Thinning Methodologies—A Comprehensive Survey

Louisa Lam, Seong-Whan Lee, *Member, IEEE*, and Ching Y. Suen, *Fellow, IEEE*

Abstract—**This article is a comprehensive survey of thinning methodologies. It discusses the wide range of thinning algorithms, including iterative deletion of pixels and nonpixel-based methods, whereas skeletonization algorithms based on medial axis and other distance transforms will be the subject matter of a subsequent study.**

This self-contained paper begins with an overview of the iterative thinning process and the pixel-deletion criteria needed to preserve the connectivity of the image pattern. Thinning algorithms are then considered in terms of these criteria as well as their modes of operation. This is followed by a discussion of nonpixel-based methods that usually produce a center line of the pattern directly in one pass without examining all the individual pixels. Algorithms are considered in greater detail and scope here than in other surveys, and the relationships among them are also explored.

Index Terms—**Parallel thinning, sequential thinning, skeleton, skeletonization, thinning.**

I. Introduction

IN THE EARLY years of computer technology, it was realized that machine recognition of patterns was a possibility, and together with this arose the need for reducing the amount of information to be processed to the minimum necessary for the recognition of such patterns. It seems that the earliest experiments in data compression were conducted on character patterns in the 1950's. In [37], it was found that an averaging operation over a square window with a high threshold resulted in a thinning of the input image, and in [61], the "custer" operation was an early attempt to obtain a thin-line representation of certain character patterns. The thinned characters were used for recognition in [107], [32], and [2]. This practice has been widely used since then (for example, [18], [117], [67], [121], and [69]), and integrated circuits have been designed for this purpose [96].

During these years, many algorithms for data compression by thinning have been devised and applied to a great variety of patterns for different purposes. In the biomedical field, this technique was found to be useful in the early 1960's, when a "shrink" algorithm was applied to count and size the constituent parts of white blood cells in order to identify abnormal cells [58], [94]. Since that time, applications in this area have included analyses of white blood cells [36] and chromosomes [51], [52], automatic X-ray image analysis [95], and analysis of coronary arteries [81]. In other sectors, thinned images have found applications in the processing of bubble-chamber negatives [73], the visual system of an automaton [43], fingerprint classification [74], quantitative metallography [68], measurements of soil cracking patterns [83], and automatic visual analyses of industrial parts [75] and printed circuit boards [133].

This wide range of applications shows the usefulness of reducing patterns to thin-line representations, which can be attributed to the need to process a reduced amount of data as well as to the fact that shape analysis can be more easily made on line-like patterns. The thin-line representation of certain elongated patterns, for example, characters, would be closer to the human conception of these patterns; therefore, they permit a simpler structural analysis and more intuitive design of recognition algorithms. For example, a partially abstract, graph-like skeleton has been considered to be a link between the abstract description of a letter and its physical representation as a character [28]. In addition, the reduction of an image to its essentials can eliminate some contour distortions while retaining significant topological and geometric properties. In more practical terms, thin-line representations of elongated patterns would be more amenable to extraction of critical features such as end points, junction points, and connections among the components (e.g., see [56]), whereas vectorization algorithms often used in pattern recognition tasks also require one-pixel-wide lines as input. Naturally, for a thinning algorithm to be really effective, it should ideally compress data, retain significant features of the pattern, and eliminate local noise without introducing distortions of its own. To accomplish all that using the simplest and fastest algorithm is the challenge involved.

Perhaps as a result of its central role in the preprocessing of data images or perhaps because of its intrinsic appeal, the design of thinning or skeletonization algorithms has been a very active research area. About 300 articles have been published on various aspects of this subject since its inception; therefore, it is felt that a comprehensive survey of this area is due. There have been other comparisons and survey articles in the field ([120], [53], [30], [111]), and some authors have included comparisons with other results in their publications (for example, [122], [77], [26], and [24]). The aim here is to collect most (if not all) of the articles in this field published in English, to classify and discuss them according to the methodologies employed, and to include areas not usually covered in the other surveys—morphological skeletonization, 3-D skeletonization, and grey-scale thinning. All of these will

Manuscript received August 6, 1990; revised June 21, 1991. This work was supported by the Natural Sciences and Engineering Research Council of Canada, the National Network of Centres of Excellence program of Canada, and the FCAR program of the Ministry of Education of the province of Québec. Recommended for acceptance by Associate Editor C. Dyer.

L. Lam and C. Y. Suen are with the Centre for Pattern Recognition and Machine Intelligence, Concordia University, Montreal, Québec, Canada H3G 1M8.

S. -W. Lee is with the Department of Computer Science, Chungbuk National University, Cheongju, Chungbuk, Korea.

IEEE Log Number 9201772.

be discussed under a consistent set of terminologies (where variations will be mentioned) in the hope that such a unified treatment would be helpful.

This article is organized as follows. Section II contains an overview of the thinning process and pixel deleting criteria. In Sections III and IV, sequential and parallel thinning algorithms, respectively, will be discussed, whereas the noniterative nonpixel-based methods will be considered in Section V. This article will conclude with some observations and remarks in Section VI. In a separate follow-up article, we will consider medial axis and other distance transforms, extensions of thinning algorithms to 3-D and grey-scale images, and we will include a brief discussion of morphological skeletonization.

II. Overview of Thinning

The term "skeleton" has been used in general to denote a representation of a pattern by a collection of thin (or nearly thin) arcs and curves. Other nomenclatures have also been used in different contexts. For example, the term "medial axis" is also used to denote the locus of centers of maximal blocks that are also equivalent to the local maxima of chessboard distance [101]. Some authors also refer to a "thinned image" as a line-drawing representation of a pattern that is usually elongated. In recent years, it appears that "thinning" and "skeletonization" have become almost synonymous in the literature, and the term "skeleton" is used to refer to the result, regardless of the shape of the original pattern or the method employed. In this paper, we will observe the general usage of the term "skeleton"; however, we will refer to methods involving determination of centers of maximal blocks as "medial axis transformations," whereas the reduction of generally elongated patterns to a line-like representation will be considered as "thinning." This seems to be a reasonable way to distinguish between different methodologies while recognizing that the results can be quite similar in appearance.

In this section, we consider some general aspects of iterative thinning algorithms or, more precisely, the algorithms that delete successive layers of pixels on the boundary of the pattern until only a skeleton remains. The deletion or retention of a (black) pixel p would depend on the configuration of pixels in a local neighborhood containing p. According to the way they examine pixels, these algorithms can be classified as sequential or parallel. In a sequential algorithm, the pixels are examined for deletion in a fixed sequence in each iteration, and the deletion of p in the nth iteration depends on all the operations that have been performed so far, i.e., on the result of the $(n-1)$th iteration as well as on the pixels already processed in the nth iteration. On the other hand, in a parallel algorithm, the deletion of pixels in the nth iteration would depend only on the result that remains after the $(n-1)$th; therefore, all pixels can be examined independently in a parallel manner in each iteration.

In the following discussion (as in the literature), the terms iteration and cycle are used interchangeably, as are the terms subiteration and subcycle. The term pass has been used to denote either iteration or subiteration, but it will be used to mean subiteration here.

x_4	x_3	x_2
x_5	p	x_1
x_6	x_7	x_8

Fig. 1. Pixels of $N(P)$.

The following notations will be adopted in the discussion below. It is understood that a pixel p examined for deletion is a black pixel, and the pixels in its 3×3 window are labeled as shown in Fig. 1.

The pixels $x_1, x_2, \ldots, x_8$ are the 8-neighbors of p and are collectively denoted by $N(p)$. They are said to be 8-adjacent to p. The pixels x_1, x_3, x_5, x_7 are the 4-neighbors of p and are 4-adjacent to p.

We will use x_i to denote both the pixel and its value 0 or 1, and x_i is called white or black, accordingly. The number of black pixels in $N(p)$ is denoted by $b(p)$. A sequence of points $y_1, y_2, \ldots y_n$ is called an 8-path (4-path) if y_{i+1} is an 8- (or 4-) neighbor of $y_i, i = 1, 2, \ldots, n-1$. A subset Q of a picture P is 8- (or 4-) connected if for every pair of points x, y in Q there exists an 8- (or 4-) path from x to y consisting of points in Q. In this case, Q is said to be an 8- (or 4-) component of P. The order of connectivity of P is the number of components of its complement $\bar{P}$, and if this order is 1, we say that P is simply connected; otherwise, P is multiply connected.

