decide on the removal of pixels

5.1.3.2 Larger windows

The (3 x 3) window is included in larger windows.

We simulate the (3 x 3) part of larger windows by a 256-entry table. The periphery of this (3 x 3) part will be taken care of by the different rules in the MTMT (see Fig. 24).

We reduce this periphery of pixels by half with *biasing*, *i.e.*, test one side and process the other side unconditionally.

Further reduction to this periphery of pixels may be done by considering other properties (as was done for example, in the Wu and Tsai algorithm). The net result ends up writing rules to test at most two pixels.

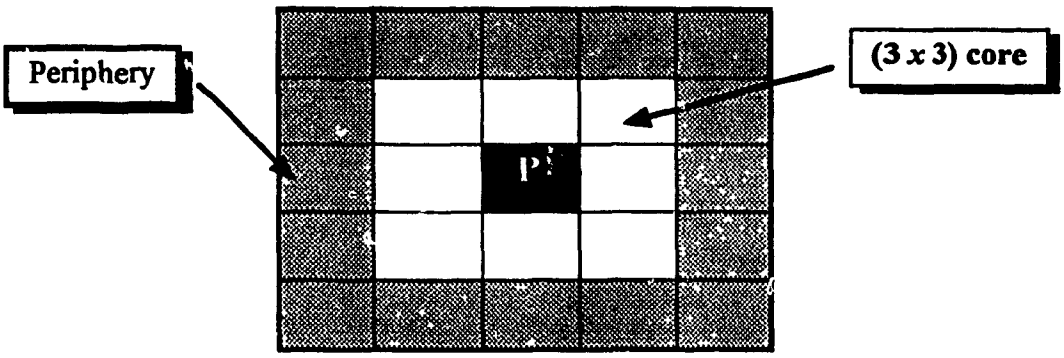


Figure 24: 5 x 5 window with core and periphery.

5.2 Parameters:

How to compare parallel algorithms?

A thinning algorithm should be fast and efficient.

Efficiency can be measured by:

Skeleton number:

The total number of skeletal pixels minus those which are 8-deletable or forming noise. This parameter ensures 8-connectivity and noise-free done. Working in binary figures: 2-pixel diagonal lines can be differently reduced to a 1-pixel line. It is better to inspect manually the 8-deletable and noisy pixels. The closer the skeleton number is to a manually done skeleton the efficient is the algorithm.

Speed, on the other hand, is the simplest to compare: time the algorithms. This procedure is not interpretable, we should know the parameters which affect speed.

Table-mapping offers a platform to compare parallel algorithms by these parameters:

(1) **Number of iterations**: in every iteration, and in the absence of any optimization whatsoever, each pixel look-up takes 8 time units (8 neighbors to test and count their numbers). This time-units number is multiplied by the number of pixels in the bitmap. The number of iterations is then sufficient to rule out the algorithm with more iterations.

The number of iterations can be attributable to two other parameters: the number of missed pixels and the shape of the patterns thinned.

(1.1) **Missed pixels:** are pixels removed in the current iteration, where they should have been removed in the previous iterations. This parameter is attributable to the **number and kind of thinning templates** used. The large number of iterations in thinning character A2 (Fig. 27), by the modified Chen algorithm, is attributable to point (d). Two sub-iterations were performed just to delete this point.

(1.2) **Pattern shape**: when you miss a pixel (p), a configuration results by the removed neighboring pixels which is different from the one which could have resulted if (p) was removed. Now, if this configuration is not in the set of the thinning templates, (p) will be deferred to one more iteration.

For this reason, most parallel algorithms offer figures selected meticulously to show the performance of their algorithm In the improvements section, we see a modified thinning set to Wu's, in order to remove noise (Wu's algorithm is not noise-free as he claimed).

(2) **Number of calls to each rule tested**: for algorithms with the same number of iterations, we look at the number of times each rule is performed. Although these rules are simply testing one or two pixels, nevertheless, they should be compared.

Rule 0: No test done (a break template, or a one missing from the thinning set).

Rule 1: Remove unconditionally.

Rule 2: test one pixel (leftward).

Rule 3: test one pixel (downward).

Rule 4: test two pixels (leftward and downward).

Rule 5: test 5 pixels (Holt's isolated square).

MTMT offers a uniform platform to compare parallel algorithms. 4 algorithms are coded and optimized in the MTMT: two 1-pass running on special hardware and implementing template matching; one 1-pass implementing formulas, and one 2-pass implementing table mapping.

From tables 3 to 6, you can see that the Chin algorithm is slower than the 2-pass version. Wu-M is the modified algorithm proposed in this thesis to account for noise removal. Wu's original, Wu-M, and Holt algorithms all behave (speedwise) approximately the same.

Holt's algorithm produces 8-deletable pixels (efficiency problem).

Chin's slow performance is due to the high number of missed pixels. Template 63 and its rotations are missing from his thinning set. These templates test corners, which are features often encountered in patterns. Missed pixels also produce noisy skeletons.

Unnecessary rules produce noisy skeletons or distorted shapes (see Fig 25). Chin's thinning set includes templates 15 and 120. Template 15 tests if there is a black pixel in the 3 *x* 4 mask, whereas (p) in template 120 is unconditionally deleted. On the other hand, Wu unconditionally deletes template 15, while template 120 is not in his thinning set.

Figure 25: 2 templates from Chin's thinning set favoring diagonal over straight lines.

	WU	WU-M	HOLT	CHIN	CHIN-T	CHEN
Rule 1	36	44	33	16	28	44, 44
Rule 2	3	3	3	12	16	
Rule 3	3	3	3	12	16	
Rule 4				8	8	
Rule 5 (Holt)			1			
Total (templates)	42	50	40	48	68	44,44

Table 2: Number of rules in compared algorithms

Pattern = A1 Bitmap = 308 Image = 179	WU	WU-M	HOLT	CHIN	CHIN-T	CHEN
Iterations	4	4	4	6	12	6
Missed pixels	0	2	2	31	53	0
Skeleton pixels	47	44	44	54	29	44
8-deletable, Noise	3n	0	0	1d, 7n	1d	0
Rule 0	119	110	110	240	351	189
Rule 1	101	104	104	54	77	135
Rule 2	32	32	32	52	60	0
Rule 3	10	10	10	34	49	0
Rule 4	0	0	0	17	18	0
Rule 5	0	0	0	0	0	0

Table 3: Comparison of character A1

Pattern = man Bitmap = 3600 Image = 954	WU	WU-M	HOLT	CHIN	CHIN-T	CHEN
Iterations	9	9	9	15	31	12
Missed pixels	13	18	19	340	397	5
Skeleton pixels	153	138	149	170	72	134
8-deletable, Noise	13n	0	10d	5d, 21n	1d	0
Rule 0	936	819	888	2029	2576	1286
Rule 1	590	602	591	237	314	820
Rule 2	128	130	128	347	410	0
Rule 3	104	107	104	359	432	0
Rule 4	0	0	0	72	73	0
Rule 5	0	0	3	0	0	0

Table 4: Comparison of graphics 'man'

Pattern = A2 Bitmap = 729 Image = 312	WU	WU-M	HOLT	CHIN	CHIN-T	CHEN
Iterations	4	5	5	7	13	10
Missed pixels	2	2	2	105	128	2
Skeleton pixels	68	67	71	76	35	60
8-deletable, Noise	1n	0	5d	10d, 2n	2d, 1n	0
Rule 0	144	141	222	406	492	490
Rule 1	185	186	179	95	127	252
Rule 2	50	50	50	68	77	0
Rule 3	15	16	15	78	98	0
Rule 4	0	0	0	53	57	0
Rule 5	0	0	3	0	0	0

Table 5: Comparison of Character A2

Pattern = B Bitmap = 238 Image = 141	WU	WU-M	HOLT	CHIN	CHIN-T	CHEN
Iterations	3	3	3	4	4	6
Missed pixels	0	0	0	24	24	0
Skeleton pixels	44	44	50	42	41	43
8-deletable, Noise	0	0	5d	2d, 1n	2d	0
Rule 0	88	88	100	128	122	201
Rule 1	68	68	61	46	46	98
Rule 2	17	17	17	29	33	0
Rule 3	14	14	14	28	28	0
Rule 4	0	0	0	17	17	0
Rule 5	0	0	1	0	0	0