A pixel p is 8- (or 4-) deletable if its removal does not change the 8- (or 4-) connectivity of P. The pixels considered for deletion are contour pixels. It has been suggested [38] that the satisfying duality of P and $\bar{P}$ having different types of connectivity would eliminate the paradoxes of P and $\bar{P}$, being both connected or both disconnected [98]. Since it is desirable for a skeleton to have unit width, the choice would be to adopt 8-connectivity for P and 4-connectivity for $\bar{P}$. This is assumed in the rest of the paper unless otherwise stated, and the reader is referred to the original publications for the (usually) analogous formulations using the other metrics. One consequence of this choice is that connectivity can be preserved by deleting only those pixels of P that are 4-adjacent to $\bar{P}$. Therefore, contour pixels are usually defined as those having at least one white 4-neighbor ([104] and [18] are exceptions). We will assume the common definition of contour pixels, unless otherwise stated, and note that contour points have also been called edge points [77] and border points [62]. Noncontour black pixels are said to be interior points. We will also consider p to be an end point and retained if $b(p) = 1$; this will be referred to as the end point (or end pixel) condition. This condition is applied differently by some authors: p may be retained when there are two or three consecutive black pixels on one side of $N(p)$ [77], the condition may be applied only after the first two iterations [57], or it may be omitted entirely in order to avoid spurious branches [18].

Most of the differences between algorithms occur in the tests implemented to ensure connectedness. This property has been defined in terms of crossing number, connectivity number, and pixel simplicity.

There are two definitions of the crossing number of a pixel. Rutovitz [103] first proposed this useful measure of connectivity as the number of transitions from a white point to a black point and vice versa when the points of $N(p)$ are traversed in (for example) counterclockwise order. Therefore, this crossing number can be defined as

$$X_R(p) = \sum_{i=1}^{8} |x_{i+1} - x_i|$$

where $x_9 = x_1$, and it is equal to twice the number of black 4-components in $N(p)$. Deletion of p would not affect 4-connectivity if $X_R(p) = 2$ since the black pixels in $N(p)$ are 4-connected in these cases. However, since disjoint 4-components can be 8-connected, skeletons obtained using this crossing number can contain 8-deletable pixels, and these skeletons are sometimes said to be imperfectly 8-connected [24] or are more than one pixel wide. In order not to belabor this point, we will assume that this will be understood in later sections and that skeletons obtained this way would usually need postprocessing to replace 4-connectedness with 8-connectedness. For this purpose, various algorithms can be applied, for example, those in [100] or [8].

Hilditch [52] defined the crossing number $X_H(p)$ as the number of times one crosses over from a white point to a black point when the points in $N(p)$ are traversed in order, cutting the corner between 8-adjacent black 4-neighbors. Therefore

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

where

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

and $X_H(p)$ is equal to the number of black 8-components in $N(p)$ except when p has all black 4-neighbors, in which case $X_H(p) = 0$. Obviously, for both definitions of crossing number, a pixel having all black 8-neighbors would have crossing number 0 as would an isolated pixel. If $X_H(p) = 1$, deletion of p would not change the 8-connectedness of the pattern.

Another difference between the crossing numbers $X_H(p)$ and $X_R(p)$ is that the condition $X_H(p) = 1$ would also imply that p must also be a contour point (having at least one white 4-neighbor), whereas $X_R(p) = 2$ does not ensure such a condition since this would be satisfied if p has exactly one white corner neighbor. In order to avoid deleting p in this case (and creating a hole), another condition is needed (for example, $b(p) \leq 6$) to ensure that p is a contour pixel.

An equivalent but more readily computable form of the crossing number $X_H(p)$ is the 8-connectivity number of [134] and [135] defined as

$$N_c^8(p) = \sum_{i=1}^{4} (\bar{x}_{2i-1} - \bar{x}_{2i-1}\bar{x}_{2i}\bar{x}_{2i+1})$$

where $\bar{x}$ implies "not x," whereas the 4-connectivity number

$$N_c^4(p) = \sum_{i=1}^{4} (x_{2i-1} - x_{2i-1}x_{2i}x_{2i+1})$$

represents the number of 4-connected components containing the black 4-neighbors of $N(p)$.

Furthermore, $N_c^8(p)$ is equal to the number of times the pixel p would be traversed in a contour-following algorithm for a connected component [135]; therefore, pixels that are retained (when $N_c^8(p) > 1$) would be pixels that are traversed more than once in the tracing process. At the same time, deletable pixels have often been called simple, and it is proved [99] that in a simply 8-connected pattern, a nonisolated contour pixel p is simple if and only if $N(p)$ has only one black component, which is equivalent to $X_H(p) = 1$.

An alternative approach to preserving topology during thinning is that the genus of P (and $\bar{P}$) should remain invariant [124], where the genus of P is defined as the number of connected components of P minus the number of holes of P. For any pixel p, its effect on the genus G can be determined completely from the configuration of $N(p)$. If deletion of p does not change G, p is said to be "simple." Since there are 256 configurations of $N(p)$, these combinations can be examined, and the ones where p is simple can be stored in a look-up table. It can be readily verified that the concept of "simple" based on genus is equivalent to that based on connectivity.

Pixels with connectivity number $N_c^8(p)$ greater than one also belong to the category of multiple pixels [85]. These pixels are considered to occur where a pattern "folds" onto itself, and they include end points of branches, strokes that are two pixels in width, and pixels that should be assigned to the skeleton on the basis of the connectivity criteria. Consequently, these pixels are retained in the thinning process. This notion will be considered in more detail when the algorithms that utilize them are discussed.

According to their modes of operation and the pixel testing criteria used, many thinning algorithms can be broadly classified according to the scheme shown in Table I. Under this general scheme, sequential algorithms can operate by processing only contour pixels or by raster scanning and parallel algorithms by using 4, 2, or 1 subiterations. Both classes of algorithms can ensure connectedness by finding the crossing numbers $X_R(p)$ or $X_H(p)$ by matching against thinning windows, by deleting only simple pixels, or by retaining multiple pixels. These procedures, as well as other less widely used methods, will be discussed more fully in the next two sections.

III. Sequential Thinning Algorithms

Using sequential algorithms, contour points are examined for deletion in a predetermined order, and this can be accomplished by either raster scan(s) or by contour following.

TABLE I
GENERAL CLASSIFICATION SCHEME

OPERATION		Pixel Testing Criteria			
		$X_R(p)$	$X_H(p)$	Window matching	Multiple/simple pixels
SEQUENTIAL	Contour Pixels	Arcelli [5] Wang [129]		Beun [18] Pavlidis [86] Chu [26]	Pavlidis [85] Arcelli [10]
	Raster Scanning	Arcelli [8]	Hilditch [52] Yokoi [134]		Arcelli [14]
PARALLEL	4-subcycle		Rosenfeld [100] Hilditch [53]	Stefanelli [115]	Arcelli [6]
	2-subcycle	Deutsch [35] Zhang [136] Chen [21]	Suzuki [118] Guo [49]	Stefanelli [115]	
	1-subcycle	Rutovitz [103] Holt [54]	Chen [22]	Chin [25]	

Contour following algorithms can visit every border pixel of a simply connected object [99], and of a multiply connected picture, if all the borders of the picture and holes are followed [134]. Therefore, such methods, which have been utilized previously without proof, have been shown to be valid in this sense. Border following algorithms are also given in these papers. These algorithms have an advantage over raster scans because they require the examination of only the contour pixels instead of all the pixels in P and $\bar{P}$ in every iteration. Some algorithms that use contour tracing are seen in [10], [85], [26], [126], [130], and [47], and the contours are traced using the Freeman chain codes in [64].

When a contour pixel p is examined, it is usually deleted or retained according to the configuration of $N(p)$. To prevent sequentially eliminating an entire branch in one iteration, a sequential algorithm usually marks (or flags) the pixels to be deleted, and all the marked pixels are then removed at the end of an iteration. This generally ensures that only one layer of pixels would be removed in each cycle.

To avoid repetition in the following discussion, we will assume that a pixel p considered for deletion satisfies all the following properties unless otherwise stated:

1. p is a black pixel.
2. p is not an isolated or end point, i.e., $b(p) \geq 2$.
3. p is a contour pixel, i.e., p has at least one white 4-neighbor.

A seminal algorithm in sequential thinning is that of Hilditch [52], which utilized the crossing number $X_H(p)$. The pattern is scanned from left to right and from top to bottom, and pixels are marked for deletion under four additional conditions that were also stated explicitly in [78] and [111] but are included here for completeness:

H1: At least one black neighbor of p must be unmarked.
H2: $X_H(p) = 1$ at the beginning of the iteration.
H3: If x_3 is marked, setting $x_3 = 0$ does not change $X_H(p)$.
H4: Same as H3, with x_5 replacing x_3.

Condition H1 was designed to prevent excessive erosion of small "circular" subsets, H2 to maintain connectivity, and H3-H4 to preserve two-pixel wide lines. The author also extended the method to thinning of grey-scale chromosome images.

This algorithm has also been implemented for binary images by other researchers using various techniques. Cellular logic operations based on 3×3 neighborhoods are used in [48]; since a change in the value of p may induce a change only in the pixels of $N(p)$ in the next iteration, only these pixels need to be processed. In [80], it was found that a sequential thinning method based on raster scan and 3×3 operations can be implemented using a pipeline structure to reduce the memory and processing time required. In [91], connected horizontal runs of black pixels are coded in interval tables (the use of pointers in these tables implies that only a change of pointers would be required when a shift operation is used on local neighborhoods). The image that remains after one iteration is found by inspection of the overlapping intervals in the preceding, current, and next lines of the image. Alternatively, contour pixels can be queued, with the change of a contour pixel causing all its 4-neighbors to be queued and processed in the next iteration [126]. The crossing number $X_H(p)$ together with a rough estimate $K(p)$ of the convexity at p are used as criteria for deletion of p in [97]. $K(p)$ is the maximum number of 4-connected white points in $N(p)$ and is intended as a discrete equivalent for the notion of curvature. Under this scheme, p is deleted if $K(p) < K_T$ for some threshold K_T, and conditions H2-H4 hold. Increasing the threshold K_T results in skeletons less affected by contour protrusions (at the risk of excessive erosion).

The thinning criteria of [52] are extended to $k \times k$ windows ($k \geq 3$) in [84], where the center "core" of $(k-2) \times (k-2)$ pixels can be deleted together if the boundary pixels in the window have Hilditch crossing number 1 and if they contain more than $(k-2)$ 4-connected white pixels and more than $(k-2)$ black pixels. For every black pixel, its $k \times k$ windows are examined in the order of decreasing k until $k < 3$ or the core is deleted. With larger values of k, thicker layers of pixels can be deleted in one iteration; therefore, fewer iterations would be required to obtain skeletons of thick patterns. However, this increase in speed is achieved at the expense of "coarser" results (noisy skeletons); in the thinning of elongated patterns, the use of a larger k can be detrimental to processing speed. The same thinning algorithm has also been implemented in parallel in this work by using four subiterations and examining one type of border pixels (north, south, east, or west) in each subiteration.

In another early algorithm [134], every black pixel p is examined and labeled according to the rule

$$L(p) = \begin{cases} 2 \text{ if } x_3 = 0 \text{ or } \overline{x}_3 + \overline{x}_5 + x_7 = 0, \\ 3 \text{ if } \overline{x}_3 + x_5 = 0 \text{ or } \overline{x}_3 + \overline{x}_5 + \overline{x}_7 + x_1 = 0. \end{cases}$$

Then, two raster scans in opposite directions are performed to remove pixels labeled 2 and 3, respectively, provided these pixels are not end points and have connectivity number 1 (where 4- or 8-connectivity can be used). Although the algorithm is simple and yields connected skeletons, vertical strokes that are an even number of pixels wide and open at one end can be completely eliminated.

In [18], the different criteria of [104] are used for contour or edge points. For example, p is a west edge point if there is at most one black pixel in the first column of $N(p)$ and at least 3 black pixels in the rest of $N(p)$. In this case, an extra

(a) (b)

Fig. 2. Break point configurations; at least one pixel in each group marked x or y must be nonzero.

condition had to be imposed to prevent an interior pixel from being marked as an edge point. Edge points are marked in one raster scan after which the marked points are compared against six 3×3 windows (those shown in Fig. 2 and their 90° rotations) and deleted if they are not break points, i.e., their removal does not create breaks in the pattern. It was found that elimination of the end-point condition resulted in less spurious tails. The procedure is repeated until no pixels are deleted or for a preset maximum of three iterations, after which a single "clean-up" scan is applied to remove all black pixels that are not end points, break points, or interior points. The results obtained from this procedure were not very different from those of thinning carried through to the end, probably because this algorithm was designed and tested on (almost thin) character patterns.

The preceding algorithm assumes a smoothing preprocessing phase; this is extended in [26], where smoothing is performed before each iteration. In each iteration, contour pixels satisfying the usual stated conditions are marked and then examined for deletion as in [18] but with the endpoint condition. When no more pixels can be deleted, a final adjustment phase is introduced in which a skeletal pixel would be moved to one of its 4-neighbors if the latter pixel has a greater 8-distance from the background. In the process, connectivity of the skeleton is maintained while skeletal points are moved closer to the medial line of the original pattern.

The classical thinning algorithms mentioned in [87] also use the windows shown in Fig. 2 together with their 90° rotations to determine skeletal pixels. (For the sake of brevity, we use 90° rotations to also include 180° and 270° rotations when different windows are generated). Since these windows, in effect, only represent connectivity preservation criteria, their sequential application to a pattern could result in excessive erosion or serious shortening of branches. To overcome this difficulty, a mixture of parallel and sequential operations is proposed. Only one type of contour point is processed in each subiteration, skeletal pixels are marked for retention through all subsequent iterations, and a pixel q to be deleted is marked so that it is considered to be a black pixel when its neighboring pixels are checked in the current iteration. These conditions are designed to preserve connectivity and prevent disappearance of two-pixel wide lines, and they also retain end points without requiring a specific condition.

A variation of this algorithm is based on the compilation of an extensive list of thinning rules and consideration of 5×5 neighborhoods [42]. These larger neighborhoods are also used in [69], where the pattern is raster scanned, and deletion of the center pixel is based on comparison with 20 25-b templates stored on a chip.

The SPTA [77] is a sequential algorithm that uses two raster scans per cycle, where the first is left to right, and the second is top to bottom. In each scan, p is marked for retention (p is a safe point) if one of the following is true:

N1: $N(p)$ satisfies one of the windows in Fig. 2 or its rotations.

N2: $N(p)$ contains exactly two 4-adjacent black points.

These conditions are equivalent to a west contour point p being safe if the boolean expression

$$x_1(x_2 + x_3 + x_7 + x_8)(x_3 + \overline{x}_4)(x_7 + \overline{x}_6) = 0.$$

In the first scan, west contour points are marked, then east contour points that are not west safe points, and so on. Condition N2 is intended to prevent excessive erosion of diagonal strokes two pixels wide, but it can lead to noisy branches [65]. In [16], a modification was proposed to SPTA to eliminate (in most cases) the last pass when no pixels would be deleted, and the modified algorithm was implemented on a multiprocessor using the methods of a) function and b) data decomposition. In a), each processor executes one scan, and each completed row is moved to the processor performing the next scan, whereas in b) each processor scans a portion of the image with overlapping boundaries.

The Rutovitz crossing number $X_R(p)$ is used to determine pixel deletion in [5], [7], and [8]. In these algorithms, a slightly different definition of contour pixel is used; here, a contour pixel is a black pixel having at least one white 8-neighbor. This condition together with the use of $X_R(p)$ require an additional condition ($F = x_1x_3x_5x_7 = 0$) to ensure that holes would not be created when contour points are deleted. Complete conditions for deletability of p while maintaining connectivity are given in [5]; briefly, these useful conditions are as follows:

1. If $X_R(p) = 0$ or 8, p is not deletable.
2. If $X_R(p) = 2$, p is deletable iff $F = 0$ and p is not an end point.
3. If $X_R(p) = 4$, p is deletable iff $F = 0$ and one of the four corner pixels is 0 with 1's on both sides. The latter condition is equivalent to $\sum_{i=1}^{4} x_{2i-1}\overline{x}_{2i}x_{2i+1} = 1$.
4. If $X_R(p) = 6$, p is deletable iff one of its 4-neighbors is 0 and the other three are 1's belonging to distinct 4-components, or $\sum_{i=1}^{4} x_{2i-1} = 3$.