Table 6: Comparison of character B

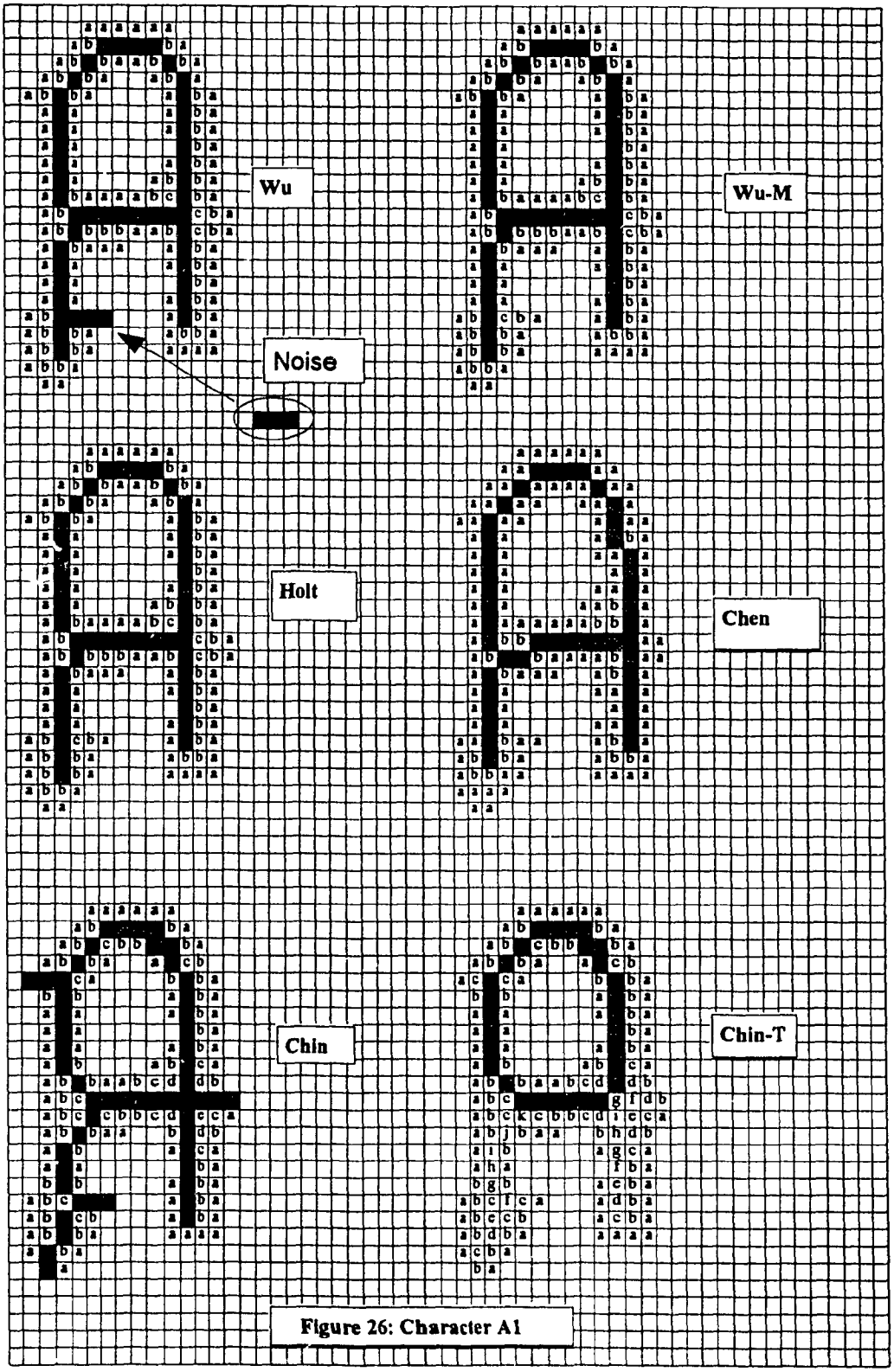


Figure 26: Character A1

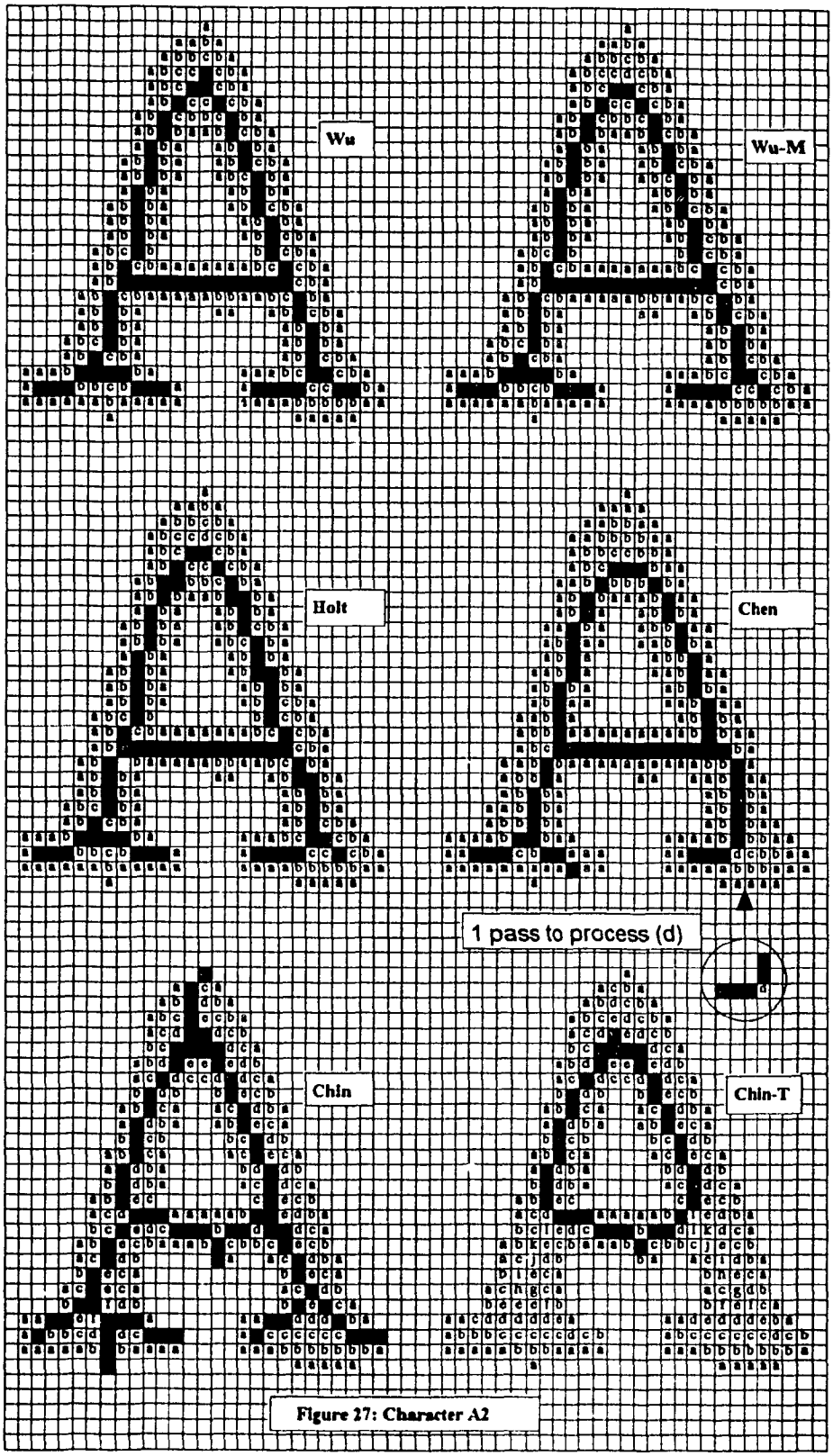


Figure 27: Character A2

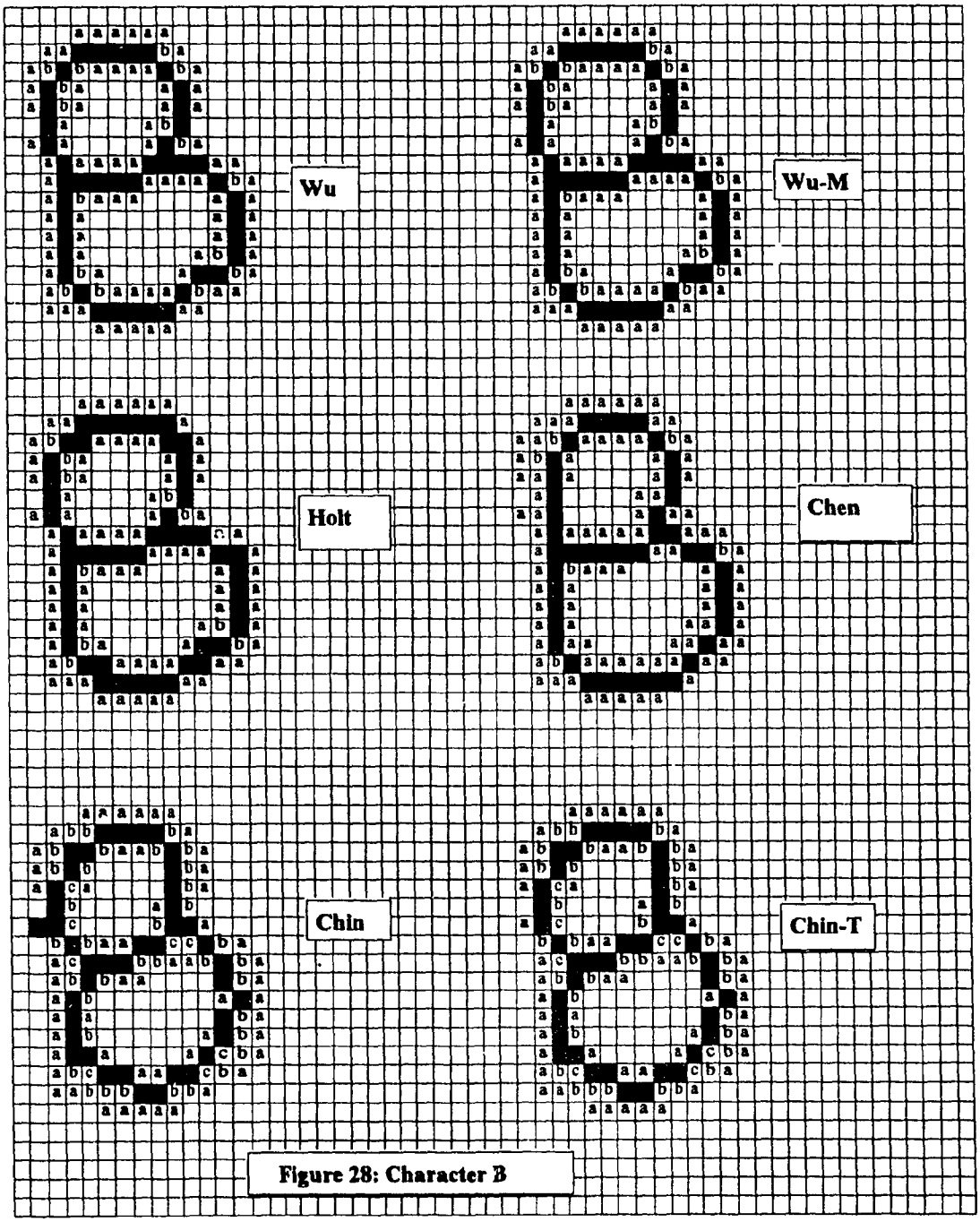


Figure 28: Character B

5.3 Improvements

- MTMT

The MTMT is a fast and efficient thinning technique running on general-purpose computers. Major improvements can be found later on, when discussing adaptation; while buffering, queuing, low-level programming, *etc.* are small improvements to speed it up.

- Other algorithms

Improvements can be made to the already discussed algorithms, if they are implemented in the MTMT.

Templates with b(p) =2 are missing from Wu's thinning set, resulting in tails in some patterns (see Fig. 26). By including these 8 templates, giving them rule 1, *i.e.*, unconditionally delete, as these templates erode and do not disconnect the pattern, we get the modified Wu-M algorithm.

If we add to Holt's thinning set the templates (10, 18, 72, 73, 80, 84, 112, 116, 200, 201) which delete diagonal lines, giving them rule 1, we get the modified Holt algorithm, which is the same algorithm as the modified Wu (Wu-M).

Chin's algorithm needs a major overhaul.

The modified thinning set produces noise-free and one-pixel wide line skeletons, as are shown in the tables. Wu-M is fast and more efficient.

The issue is not to add small modifications to an existing algorithm, but to find one definitive algorithm.

Chapter 6

Adaptation

6.1 Labeling

Consider a 3 x 3 mask: the D-neighbors of (p) generate 16 different combinations; for each one of these configurations, the I-neighbors of (p) generate 16 configurations; In total, they generate 256 configurations.

Numbers are given to the D-neighbors of (p) as in (a) in Fig. 29. We denote this labeling the D-neighbors mask. The I-neighbors mask are labeled as in (b) in Fig. 29.

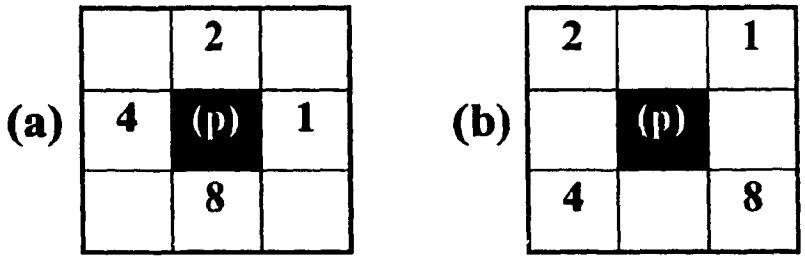


Figure 29: D-neighbors and I-neighbors masks.

As in integer masks, a cell takes its number if it is black. (p) will have a number from 0 to 15 in the D-neighbors mask. Instead of numbers, (p) takes a letter from (A to P). Actually, we work with numbers inside the computer, and for presentation purposes we express these numbers by letters.

Now, (p) has 16 different personalities, A-P, and for each personality, it is found in 16 different situations (according to the I-neighbors mask). (p) will have a subscript from 0 to 15 to describe these different situations. For example:

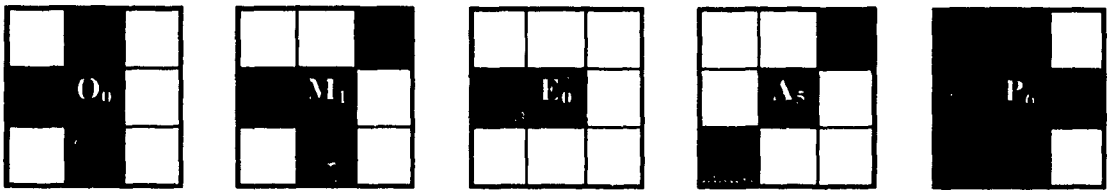


Figure 30: Different templates in LABEL mask.

In Fig. 30, (p), in mask (b), has 2 black D-neighbors in cells 4 and 8 respectively, *i.e.*, number 12 or letter M, and 1 black I-neighbor at cell 1, *i.e.*, number 1 as subscript. (p) then, has personality M in situation 1 (M_1). We can work either with labels or with labels and subscripts depending on how large we need the horizon..

6.2 Segmentation

In an INTEGER mask, each template has a unique number (0-255). To describe a set of templates, we have to refer to each one of them. In LABEL mask: for example: a group of templates having only one white D-neighbor in E position (Compass mask) is referred to as O-templates.

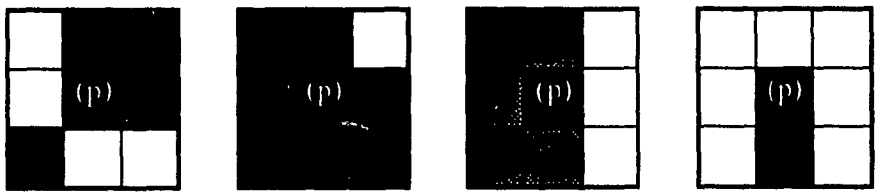
Each of the *x*-templates (*x* = A..P) can be in a certain category, classified as follows (Examples are shown in Fig. 31):

(1) Template (a) is called a Break-Template. The removal of (p) results in the pattern being disconnected.

(2) Template (b) is an Inner-Template. (p) is not on the contour (considering 4-adjacency to the background).

(3) Template (c) is a Delete-Template. The removal of (p) doesn't disrupt the connectivity of the pattern.

(4) Template (d) is an End-Template. The number of black pixels b(p) is 1.



(a) Break (b) Inner (c) Delete (d) End

Figure 31: Different kinds of templates.

Appendices b, c, d contain all the templates in a 3 *x* 3 mask classified by categories and types. The number beneath each template is its number in an integer mask.

There are 123 Break-Templates which are not involved in thinning. These will not be included in the thinning templates.

The 117 Delete-Templates include the End-Templates. The Delete-Templates are considered as thinning templates.

There are 16 Inner-Templates which are not normally involved in thinning, as they create holes.

6.3 Patterns

The 4 algorithms which were compared in chapter 5, as most other algorithms, used only a subset of the 117 thinning-templates set, which resulted in missed pixels and more iterations had to be done.

The simplest solution to this problem is to treat the set of Delete-Templates as thinning templates, but all the patterns don't exhibit the same properties: what is

considered noise in characters is not the case in fingerprints, or engineering drawings thinning.

The alternative is then to have a maximal thinning set for each type of patterns, consisting of all the delete-templates or a maximum subset from it.

Since the implementation is restricted to a 3 *x* 3 mask, rules must be set for the interactions of the different thinning templates.

6.4 Foundation Rules

The 3 *x* 3 mask is table-mapped. The rules are entrusted to enlarge the horizon by inspecting pixels in the periphery of the mask.

The rules could differ as applied to each kind of patterns; this diversity can be accounted for by testing different periphery pixels for each thinning template.

The rules should be simple, testing the fewest pixels possible.

Considering all the combinations of the thinning templates as they interact would be deterring; there must be a way to guide us through formulating the rules.

The Label mask implies direction in its code. For example: the E-templates have only one black D-neighbor in the west; they have to look in this direction only. Looking in one direction offers 8 possibilities to consider (the 3 pixels on this side of the periphery). Appendix A contains the different combinations in the 4 directions.

Two choices are available to code the rules:

(1) - take advantage of the bias by fixing two sides and treat the other two. This is what most algorithms do. The advantage is, at least half of the rules will not test any pixel.

(2) - test in four directions. The advantage is that besides connectivity, one can gain some geometric properties from the pattern, and avoid excessive erosion and

spurs. As bias will take care of connectivity alone, here we have also the advantage of detecting true features soon, such as end-lines.

For example: E_0 can test the west pixel to know if it is on a straight line, or a noise. Depending on the label of its west-neighbor, E_0 has the following possibilities:

1- B_0, B_6, or F_6: E_0 is noise

2- F_0: E_0 is on a straight line

3- B_2, B_4, F_2, and F_4: You check the label of one pixel in the periphery and you will be working in a 7 x 7 neighborhood by looking at two pixels only instead of the 2^{48} possibilities.

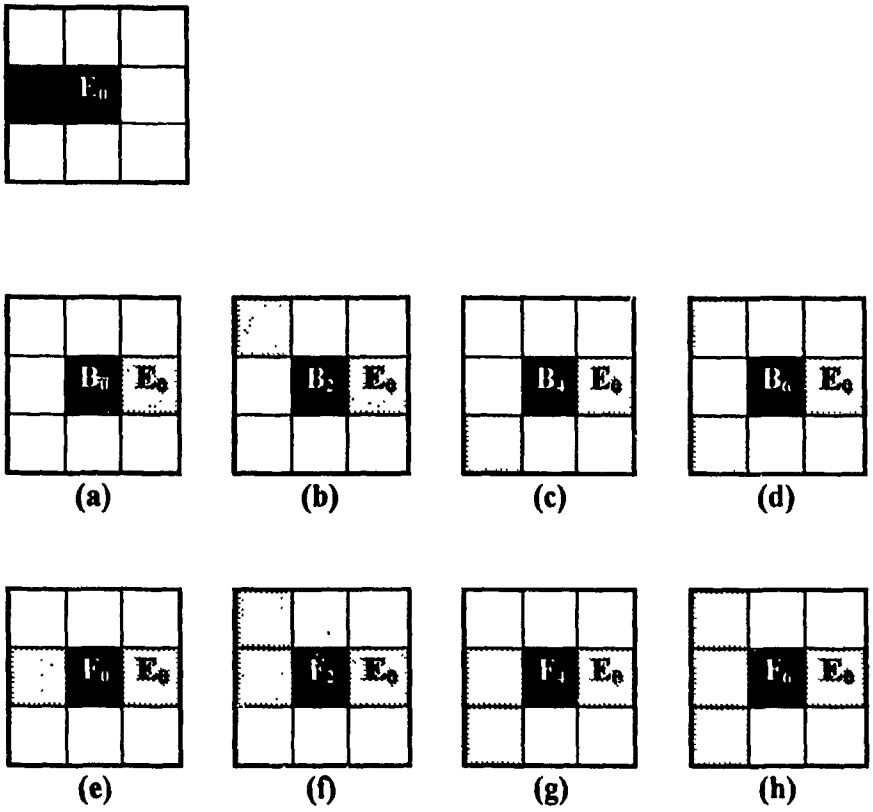


Figure 32: Templates that E_0 can meet.

6.5 New Labeling Technique

The labeling technique was not devised to aid in the discussion alone. It is a new technique to replace table-mapping. The contour pixels are initially labeled by a labeling algorithm and put in a queue. Each labeled pixel (L) can expand its horizon by looking at the labels of its neighbors in its allowed directions. If (L) is to be removed, those neighbors are put at the end of the queue to be processed, when the current contour finishes processing.

In this technique, each black pixel is processed only once, while the white pixels are not processed at all The testing will be done at most on three pixels in the 3 *x* 3 mask; since we infer from the label of the mask pixels what exists on the periphery of the 3 *x* 3 window (see Fig. 32). Depending on the application, one may test the label of one pixel in the periphery, and the horizon of a 7 x 7 neighborhood is open.

The pixels labeling algorithm is functional, the updating and queuing of pixels also works, each label knows what are the other labels facing it in all directions, what really is needed is the definition of the output in order to start formulating the rules (§6.4).

The Pattern and its rotation's syndrome

MTMT is a fast method to design parallel algorithm, Labeling is an improvement over MTMT. For example: both can resolve the problem of generating different skeletons for the same pattern taken from different angles (see Fig 34) but, with different efficiency.

The problem arises due to the following reason:

1) In the three algorithms expressed by the MTMT in chapter 5 as in all other algorithms, they use a subset of the thinning set (see Appendix C), which results in missed pixels, and as such the erosion will not be symmetrical from all sides as some thinning templates will be missing to erode from the left while those from the right are present.

2) The case of a 2-pixel wide line connected to other lines in the pattern. To maintain a 2-pixel wide line, one has to fix one side and delete the other

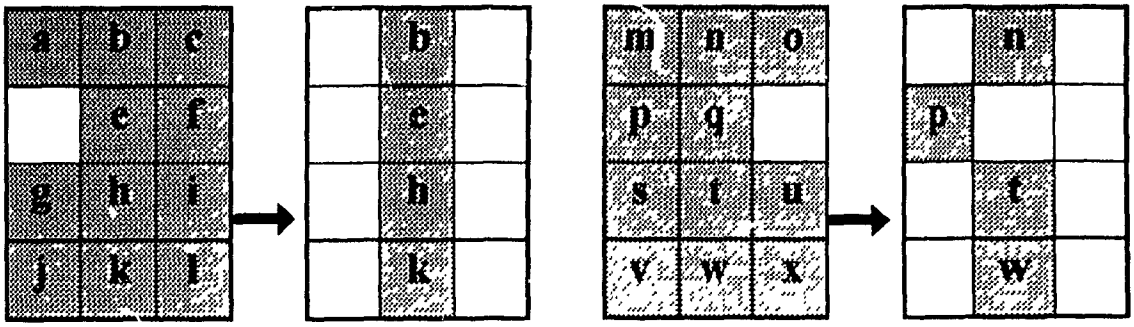


Figure 33: Different skeletons from a pattern and its rotation.