However, use of the above conditions by themselves would give spurious end points and erode corners. Therefore, a solution is proposed in [5] to test p for deletion according to the configuration of contour pixels in $N(p)$. A "second level" crossing number $CNN(p)$ represents the number of transitions from a contour pixel to a noncontour one (and vice-versa) when the points of $N(p)$ are traversed in order, and procedures are derived for the deletion of p based on $X_R(p)$, $CNN(p)$, and the number of 8-adjacent pairs of contour pixels among the 4-neighbors of p.

Further work in preserving significant contour protrusions or prominences in the thinning process are developed by Arcelli

and Sanniti di Baja [7], [8], where prominences are first detected and labeled. These significant protrusions are defined as connected subsets of the contour that are beyond a threshold distance from the interior or core (the noncontour pixels of P). They are retained while the other contour pixels with $X_R(p) = 2$ are removed iteratively according to the conditions in [5] as long as they are not necessary for maintaining connectedness between the core and the prominences. When a set S_f with an empty core is obtained, a label $e(p) = 2(x_5+x_3)+x_1+x_7+1$ is assigned to each remaining pixel p in S_f. The nonmaxima pixels under this label are removed, resulting in a skeleton that is, at most, two pixels wide; then a final pass can be made to destroy 4-connectedness in favor of 8-connectedness.

The skeletons obtained this way can be unduly influenced at the end of a branch by the presence of a sharp protrusion on one side (see Fig. 21a in [3]). In [3], a compression phase was also implemented to represent each 3×3 window of P by a grey-scale value derived from the number of black pixels in the window. The reduced grey-scale image is then thresholded with the connectivity constraints of Fig. 2 to a binary image on which thinning is performed by an implementation of [8] modified to preserve T junctions with length greater than one pixel [109]. The skeleton is then expanded to the original scale.

When contour pixels p are traced sequentially and labeled, multiple pixels, where a pattern "folds" onto itself, can be easily determined [85]. Pixel p is multiple if at least one of the following conditions holds:

P1: p is traversed more than once during tracing (connectivity number > 1).

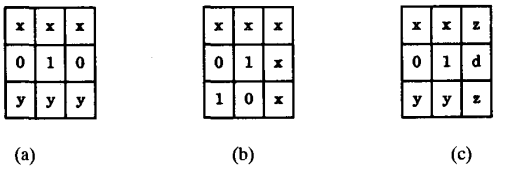
P2: p has no neighbors in the interior.

P3: p has at least one 4-neighbor that belongs to the contour but is not traced immediately before or after p.

Since P1 includes pixels with a connectivity number greater than 1, P2 includes end points, and P3 contains lines two pixels wide, the concept of multiple pixels is quite inclusive. However, if the contour is traced repeatedly and only the multiple pixels from every tracing are retained, the result may not be a connected skeleton. Therefore, in [85], multiple pixels are called skeletal, as are 8-neighbors of skeletal pixels from a previous iteration. These pixels form a skeleton that may be too thick (and thus requires editing), but the algorithm was proved to be correct—it does terminate and produces connected skeletons.

In [86] and [87], the characterization of multiple pixels is redefined in terms of local neighborhoods and can therefore be determined in sequence or parallel by comparison against a set of masks. This requires the addition of window (c) to those of Fig. 2 to produce the neighborhood patterns of Fig. 3 and their 90° rotations for multiple pixels.

The definition of multiple pixels is slightly modified in [88]. It is also proposed that a combination of sequential and parallel operations may be more efficient for images where most of the pixels do not require much processing. For such images, the pattern can be divided into fields where each is assigned to a processor. Each processor operates on the pixels of its field sequentially, and when certain steps have been completed, it waits until all other processors have completed the same steps so that the processors can be resynchronized.



(a) (b) (c)

Fig. 3. Configurations of multiple pixel p. Each group of pixels marked x or y must contain a nonzero element. In (c), at least one of the pixels marked z must be nonzero; if they are both nonzero, pixels marked x or y can have any value. The pixel marked d is a contour pixel.

The above algorithm essentially ensures skeleton connectedness by detecting and assigning multiple pixels M to the skeleton S and then finding and assigning to S suitable (skeletal, nonmultiple) pixels to connect M to the interior of P. Since this may result in unacceptably thick skeletons when P is not initially almost thin, an alternative algorithm that deletes the nonmultiple pixels from the contour C and retains the remaining set as S is proposed [10]. At the same time, contour information is used to determine whether a contour pixel should be a) regarded as noise and not labeled multiple or b) considered to be a significant convexity and assigned to S even though it is not multiple. These are accomplished by computing the n codes [45] of contour pixels from the Freeman chain codes obtained in contour tracing. The 1-code c_i of pixel p_i is the difference (modulo 8) between the chain codes at p_i, and for $n > 1$, the n code

$$c_i^n = nc_i + \sum_{k=1}^{n-1} (n-k)(c_{i-k} + c_{i+k})$$

determines the curvature of the contour at p_i. The value of c_i is used to determine a), and if c_i^n exceeds a threshold, then p_i is assigned to S so that it can represent a significant convexity. Naturally, if the threshold value is lower, the algorithm would be more sensitive to contour protrusions. Other works to preserve such prominences have already been discussed [7], [8].

In [11]–[13], the concept of multiple pixels is developed from another point of view; it is thought of as being the opposite of curve simplicity. In the continuous plane, a (closed) curve is simple if and only if it never crosses itself; consequently, a simple curve divides the plane into two connected sets called the inside and the outside. This notion is extended to the contour C of a digital pattern P for a simply connected pattern in [11] and [12] and for a multiply connected figure in [13]. In particular, C is considered to be simple, provided that it neither touches nor overlaps itself, and this global concept was found to be equivalent to the following local conditions. If we consider $P - C$ to be the inside of C and $\bar{P}$ to be the outside of C, then C is simple, provided that every p in C satisfies the following conditions:

A1: $N(p) \cap \bar{C}$ consists of one (8-connected) component on the inside and one (4-connected) component on the outside.

A2: $N(p) \cap \bar{C}$ contains at least two pixels that are horizontally or vertically aligned: one belonging to the outside and the other to the inside.

The pixels satisfying the conditions A1 and A2 are said to be regular, and those that are not regular coincide with multiple pixels where contour arcs either coincide or are adjacent to each other. Furthermore, since A1 is computationally complicated, it is shown in [13] that equivalently to A1 and A2, a contour pixel p is multiple if it satisfies at least one of the following conditions:

A3: Neither the horizontal nor the vertical neighbors of p are such that one belongs to $P - C$ and the other to $\bar{P}$.

A4: $N(p)$ has three consecutive points such that the intermediate one is a diagonal neighbor and belongs to C, whereas the other two belong to $\bar{P}$.

Conditions A3 and A4 have the advantage of being local conditions that are easily verified once the contour pixels are located, and this is the method employed in [14]. By testing for multiple pixels on the 4-distance transform of P, each successive contour is denoted by its label on the distance transform; therefore, thinning can be accomplished in one raster scan. During this pass, however, the already-visited neighbors of a detected multiple pixel q must be examined again to verify whether any of them has been induced to become a multiple pixel by the constraint to maintain connectedness. The skeleton consists of all the multiple pixels detected, and it can be reduced to unit width.

The Freeman chain codes are also used in [47] to detect features such as 90 and 45° corners and T junctions in the contours. This information is then used to retain certain pixels in the contour-stripping process in an attempt to preserve such features. Otherwise, this method deletes the contour pixel p if $X_H(p) = 1$ and $2 \leq b(p) \leq 6$.

In [132], the four types of contour pixels (east, north, west, and south) are placed in buffers. Each type of buffer point is sequentially processed and checked against windows for connectedness and end-point preservation. If a point is removable, its 4-neighbors are examined for inclusion in the next contour and placed in the appropriate buffers. The process is repeated until the buffers are empty.

The algorithms discussed so far are based on an examination of contour pixels for deletion or retention. A different implementation method to produce a skeleton is that of contour generation or the iterative generation of a new contour inside the existing one until only a skeleton remains. This process is based on the direction of the contour pixels in [131] and [64]. When contour pixels are followed in sequence, three such consecutive pixels would form an angle θ with its vertex at the current pixel p. The interior pixel in $N(p)$ closest to the bisector of θ is considered to be a point on the next contour [131], and p is deleted. When this procedure is repeated until no interior pixels are left, a pseudo skeleton is formed by the last contour. This simple method is greatly refined and expanded in [64] by using the Freeman chain codes of the contour pixels to generate the new contours. These chain codes are used, together with breakpoint and endpoint considerations, to derive a set of rules for the generation of the new contour. When new contour pixels are determined, their chain codes are also generated. The end pixel conditions of algorithms [77], [6], and [136] can be incorporated into this algorithm [65] to simulate the behavior of the previous algorithms, at least at branch ends. This algorithm can also be implemented in a distributed environment [66] by assigning nonoverlapping subsets of the pattern to different processors for thinning and then synchronizing information about the borders at the end of each iteration. A similar procedure of successive contour generation using chain codes is implemented in [127]. In this recent work, a contour pixel is deleted if its 4-neighbors on the "inward" side (away from the background) are all black, in which case, the chain codes for the corresponding section of the new contour are derived from a predefined set of rules related to local curvature. These rules are simpler than those of [64], and it is claimed that the same results are obtained when pixels are processed in the same order.

In general, when the pixels of P are processed sequentially, there is no problem in preserving the connectivity of P and $\bar{P}$ when suitable 3×3 local operations are used. Therefore, the requirement of topological preservation is met by these algorithms. However, it would require more global information to preserve the geometric features that are significant for a shape analysis of P. For this purpose, a skeleton obtained from a raster scan of P is seldom meaningful [5], whereas an algorithm based on sequential following of the contour elements may lead to better results. This latter method can allow some correlation between the shape of the skeleton and the external contour of the pattern that may not be possible to achieve by local operations only. Of course, these more global considerations would lead to increases in computation time and more complicated procedures, and much of the complexity in sequential algorithms (for example, [8], [10] and [85]) is the result of efforts made to preserve more subtle geometric features.

IV. Parallel Thinning Algorithms

In parallel thinning, pixels are examined for deletion based on the results of only the previous iteration. For this reason, these algorithms are suitable for implementation on parallel processors where the pixels satisfying a set of conditions can be removed simultaneously. Unfortunately, fully parallel algorithms can have difficulty preserving the connectedness of an image if only 3×3 supports are considered; for example, a horizontal rectangle two pixels in width may completely vanish in such a thinning process. Therefore, the usual practice is to use 3×3 neighborhoods but to divide each iteration into subiterations or subcycles in which only a subset of contour pixels are considered for removal. At the end of each subiteration, the remaining image is updated for the next subiteration. Four subcycles have been used in which each type of contour point (north, east, south, and west) is removed in each subcycle [115], [101], [17]. These have also been combined into two subiterations [115], [136], [21], [118], [49] with one subiteration deleting the north and east contour points and the other deleting the rest, for example. Other two-subcycle algorithms have been devised to operate on alternate subfields of the pattern that is partitioned in a checkerboard manner. Recently, one-subiteration algorithms have been implemented, but these invariably have to use information from a larger

context in order to preserve connectivity. These algorithms will be examined in greater detail below.

A fundamental parallel algorithm was proposed in [103], where a pixel p is deleted iff all the following are true:

R1: $b(p) \geq 2$
R2: $X_R(p) = 2$
R3: $x_1x_3x_5 = 0$ or $X_R(x_3) \neq 2$
R4: $x_7x_1x_3 = 0$ or $X_R(x_1) \neq 2$.

This algorithm is also described in [115] with the added condition that $b(p) \leq 6$ to ensure that p has a white 4-neighbor; therefore, deletion of p would not create a hole. This is a one-subiteration algorithm that uses information from a 4×4 window, and it does yield connected skeletons that are insensitive to contour noise [115] but can result in excessive erosion [116], [79], [70], [39].

Since disjoint 4-components may be 8-connected, the removal of all pixels satisfying the above conditions does not reduce diagonal lines to unit pixel width. Additional conditions were added specifically to address this problem and to allow for deletion of pixels p with $X_R(p) = 4$ when p lies on a diagonal line two pixels wide [33], and this was proved to preserve connectedness [34]. Due to the asymmetric nature of conditions R3 and R4, the skeleton would not lie centrally; therefore, 180° rotations of these rules were introduced [35] to result in the following complete set of rules for the removal of p:

D1: $X_R(p) = 0, 2, \text{or } 4$
D2: $b(p) \neq 1$
D3: $x_1x_3x_5 = 0$
D4: $x_1x_3x_7 = 0$
D5: If $X_R(p) = 4$, then in addition, a) or b) must hold:
a) $x_1x_7 = 1, x_2 + x_6 \neq 0$, and $x_3 + x_4 + x_5 + x_8 = 0$
b) $x_1x_3 = 1, x_4 + x_8 \neq 0$, and $x_2 + x_5 + x_6 + x_7 = 0$
D6-D8 are 180° rotations of D3-D5.

It was also suggested that two subiterations should be used, where the first one deletes pixels satisfying D1-D5, and the second deletes according to D1, D2, and D6-D8. Of course, this algorithm deletes isolated pixels. The complexity of the rules D1-D5 led to the observation [44] that if two consecutive 4-neighbors of p are 1's, then the value of the corner pixel q in between has no effect on whether p should be deleted; therefore, q can be considered to be 1. In such cases, p can be deleted if its modified crossing number is 2 provided $b(p) > 1$. It should be noted that this modification of the crossing number is the corner-cutting process in [52]; therefore, the resulting number is actually twice $X_H(p)$, and, of course, it measures 8- rather than 4-connectivity.

The crossing number $X_R(p)$ is also used as a criterion for a filling operation before thinning. In [123], a white pixel q with $X_R(q) \geq 2$ is considered to be a connecting point and is filled. Then, a 4-subiteration algorithm is implemented in which end points (having exactly one black 4-neighbor) and pixels p with $X_R(p) > 2$ are considered skeletal, whereas nonskeletal contour pixels are deleted.

The much-cited work of Zhang and Suen [136], which is also summarized in [46], is an implementation of a subset of conditions D1-D8 in two subiterations. In the first subiteration, p is deleted if it satisfies the following conditions:

Z1: $2 \leq b(p) \leq 6$
Z2: $X_R(p) = 2$
Z3: $x_1x_3x_7 = 0$
Z4: $x_1x_7x_5 = 0$.

In the second subiteration, Z3 and Z4 are replaced by their 180° rotations. Therefore, the first subcycle deletes 4-simple pixels on the south and east borders as well as north-west corner pixels, whereas the second subcycle deletes pixels with opposite orientations. This is a simple and efficient algorithm that is also immune to contour noise [21]; however, two-pixel-wide diagonal lines can be seriously eroded, and 2×2 squares would vanish completely [70], [71]. It was suggested that condition Z1 should be replaced by $3 \leq b(p) \leq 6$ in order to retain such structures. As can be expected, such a modification creates problems of its own in retaining extraneous pixels; therefore, an additional pass was proposed [128] in order to thin to unit thickness, further increasing the computation time.

Although excessive erosion of diagonal lines is not a topological problem, complete disappearance of a 2×2 square in the thinning process renders the algorithm invalid from the topological point of view. This was shown [39] to be the only possible configuration where the algorithm is not valid, and a modification to [136] was suggested so that all conceivable cases could be correctly processed. The modification depends on the number of black 4-neighbors of p when $X_R(p) = 2$. If p has none or one such neighbor, no change would be required. If p has two or three such neighbors and p satisfies the existing conditions, then p can be deleted in the first subiteration if $x_1 = 0$ or $x_7 = 0$ and in the second if $x_3 = 0$ or $x_5 = 0$.