Suppose the bias is from the left side, i.e., we keep the left pixel of a 2-pixel wide line and we delete the pixel on the right. In this case pixels (e) and (p) are kept while pixels (f) and (q) are removed, generating different skeletons (Fig. 33). In Fig. 34, the (a) and (c) patterns have the same skeletons (as both are biased to the same side). The (a) and (b) patterns do not have the same skeletons due to the bias and the missing of some thinning templates (the case of Wu's templates).

Including all the thinning templates in the thinning set solves (1) above. The solution to (2) - ex: the case for a vertical line - is to rank the templates by priority on removing pixels from the left and right of a 2-pixel wide line, keeping a template and its symmetry one after the other. For example in Fig. 33, the template which has (e) as the center will be ranked first and its symmetry (q) second, while (p) will be ranked third and its symmetry (f) fourth, etc. if (e) meets (q) then (e) takes precedence (both test the periphery for the existence of the other). If (p) meets (q) then (q) takes precedence and thus we ensure generating the same skeleton for different rotations of a pattern.

In the MTMT that means we have to test 3 pixels in the periphery at most instead of just one, not a maior drawback. In the Label Technique, the three pixels in the periphery can be inferred from the label of a pixel in the 3x3 window (see § 6 4).

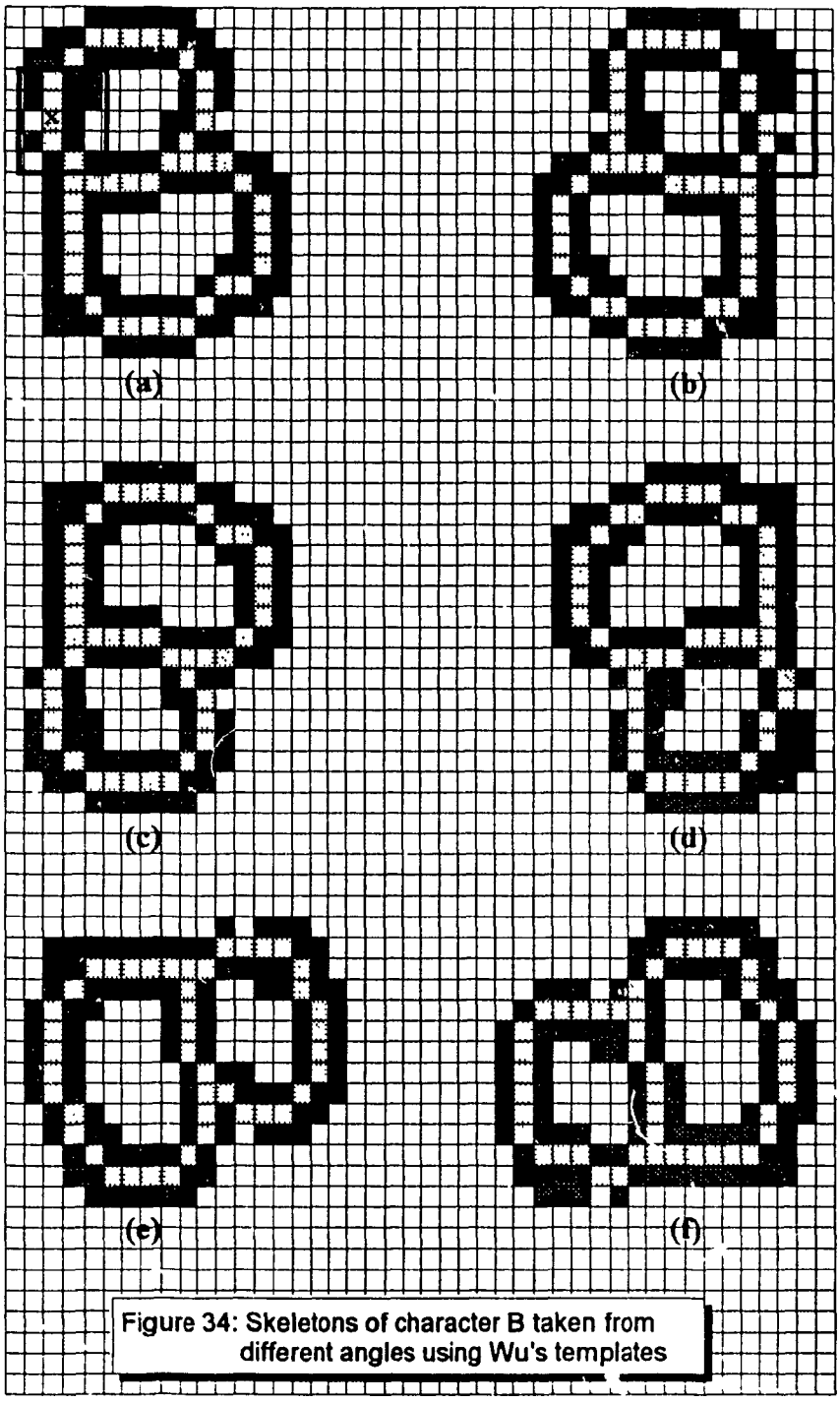


Figure 34: Skeletons of character B taken from different angles using Wu's templates

Chapter 7

Conclusion

7.1 Summary of contributions

Thinning algorithms in the literature are primarily concerned with how to maintain connectivity of the skeleton while thinning the pattern A contour point (p) highlighted by a 3 *x* 3 window will be in any one of 256 different configurations. In some of these configurations (117 thinning templates), (p) will not break the skeleton if it is properly removed (see § 6.2).

In order to maintain connectivity, most algorithms remove (p) if its neighborhood matches any one of these thinning templates, and test also two other pixels neighborhoods. As expressing all the thinning templates by computation or formulas is complex and slows down the process, existing algorithms differ by expressing only subsets of the thinning set in order to speed up their algorithms.

In the MTMT, expressing templates is not a factor affecting speed, as each mask is computed the same way and the sum of the cells (numbered by modulo 2) points to an address in a table to decide the removal of a pixel The entries in the table point to rules to address pixels in the periphery of the 3 *x* 3 window

In other algorithms, formulas or computations express a subset of the thinning set *globally*; in the MTMT, all the thinning set is expressed and most importantly each and every template is addressed and considered in its neighborhood *individually*. More than that, each template has its unique personality and can interact with its neighborhood

differently permitting us to define different rules to be adapted to different kinds of patterns.

The MTMT is faster than other thinning methodologies as it tests and counts 8 cells of a 3 *x* 3 mask and tests one to three pixels in the periphery, while formulas or computation-based algorithms test and count 8 cells 2 to 3 times plus some additional calculations.

The MTMT can further be optimized using assembly language: each of the 3 registers will hold part of a different line from the bitmap. Each register will be rotated with the carry into the accumulator. The address is formed in the accumulator.

The Label Mask Technique (LMT) is a faster technique than the MTMT. LMT uses MTMT as part of its calculations but, on at most 3 instead of 8 pixels. Besides, the code in the pixels infers the contents of other cells in the mask and periphery. By testing 2 or 3 pixels one can work in 5 *x* 5 and 7 *x* 7 masks.

7.2 Future research

The MTMT offers a medium by which we can find one definitive set of templates and rules for each kind of pattern.

The LABEL mask is another medium, which is faster and more horizon-enlarging than the MTMT. Hence we have two flexible multi-use tools; what remains is to define the job to be done. However, each pattern must be studied extensively in all aspects till we find a set of prominent characteristics. One way to do this is by collecting all the variant shapes that the pattern could take, and skeletonize them by hand [8].

The most important part here is to agree on the set of manually-done skeletons. Once this set is produced, if not universally agreed on but at least recognized by the academic field as a valid set, then the rest is easy; we will end up with one set of simple rules to do the thinning.

We could have many different sets of rules to do the job; we test on the speed and how close they are to the manually thinned pattern.

Considering each template from the thinning set, the different rules can be limited by considering the LABEL masks (§ 6.4).

Finally, one definitive and unique thinning algorithm will emerge for each different kind of pattern, and this can be foreseen by:

- defining reference skeletons done by hand [8, 47].
- going through all the possibilities in the LABEL mask.
- setting simple rules.
- doing it yourself, or letting the computer do it for you.

The MTMT and the LABEL mask shift the task of thinning from the mere concern of preserving connectivity into the realm of preserving topology.

References

[1] L. Lam, S. -W. Lee, and C. Y. Suen, Thinning Methodologies - A Comprehensive Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 9, pages 869-885, September, 1992.

[2] M. Del Sordo and T. Kasvand. Neighborhood look-up tables for skeletonization. Proc. 4th Scand. Conf. Image Anal. (Trondheim, Norway), 1985, pages 663-670.

[3] R. -Y. Wu and W. -H. Tsai. A new one-pass parallel algorithm for binary images. Pattern Recognition Letters, vol. 13, pages 715-723, 1992.

[4] C. M. Holt, A. Stewart, M. Clint, and R. H. Perrot. An improved parallel thinning algorithm. Comm. ACM, vol. 30, no. 2, pages 156-160, 1987.

[5] R. T. Chin, H. -K. Wan, D. L. stover, and R. D. Iverson. A one-pass thinning algorithm and its parallel implementation. Computer Vision, Graphics, and Image Processing, vol. 40, pages 30-40, 1987.

[6] Y. -S. Chen and W. -H. Hsu. A comparison of some one-pass parallel thinnings. Pattern Recognition Letters, vol. 11, no. 1, pages 35-41, 1990.

[7] Y. -S. Chen and W. -H. Hsu. A systematic approach for designing 2-subcycle and pseudo 1-subcycle parallel thinning algorithms. Pattern Recognition, vol. 22, no. 3, pages 267-282, 1989.

[8] R. Plamondon and C. Y. Suen. Thinning of digitized characters from subjective experiments: A proposal for a systematic evaluation protocol of algorithms. Computer Vision, and Shape Recognition (A. Krzyzak, T. Kasvand, and C. Y. Suen, Eds.). Singapore: World Scientific, 1989, pages 261-272.

[9] D. Rutovitz. Pattern Recognition. J. Roy. tat. Soc., vol. 129, Series A, pages 504-530, 1966.

[10] T. Y. Zhang and C. Y. Suen. A fast thinning algorithm for thinning digital patterns. Comm. ACM, vol. 27, no. 3, pages 236-239, 1984.