In [129] and [130], conditions D1, D2, and D8 from [35] are implemented sequentially with minor differences. The more recent algorithm does not delete isolated pixels ($X_R(p) = 0$). It requires $2 \leq b(p) \leq 6$, and it does not preserve 4-connectivity. Therefore, a pixel p at the vertex of a right angle formed by two 4-neighbors would be deleted, thus leading to condition $x_2 + x_6 \neq 0$ being unnecessary in rule D8(a) and analogously for D8(b). These authors also implemented a two-subcycle parallel algorithm that deletes (in the first subiteration) east or south boundary points and northwest corner points with $X_R(p) = 2$ as well as contour pixels with $X_R(p) = 4$ satisfying condition D8; pixels with opposite orientations are deleted in the second subiteration. Another two-subiteration implementation of conditions D1–D8 of [35] is that of [21]; it contains the same minor differences as [129] and [130], and it claims to be an improvement over [136] in eliminating excessive erosion. In this work, the rules are implemented by table lookup; each of the 256 configurations of $N(p)$ is converted into an address in the thinning tables (one for each subiteration), where the new value of p (0 or 1) is defined.

In order to use only one subiteration per cycle, information from a 4×4 window containing p is considered [54] in order to determine its deletion. For this algorithm, p is an edge point (edge p is true) iff $N(p)$ contains between two and six 4-connected black pixels, i.e., $X_R(p) = 2$ and $2 \leq b(p) \leq 6$.

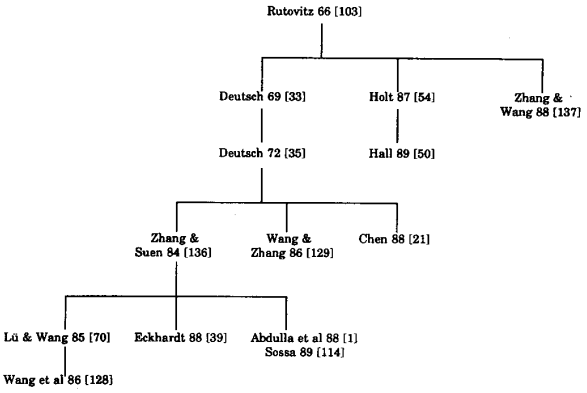


Fig. 4. Family tree of thinning algorithms based on Rutovitz.

Then, p is deleted iff condition H is true, where

$$\begin{aligned} \mathrm{H}: \ & (\text{edge } p) \wedge (\sim \text{edge } x_1 \vee \overline{x}_3 \vee \overline{x}_7) \\ & \wedge (\sim \textit{edge } x_7 \vee \overline{x}_5 \vee \overline{x}_1) \\ & \wedge (\sim \text{edge } x_1 \vee \sim \text{edge } x_8 \vee \sim \text{edge } x_7) \end{aligned}$$

With this condition, edge information on neighboring pixels is used to prevent the disappearance of vertical lines that are two pixels wide. This algorithm is implemented [55] on SIMD and MIMD machines with a modification to reduce redundancy in the edge computation of elements with common neighbors. Condition H is shown [137] to be almost equivalent to R1-R4 of [103], where the crossing number instead of edge information of neighboring pixels is considered. The only differences are that the earlier work stipulates $b(p) \geq 2$, and the conditions of [54] would add the condition "or $b(x_3) \notin [2, 6]$" to R3 (and "or $b(x_1) \notin [2, 6]$" to R4). In turn, [137] also proposes (without results) to modify the conditions R1-R4 by changing R1 to $2 \leq b(p) \leq 6$ and the second parts of R3 and R4 to $y_3 = 1$ and $y_1 = 1$, respectively, where y_1 is the east neighbor of x_1, and y_3 is the north neighbor of x_3. The first modification is also suggested by [50] to preserve diagonal lines in [54].

Besides all the modifications mentioned above, [1] and [114] also suggest procedures to reduce, to unit thickness, the skeletons obtained by [136]. Therefore, the algorithm originally proposed by Rutovitz and modified by Deutsch has been implemented with various levels of changes in a number of articles, and the relationships between them are shown in Fig. 4.

Despite the somewhat baffling array of modifications proposed to the algorithm originating from Rutovitz, there are many similarities among the results. Basically, the algorithms of [103], [136], [54], [39], and [137] have the common deficiency of possible excessive erosion in the thinning of diagonal lines. The problem may depend on whether the lines are at 45 or 135° to the horizontal, and lines that are an even number of pixels in width seem to be more vulnerable to erosion. The simple modification of [70] and [50] solves this problem by retaining extraneous pixels and creating skeletons of more than unit width. The skeleton is then thinned (incompletely) by a second pass in [128]. Fig. 5 shows a typical example of thinning a problematic pattern using these algorithms.

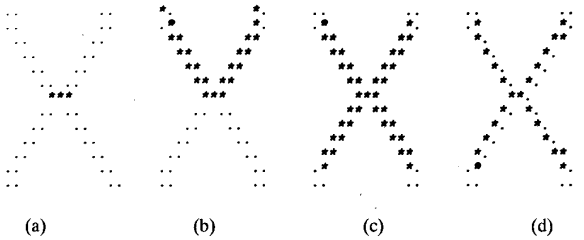


Fig. 5. Results of thinning by different algorithms based on Rutovitz: (a) See [54], [103], [136], and [138]; (b) see [39]; (c) see [50] and [70]; (d) see [128].

Necessary and sufficient conditions for preserving topology while deleting border points in parallel are given in [100]. If only 3×3 local neighborhoods are considered, then an operation that deletes (for example) north border pixels would preserve topology iff the only north border points it removes are simple and have at least two 1's as 8-neighbors. It is proved [124] that this operation would not change the genus number of P. This algorithm was implemented in [101], where 4-connected skeletons are observed to contain more noisy branches.

Equivalent conditions for the simplicity of a border point are given in [6], where it is shown that a north border point with at least two 1's as 8-neighbors would be simple iff $x_5\overline{x}_7x_1 = 0$ and $\sum_{i=1}^{4} \overline{x}_{2k-1}x_{2k}\overline{x}_{2k+1} = 0$. These conditions are combined into one in [102].

Another 4-subiteration algorithm is that of [17], where skeletal pixels found in each iteration are assigned the iteration number for the subsequent calculation of object width. The values of skeletal pixels from previous iterations are left unchanged, whereas those of interior pixels are incremented by one in each iteration. In the nth iteration, pixel p is tested for deletion as a north border element if $x_3 = 0$, p has value n, and $x_7 \neq 0$. Such a border element p is considered skeletal if there exists $x_i \in N(p)$ such that $x_i > 0$ and $N(x_i) \cap N(p) = \{p, x_i\}$; otherwise, p is assigned the value 0 and deleted. This algorithm preserves topology but was found to be sensitive to boundary noise and the order of the subiterations.

Parallel algorithms using both four and two subiterations have also been implemented by means of matching against windows analogous to those of Fig. 2. Stefanelli and Rosenfeld [115] implemented two such algorithms and proved that they preserve topology. In each subcycle, the final (skeletal) pixels are stored. In order to avoid deletion of contour points that are also final points, all the contour points are first deleted, after which the final points are added. When four subcycles are used and north border points are considered for deletion, the final point conditions are those shown in Fig. 6 together with the 90° rotations of (a) and (b). In this and subsequent figures, pixels that are left blank may be 1 or 0 or are "don't care" pixels.

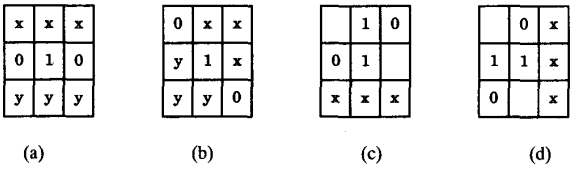


Fig. 6. Final point conditions of [115].

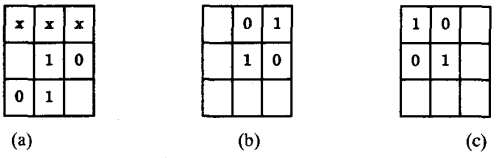


Fig. 7. Additional final point conditions [115].

Fig. 8. Pixel deletion conditions in [4].

The difference between window (b) here and that in Fig. 2(b) implies that the present algorithm would produce imperfectly 8-connected skeletons, as is indeed the case. The two-subcycle algorithm combines deletion of south and east contour points in the first subcycle and the rest in the second. For the second subcycle, the final point conditions are the ones given above in Fig. 6 together with those shown in Fig. 7. Conditions (b) and (c) are added to ensure connectivity and preserve diagonal lines; however, thin horizontal or vertical lines are not preserved. The processing speed of this algorithm was found to be comparable with that of [103], but the use of only 3×3 windows here allows for easier implementation on a cellular network.

Another algorithm using thinning windows is [4], where the masks for pixel deletion are those shown in Fig. 8 together with their 90° rotations. Each mask is applied in parallel to P, and the masks are applied in the order (a), (b), followed by their 90° rotations, and so on. Therefore, this algorithm removes pixels from eight borders in the order nw, w, etc. The asymmetric nature of mask (b) results from the requirement that P should not vanish completely in the parallel process. This algorithm was proved to operate correctly; it was especially suitable for implementation on parallel processors that have the ability to extract "1" and "0" elements having a predetermined number of "1" or "0" elements in chosen neighborhood positions, and it was implemented on a Clip 4 parallel processor [53]. Since not all deletable pixels were removed, other masks were added to produce those of Fig. 9 and their rotations.

In addition, it was found [53] that superior results are obtained when removal is restricted to pixels that do not only satisfy the criteria so far but would have satisfied them at the start of the current iteration. This ensures that just

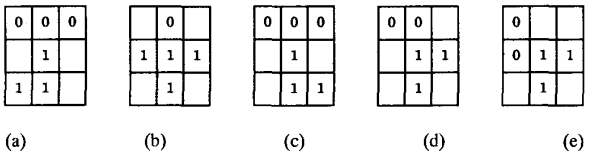


Fig. 9. Pixel deletion condition in [53].

one layer of pixels is removed all around P and results in a more predictable algorithm with fewer anomalies at the corners where two layers of pixels may be removed otherwise. The author calls this a border parallel operation, whereas the original is border sequential. Implementation of a border parallel algorithm on the Clip 4 processor does not involve extra computation since the 1's and 0's are detected in separate steps in any case; therefore, it is necessary only to test the 1's in the pattern at the beginning of the iteration, whereas the 0's are tested in the pattern obtained so far. The longer computation time required is the result of deleting fewer pixels in each cycle. In the same paper, it is also shown that the border parallel conditions of Fig. 9 are equivalent to the four-subiteration border parallel deletion of contour pixels with crossing number $X_H(p) = 1$. Fig. 10 shows the results of using this crossing number to thin a pattern by the border parallel and border sequential algorithms. Fig. 10(a) shows the results after one iteration when the contour pixels are deleted according to the sequence north, east, south, and west. Typically, the border-sequential algorithm deletes more pixels at the corners; as a result, the end pixel condition may be encountered sooner with the preservation of these pixels leading to more short (and possibly noisy) branches. For this reason, the right end of the horizontal stroke is retained by this algorithm in the final skeleton shown in Fig. 10(b), thus illustrating the tradeoff that is sometimes involved when different thinning algorithms are used.

By using look-up tables, a simulation of the border parallel conditions of Fig. 9 has been implemented [112], [113], in which mask (c) and its rotations are omitted and endpoint conditions added. To prevent excessive erosion of two-pixel-thick lines, additional information is used so that the current pixel is retained if its preceding neighbor pixel has been removed.

Two recent papers [118], [49] have implemented parallel thinning using the crossing number $X_H(p)$ to examine pixels for deletion in two subiterations. In [118], the (approximately) 4×4 window used is shown in Fig. 11. Pixels removed in the first subcycle are a) deletable north contour pixels, and b) west contour pixels that are deletable after the pixels in a) have been removed. It is this latter requirement that produces more consistent results while making it necessary to increase the size of the neighborhood and the complexity of the algorithm. In this subcycle, p is deleted iff

S1: $X_H(p) < 2$,
S2: $(b(p) > 2) \vee [(b(p) = 2) \wedge \sum_{i=1}^{8} x_k x_{k+1} = 0]$, and
S3: $(x_3 = 0) \vee [(x_5 = 0) \wedge S4 \wedge S5]$, where
S4: $x_2 \vee \bar{x}_8(y_1 \vee y_2 \vee y_3) \vee \bar{y}_2 y_3 = 1$, and
S5: $\bar{x}_4 \vee y_5 \vee \bar{x}_2 y_4 = 1$.

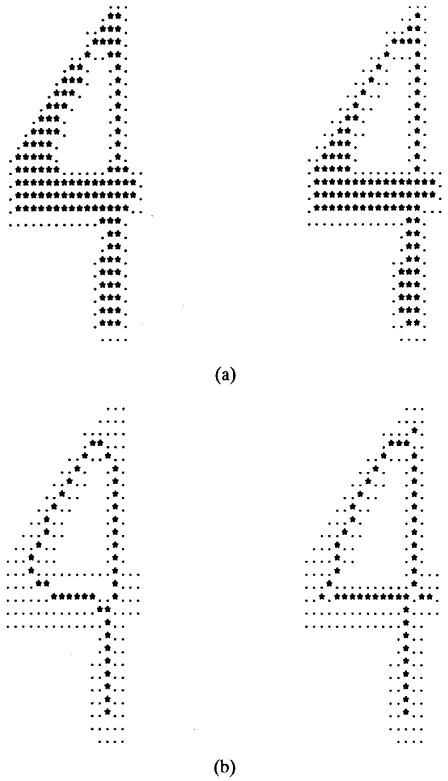


Fig. 10. Thinning by border parallel and border sequential algorithms of [53]: (a) After one cycle; (b) final skeleton. The figures to the left in both (a) and (b) show the border parallel (four cycles), and the figures to the right show the border sequential (three cycles).

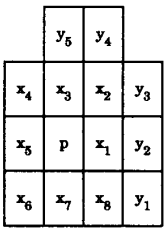


Fig. 11. Neighborhood of [118].

For the second subcycle, the conditions are S1, S2, and the 180° rotations of S3-S5.

In [49], two such algorithms are implemented. For the first one, the image is divided into two distinct subfields in a checkerboard pattern, and each subiteration deletes pixels p in a subfield iff p is a contour pixel, $b(p) > 1$, and $X_H(p) = 1$. This procedure results in noise spurs and zigzagging vertical or horizontal lines. The other algorithm is a modification of [136] to thin to 8-connected skeletons and retain diagonal lines and 2×2 squares. Under this scheme, p is deleted iff

G1: $X_H(p) = 1$

G2: $2 \leq \min \{n_1(p), n_2(p)\} \leq 3$, where $n_1(p) = \sum_{i=1}^{4} x_{2k-1} \vee x_{2k}$ and $n_2(p) = \sum_{i=1}^{4} x_{2k} \vee x_{2k+1}$ represent the number of 4-adjacent pairs of pixels in $N(p)$ containing one or two black pixels, and

G3: $(x_2 \vee x_3 \vee \bar{x}_8) \wedge x_1 = 0$ in the first subiteration and its 180° rotation in the second.

Labeling schemes have also been suggested for use in parallel thinning. In [41], one-subcycle thinning is accomplished by recoding the pattern pixels to incorporate connectivity information from a 5×5 window into the coded pixels in $N(p)$. Two initial scans are used to recode the pixels into core, interior, rim, and skeleton points (c, i, r, and s, respectively). The basic rule is to replace r pixels by b (background) pixels if a horizontal or vertical irb sequence is present, with exceptions made to preserve connectivity and end points. This is similar to the "ideal" method for thinning proposed in [40], where p is D perfect if it belongs to a horizontal or vertical configuration ipb, where i is an interior point, and b is a background pixel. (Actually, D-perfect points satisfy one of the two conditions for regular points in [11] and [12]). An I-perfect point is defined analogously in terms of a diagonal alignment with the added condition that the two pixels 4-adjacent to both p and b must both be white. This paper suggests that parallel deletion of pixels that are simple and perfect (D or I perfect) would produce well-defined pseudo skeletons that are invariant to translation and rotation. Retention of endpoints is obviously achieved because they are never perfect.