[11] Y. Y. Zhang and P. S. P. Wang. A modified parallel thinning algorithm. Proc. 9th Int. Conf. Patt. Recogn. (Rome, Italy), 1988, pages 1023-1025.

[12] Y. -S. Chen and W. -H. Hsu. A modified fast parallel algorithm for thinning digital patterns. Pattern Recognition Letters, vol. 7, no. 2, pages 99-106, 1988.

[13] C. -S Chen and W. -H. Tsai. A new fast one-pass thinning algorithm and its parallel hardware implementation. Pattern Recognition Letters, vol. 11, pages 471-477, 1990.

[14] U. Eckhardt. A note on Rutovitz method for parallel thinning. Pattern Recognition Letters, vol. 8, no. 1, pages 35-38, 1988.

[15] R. W. Hall. Fast parallel thinning algorithms: Parallel speed and connectivity preserving. Comm. ACM, vil. 32, no. 1, pages 124-131, 1989.

[16] P. S. P. Wang. An Improved fast parallel thinning algorithm for digital patterns. Proc. Int. Conf. Comput. Vision Patt. Recogn. (San Fransisco), 1985, pages 364-367.

[17] A. Rosenfeld. A characterization of parallel thinning algorithms. Inform. Contr., vol. 29, no. 3, pages 286-291, 1975.

[18] P. S. P. Wang, L. -W. Hui, and T. Fleming. Further improved fast parallel thinning algorithm for digital patterns. Computer Vision, Image Processing and Communications - Systems and applications (P. S. P. Wang, Ed.). Singapore: World Scientific, 1986, pages 37-40.

[19] C. J. Hilditch. Linear skeletons from square cupboards. Machine Intell. (B. Meltzer and D. Michie, Eds.). New York: Amer, Elsevier, 1969, pages 403-420, vol. 4.

[20] N. J. Naccache and R. Shinghal. SPTA: A proposed algorithm for thinning binary patterns. IEEE Trans. Syst. Man Cybern., vol. SMC-14, no. 3, pages 409-418, 1984.

[21] P. C. K. Kwok. A thinning Algorithm by contour generation. Comm. ACM, vol. 31, no. 11, pages 1314-1324, 1988.

[22] P. C. K. Kwok. Customising thinning algorithms. Proc. IEEE Int. Conf. Image Processing Applications, 1989, pages 633-637.

[23] L. J. Vliet and B. J. H. Verwer. A contour processing method for fast neighborhood operations. Pattern Recognition letters, vol. 7, no. 1, pages 27-36, 1988.

[24] W. Xu and C. Wang. CGT: A fast thinning algorithm implemented on a sequential computer. IEEE Trans. Syst. Man Cybern., vol. SMC-17, no. 5, pages 847-851, 1987.

[25] A. Favre and H. Keller. Parallel syntactic thinning by recoding of binary pictures. Comput. Vision Graphics Image Processing, vol. 23, pages 99-112, 1983.

[26] J. Piper. Efficient implementation of skeletonization using interval coding. Pattern Recogn. Letters, vol. 3, no. 6, pages 389-397, 1985.

[27] L. O'Gorman. k x k thinning. Comput. Vision Graphics Image Processing, vol. 51, pages 195-215, 1990.

[28] Y. Y. Zhang and P. S. P. Wang. A maximum algorithm for thinning digital patterns. Proc. 9th Int. Conf. Patt. Recogn. (Rome, Italy), 1988, pages 942-944.

[29] A. M. Vossepoel, J. P. Buys, and G. Koelewijn. Skeletons from chain-coded contours. Proc. 10th Int. Conf. Patt. Recogn. (Atlantic City), 1990, pages 70-73.

[30] V. K. Govindan and A. P. Shivaprasad. A pattern adaptive thinning algorithm. Pattern Recogn. vol. 20, no. 6, pages 623-637, 1987.

[31] C. Arcelli. Pattern thinning by contour tracing. Comput. Graphics Image Processing, vol. 17, pages 130-144, 1981.

[32] R. M. K. Sinha. Primitive recognition and skeletonization via labeling. Proc. Int. Conf. Syst. Man Cybern. (Halifax, Canada), 1984, pages 272-279.

[33] T. Pavlidis. A thinning algorithm for discrete binary images. Comput. Graphics Image Processing, vol. 13, pages 142-157, 1980.

[34] O. Baruch. Line thinning by line following. Pattern Recognition Letters. vol. 8, no. 4, pages 271-276, 1988.

[35] J. -D. Dessimoz. Specialized edge-trackers for contour extraction and line-thinning. Signal Processing, vol. 2, no. 1, pages 71-73, 1980.

[36] R. N. Jones and M. C. Fairhurst. Skeletonization of binary patterns: A heuristic approach. Electron. Lett., vol. 14, no. 0, pages 265-266, 1978.

[37] M. P. Martinez-Perez, J. Jimenez, and J. L. Navalon. A thinning algorithm based on contours. Comput. Vision Graphics Image Processing, vol 38. pages 186-201, 1987.

[38] B. Moayer and K. S. Fu. A syntactic approach to fingerprint pattern recognition, Pattern Recogn., vol. 7, pages 1-23, 1975.

[39] C. Lantuejoul. Skeletonization in quantitative metallography. Issues in Digital Image Processing (R. M. Haralick and J. C. Simons, Eds.). Amsterdam: Sijthoff and Noordoff, 1980, pages 107-135.

[40] A. Nakayama, F. Kimura, Y. Yoshida, and T. Fukumura. An efficient thinning algorithm for large scale images based upon pipeline structures. Proc. 7th Int. Conf. Patt. Recogn. (Montreal), 1984, pages 1184-1187.

[41] T. V. Nguyen and J. Slansky. A fast skeleton-finder for coronory arteries. Proc. 8th Int. Conf. Patt. Recogn. (Paris, France), 1986, pages 481-483.

[42] J. F. O'Callaghan and J. Loveday. Quantitative measurement of soil cracking patterns. Pattern Recogn., vol. 5, pages 83-98, 1973.

[43] J. -I. Toriwaki and S. Yokoi. Distance transformation and skeletons of digitized pictures with applications. Progrss in Pattern Recogn. (L. N. Kanal and A. Rosenfeld, Eds.). New York: North Holland, 1971, pages 189-264.

[44] L. O'Gorman and R. Kasturi. Document Image Analysis. pages 3, 15-22.

[45] Poty and S. Ubeda. A parallel thinning algorithm using k x k window. Thinning Methodologies for Pattern Recognition (C. Y. Suen and P. S. P. Wang, Eds.). Singapore: World Scientific, 1994, pages 220-261

[46] S. -W. Lee, L. Lam, and C. Y. Suen. A systematic evaluation of skeletonization algorithms. Thinning Methodologies for Pattern Recognition (C. Y. Suen and P. S. P. Wang, Eds.). Singapore: World Scientific, 1994, pages 239-261.

[47] R. Plamondon, C. Y. Suen, M. Bourdeau, and C Barriere. Methodologies for evaluating thinning algorithms for character recognition. Thinning Methodologies for Pattern Recognition (C. Y. Suen and P. S P. Wang, Eds.). Singapore: World Scientific, 1994, pages 283-305

[48] B. J. H. Verwer, L. J. V. Vliet, P. W. Verbeek. Binary and gray-value skeletons: Metrics and Algorithms. Thinning Methodologies for Pattern Recognition (C. Y. Suen and P. S. P. Wang, Eds.). Singapore: World Scientific, 1994, pages 323-344.

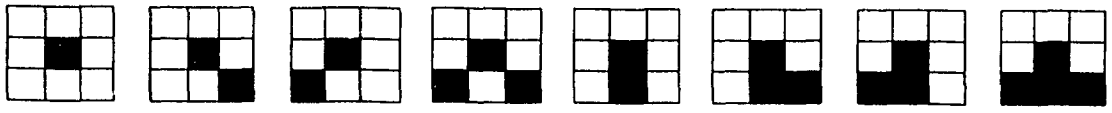
[49] B. K. Jang and R. T. Chin. Reconstructable parallel thinning. Thinning Methodologies for Pattern Recognition (C. Y. Suen and P. S. P. Wang, Eds.). Singapore: World Scientific, 1994, pages 191-216.

[50] Y. Y. Zhang and P. S. p Wang. Analytical comparison of thinning algorithms. Thinning Methodologies for Pattern Recognition (C. Y. Suen and P. S P Wang, Eds.). Singapore: World Scientific, 1994, pages 263-282.

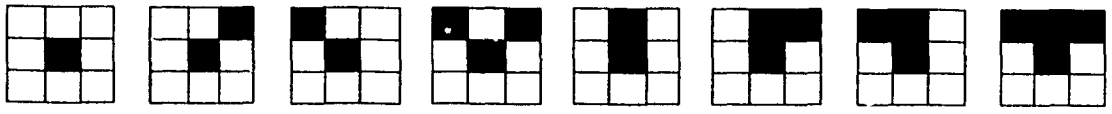
Appendix A

Combinations in the 4 directions of the periphery facing a pixel

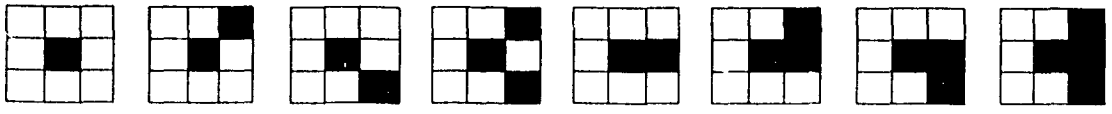
DOWN



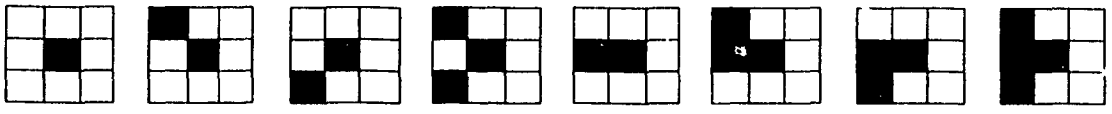
UP



RIGHT

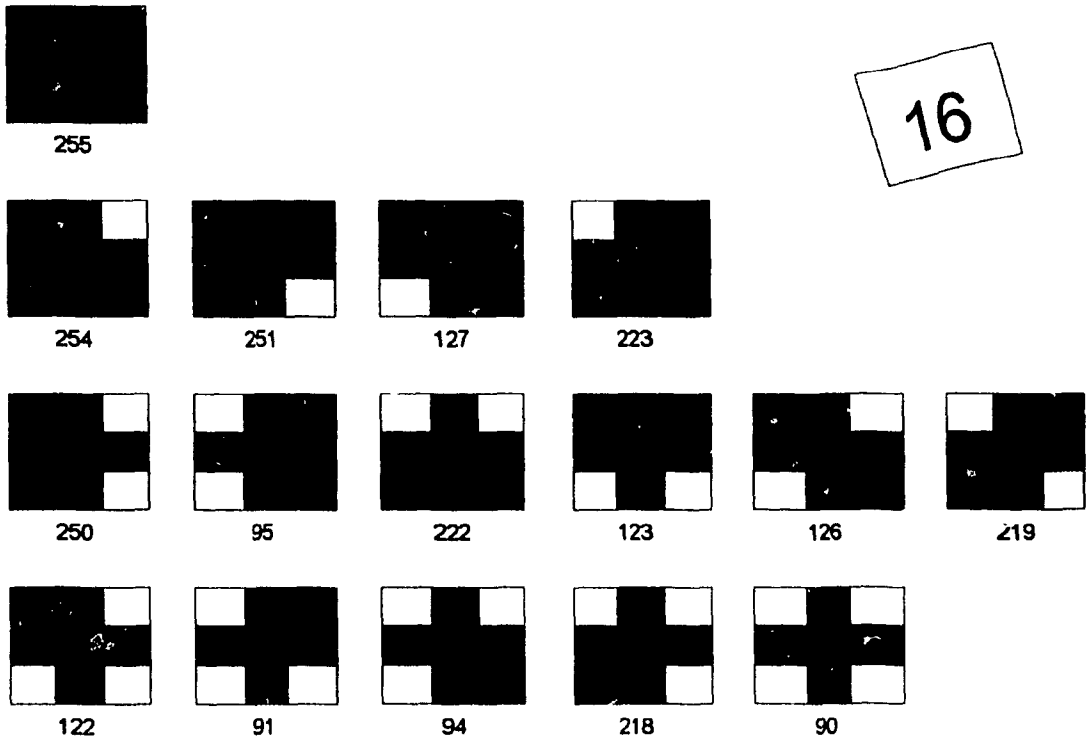


LEFT



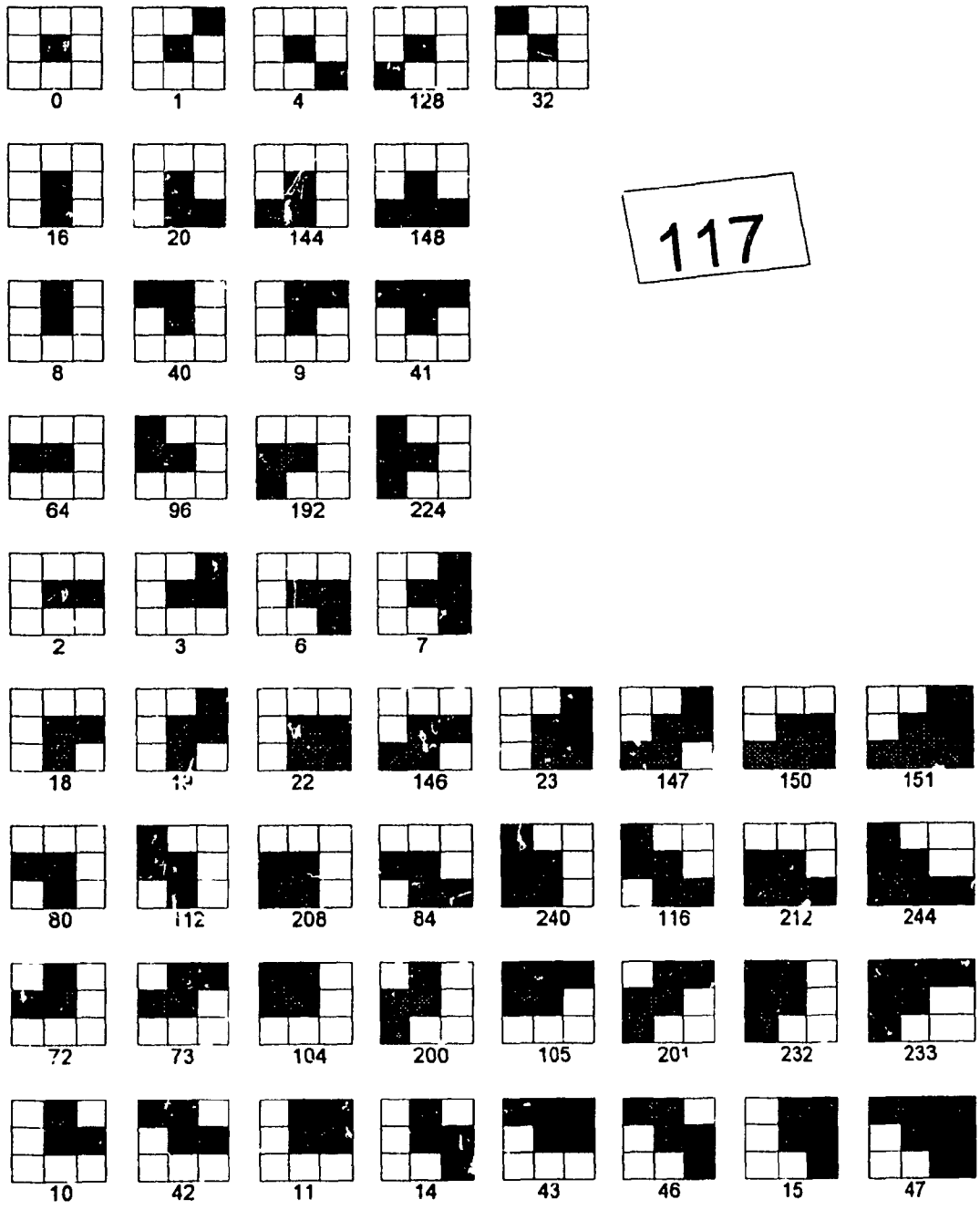
Appendix B

Inner Templates



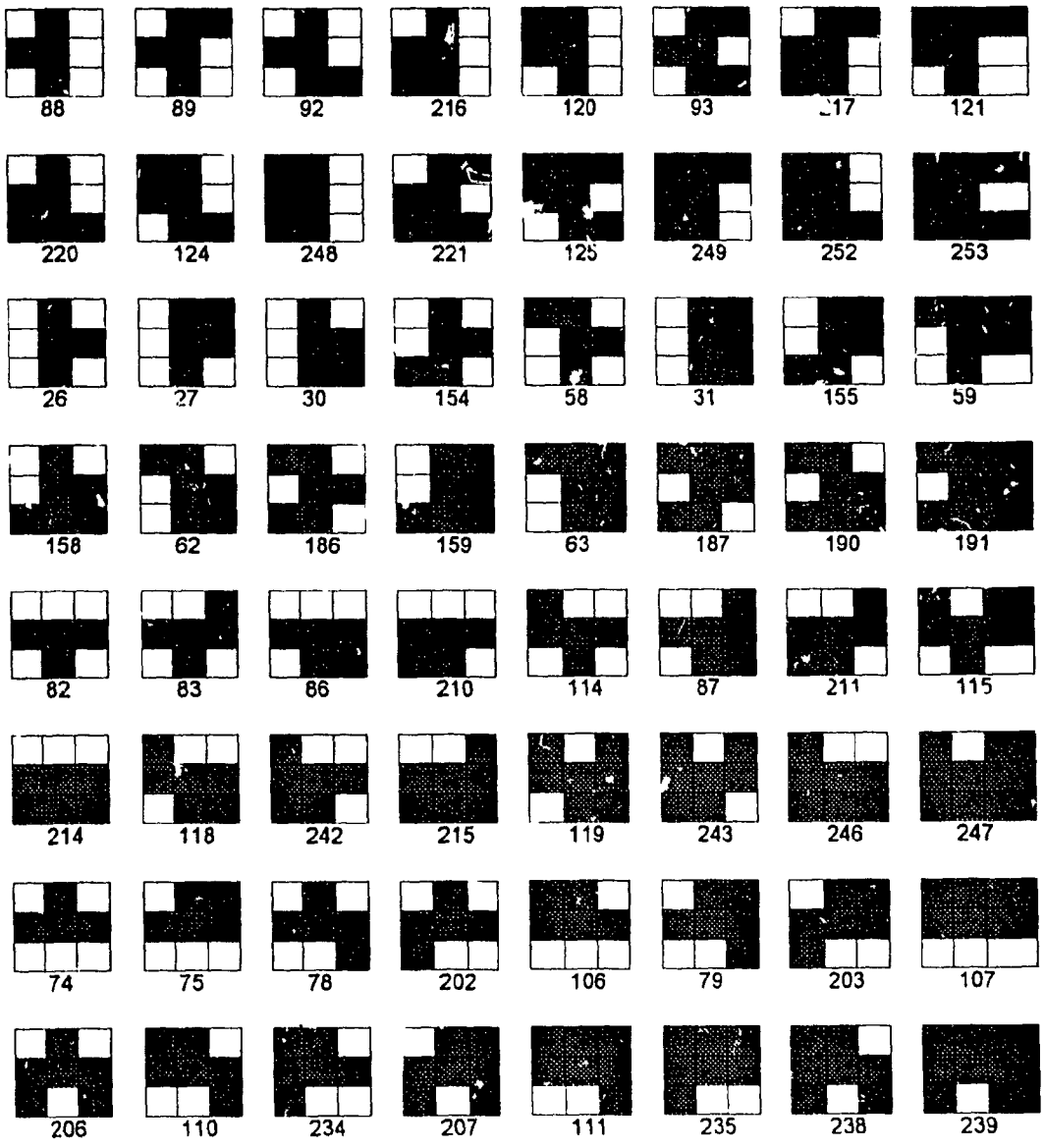
16

Appendix C Delete Templates

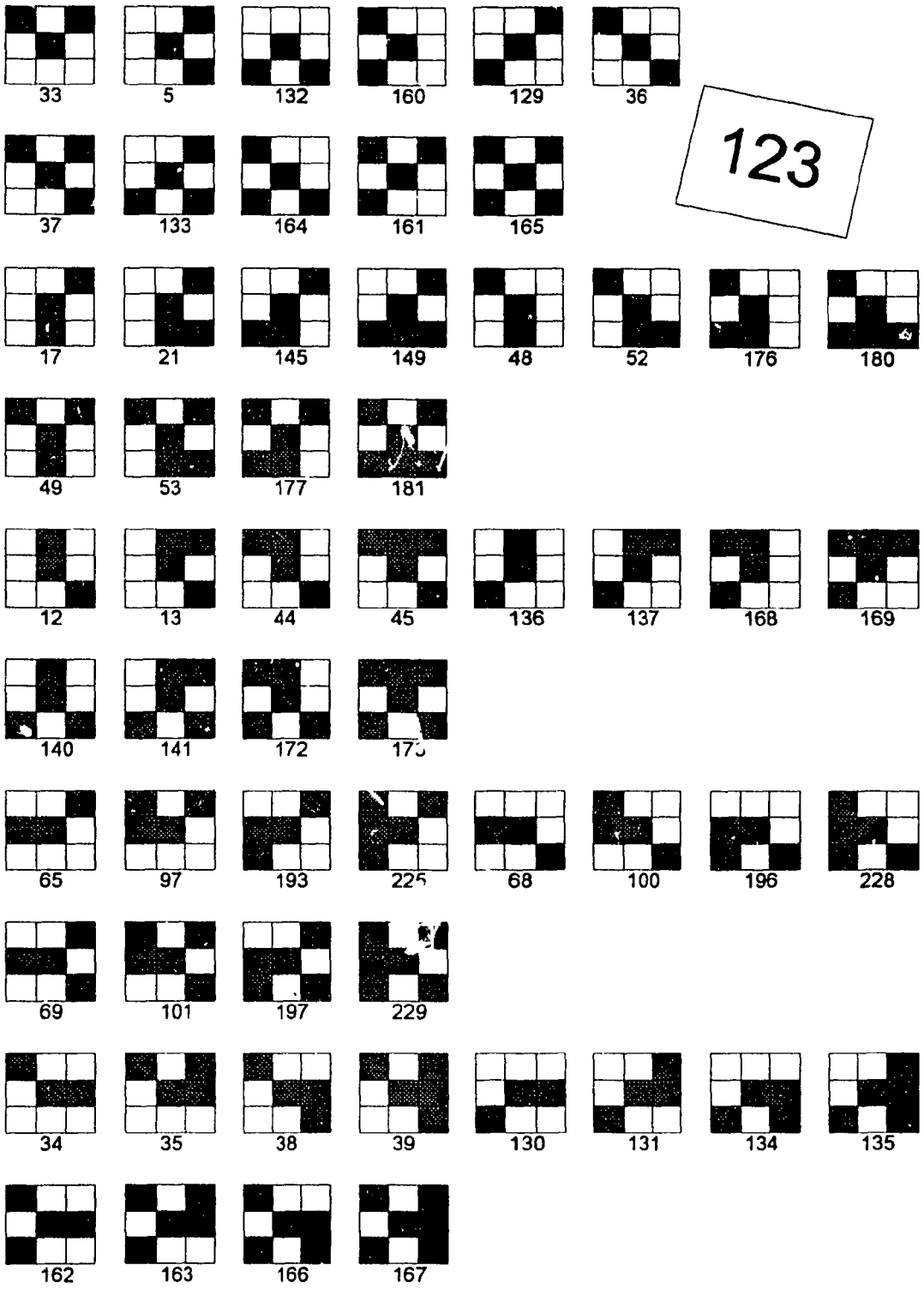


117

Appendix C Delete Templates

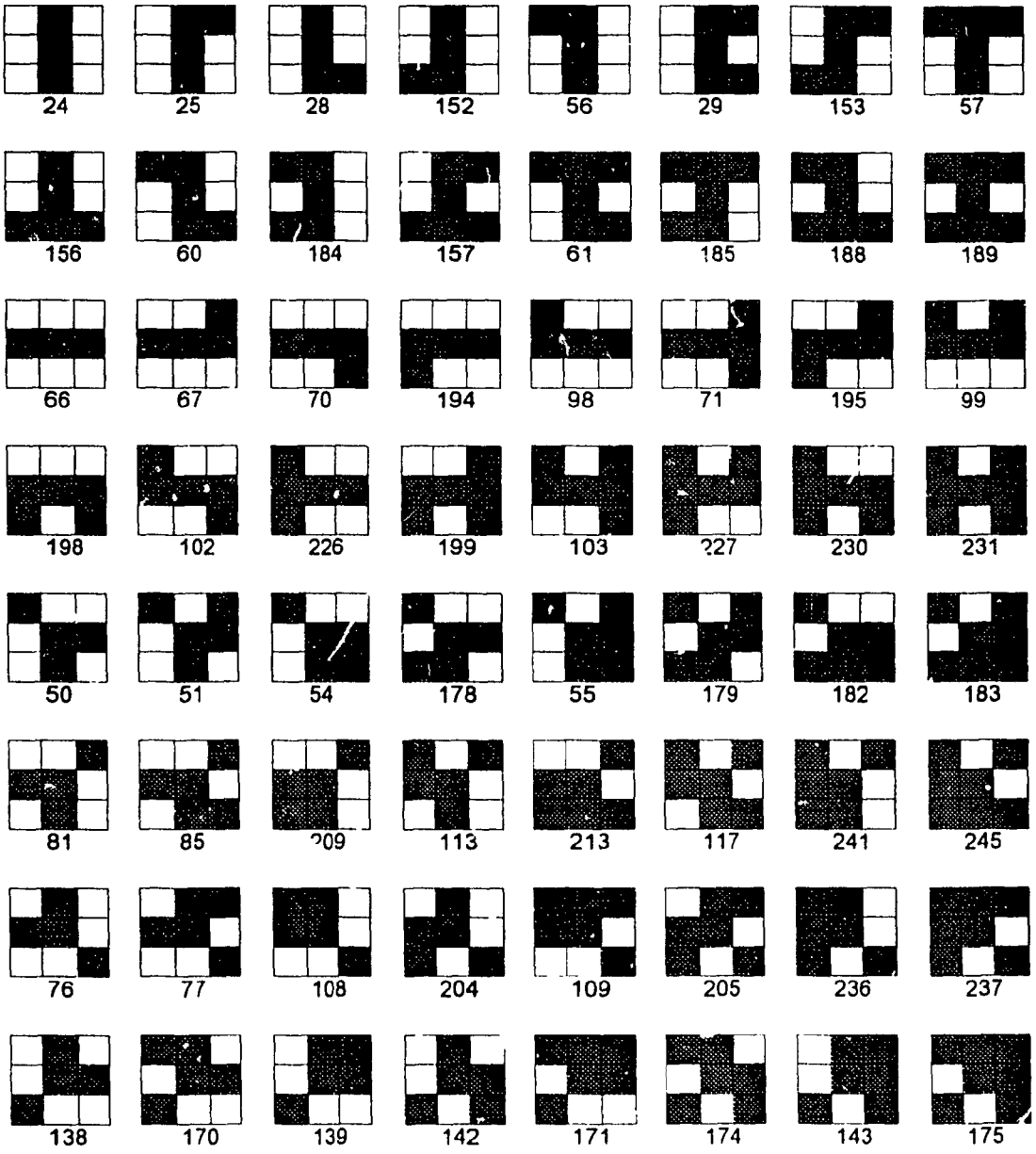


Appendix D Break Templates

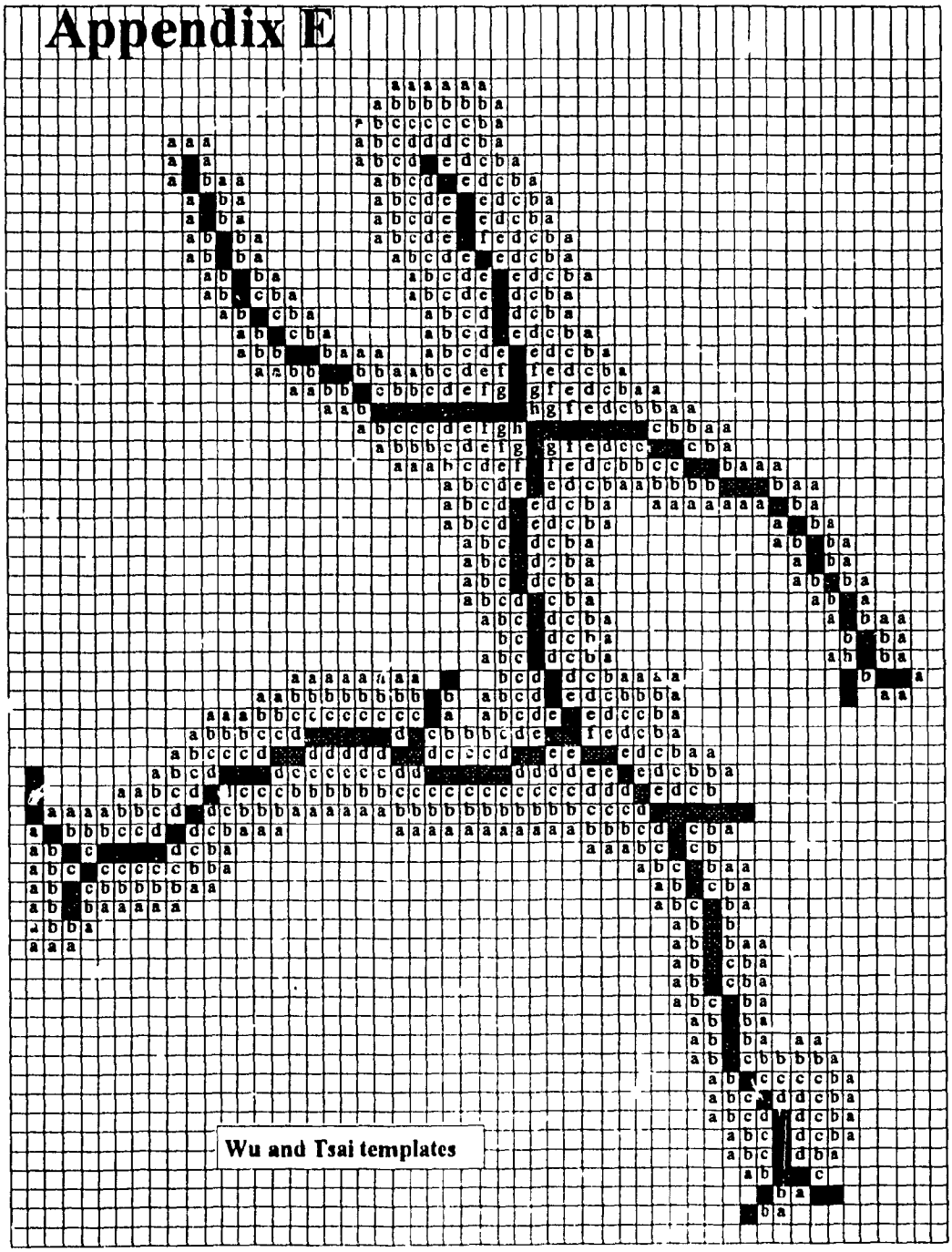


123

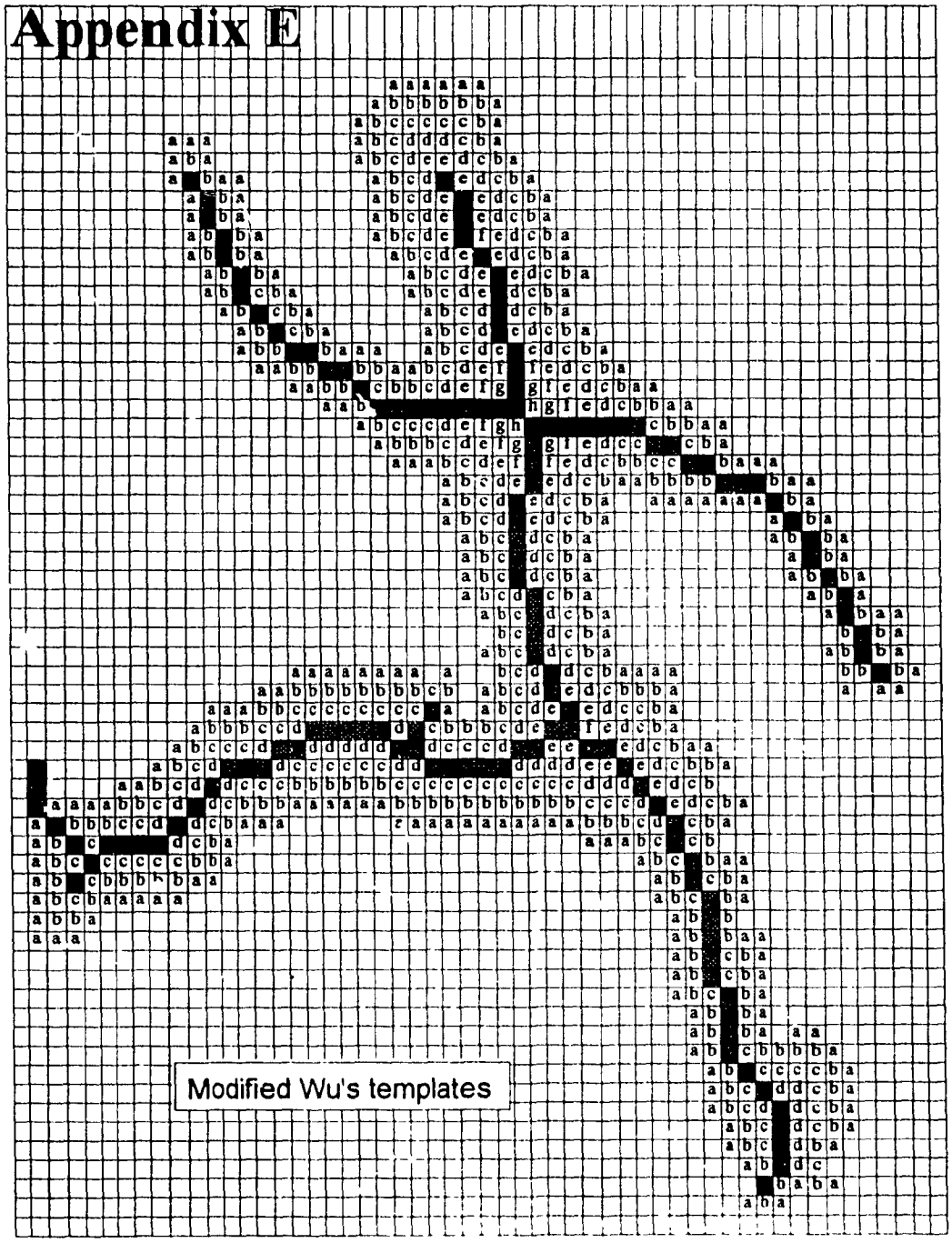
Appendix D Break Templates



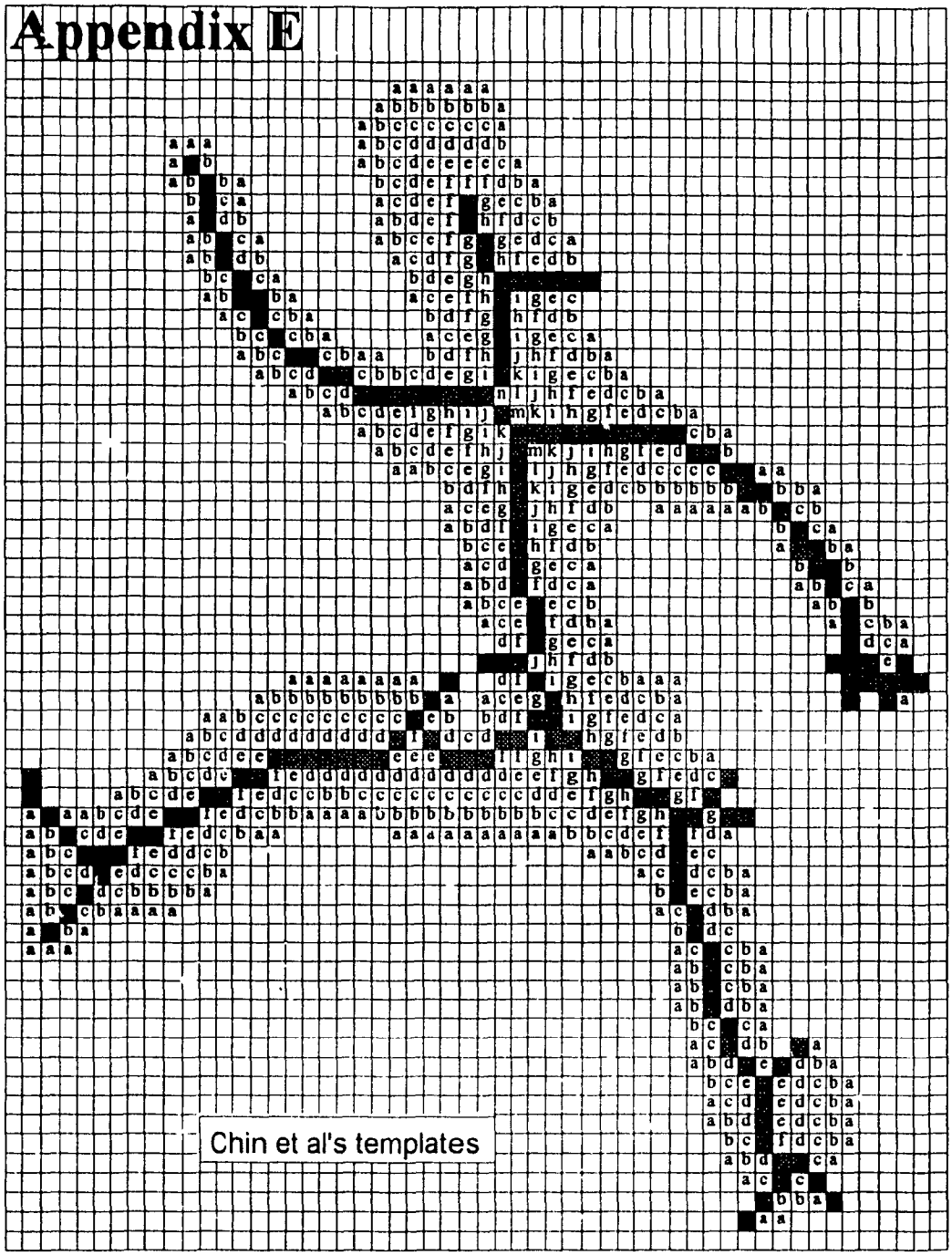
Appendix E



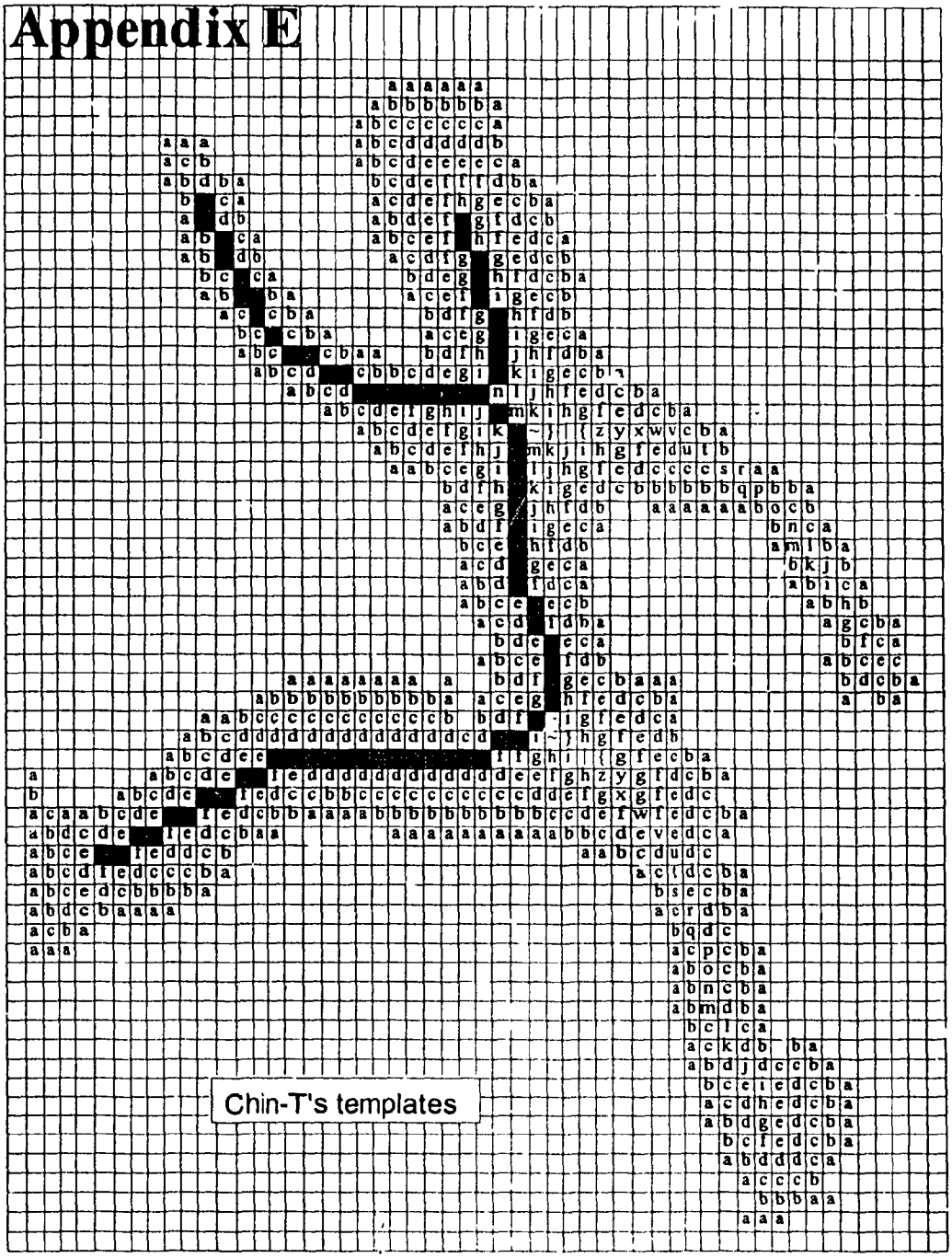
Appendix E



Appendix E



Appendix E



Appendix E