Since it is impossible to determine whether a neighbor of a border point is an interior point if only 3×3 windows are used, additional information is incorporated [125] from outside the window by defining nonend points. If SC is the set of simple contour points, and $B = P - SC$, then $p \in P$ is a nonend point iff

T1: p is adjacent to a point in B,

T2: every neighbor of p in P is in B or adjacent to a point in $B \cap N(p)$,

T3: the points in $B \cap N(p)$ are connected in $N(p)$, and

T4: $b(p) \geq 2$.

The operation that removes simple, nonend, contour points in parallel is shown to preserve topology.

In order to extend the limitations imposed by 3×3 local operations, hybrid schemes have also been proposed to combine distance transforms with thinning. The use of distance transforms provides more global information. It has also been argued that since thinning operations can preserve connectivity while distance transforms possess the reconstruction capability, combining the two operations would be logical if both properties are desired [125]. The suggested procedure is that for each iteration, the set I of interior points of the existing patterns P should be determined, and its expansion $E(I)$ is defined as $E(I) = \{q | q \in I$ or has a neighbor in $I\}$. Then, the set of local maxima (in convexity) can be obtained as $P - E(I)$ [9], and contour points can be deleted unless they are local maxima, nonsimple, or nonend points (defined above). The remaining contour points are added to the skeletal set, and the process is repeated with I as the existing pattern. It is also argued [63] that such a hybrid scheme may be most efficient since a distance transform can be used initially to remove the bulk of the exterior

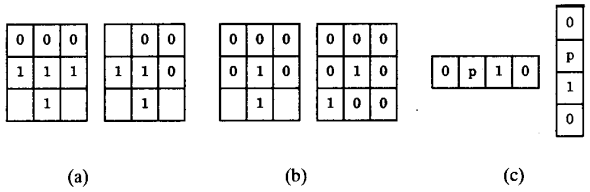


Fig. 12. Templates of [25]: (a) Thinning; (b) trimming; (c) restoring.

pixels in a fixed number of passes, and then, a peeling algorithm can be used to thin the remaining image to unit width.

The one-subiteration algorithms of [103], [54], [41], and [125] have already been discussed, where all utilize information from a larger than 3×3 neighborhood to maintain connectedness. Other algorithms of this group are [25] and [22]. In [25], eight thinning, two restoring, and eight trimming templates are applied in parallel in one pass in which pixels matching the 3×3 thinning or trimming, but not those of the 1×4 or 4×1 restoring templates, are removed. These templates are those of Fig. 12 and 90° rotations of (a) and (b).

The thinning templates delete border and corner points and the restoring templates retain 2-pixel-wide lines, whereas the trimming templates reduce noise spurs at the obvious expense of shortening branches. This algorithm is also implemented [20] using only 4×4 thinning and restoring templates (by filling in with don't care pixels). Only two iterations are used in different directions, after which postprocessing was applied to obtain a medial line that is not usually one-pixel wide.

In [22], a pseudo one-subcycle algorithm is implemented that essentially uses information from a 5×5 window. Despite a rather involved formulation employing many new terms that will not be included here, a pixel p is basically a candidate for deletion iff

C1: $X_H(p) = 1$ (equivalent to $LC(p) = T_b - 1(\neq 0)$), and
C2: (a) the black pixels in $N(p)$ are 4-connected ($E(p) = LC(p)$), or
(b) p belongs to a two-pixel-wide diagonal line.

The conditions for C2(b) had been given in [33] and modified in [21], and it is the latter scheme that is used here and stated as $S_4(p) = 2$ and $L(p) = 1$. If p satisfies the conditions for deletion, then its 4-neighbors are checked for possible deletion. If these are not deletable or connectivity would not be changed, then p is removed. The information is stored in look-up tables (as in [21]) for the checking process. In addition, a specially designed thinning condition can be added [23] to the algorithm to better preserve L-shaped patterns by removing more pixels from a concave corner than before. The condition is that p can be deleted if its four corner neighbors contain one white pixel while the other three have all black 8-neighbors. In [24], comparisons are made among [54], [25], and [22] in terms of thinning conditions and a connectivity-preserving function; The method in [22] is considered to be superior because it has more complete thinning conditions and a smaller ratio of cases guarded for connectivity preservation (hence, fewer iterations would be required).

For parallel algorithms, much of the attention in recent years has been focused on processing speed, and this has led to many comparisons either in terms of computing time or the number of iterations or subiterations used [118], [20], [49], [24]. However, some comparisons of the number of iterations may not be entirely valid [50]. A one-pass algorithm ([54], for example) is usually achieved by means of a larger support than 3×3, and this creates the need for computation of intermediate results or edge information on neighboring pixels. For this reason, a common framework is needed to define an "iteration" so that parallel algorithms can be meaningfully compared. The proposed scheme [50] assumes that in one iteration, each element of a mesh computer can compute any logical function of a 3×3 neighborhood, and the parallel speed of an algorithm is measured by the number of such iterations needed. Using this criterion of measurement, the parallel speed of [54] is not an improvement over that of two-subiteration algorithms such as [136].

In conclusion, it would appear that, as opposed to sequential algorithms, much of the complexity of parallel algorithms results from the need to preserve connectedness while using parallel operations in a small local neighborhood. This is a problem particular to the nature of parallel thinning, and it has been addressed by using subiterations (and serializing the procedure to some extent) or by enlarging the neighborhood to be examined. In either case, intermediate results are actually computed and used in some form in order to preserve more global structures.

V. Noniterative Thinning Methods

In the preceding sections, we discussed algorithms that produce a skeleton by examination and deletion of contour pixels. In this section, the algorithms to be considered are nonpixel based; they produce a certain median or center line of the pattern directly in one pass without examining all the individual pixels. Since the algorithms that accomplish this by medial axis or distance transforms are discussed in a separate paper, we are mainly concerned here with methods that determine center lines by line following or from run length encoding. It has been argued that this is the way human beings would perform thinning and that it is possible to retain global features and maintain connectedness in the process [15].

The simplest category of these algorithms determines the midpoints of black intervals in scan lines and connects them to form a skeleton. These methods have the advantage of being computationally efficient, but they also have natural disadvantages and would create noisy branches when the strokes are nearly parallel to the scan line [87]. They are valid in certain specialized applications where the scanning direction can be approximately perpendicular to that of the stroke. For example, in [75], the objects are rectangular with only small undulations along two parallel and much longer sides; therefore, a useful median line can be obtained by maintaining a constant scanning direction perpendicular to

these sides. In other applications, the scanning directions are variable by rotations of 90 [74] or 45° [81]. The scanning in [60] is in both the x- and y-directions and the direction of smaller length (within certain constraints) would be selected.

Some algorithms obtain approximations of skeletons by connecting strokes having certain orientations. For example, four pairs of window operations are used in four subcycles [76] to test for and determine the presence of vertical, horizontal, right, or left diagonal limbs in the pattern. At the same time, the operators also locate turning points and end points by a set of final point conditions, and these extracted points are connected to form a line segment approximation of the skeleton. In [108] and [110], the boundary pixels are first labeled according to the above four local orientations. For each boundary pixel, a search is made for the same kind of label on the opposite side of the boundary (within a maximum stroke width) in the direction perpendicular to that given by the label. The midpoints of these pairs are then connected to form a skeleton. These methods are not appropriate for general applications since they are not robust, especially for patterns with highly variable stroke directions and thicknesses.

In [89], the vector form of a skeleton is obtained from the run length coding of P. Consecutive horizontal black intervals are grouped if they have approximately the same width and are roughly collinear. Based on a width-versus-height criterion, each such group is represented by either a horizontal vector or a vector joining the midpoints of the first and last intervals. Since this criterion alone is not sufficient, certain complex rules for a "compound vectorization" are devised to consider the relationship between groups to produce the final result. Obviously, the skeletons cannot be expected to preserve the geometric properties of the pattern, but this vector representation is convenient for use in pattern recognition. A recent thinning method [119] also produces a graph-like representation of a pattern by dividing the pattern into small units called meshes, on which partial recognition can be performed, and then merging the meshes to form partial graphs.

Another set of algorithms determine the center line of P by tracking the two contours of each curve simultaneously. In [31], the edge trackers move under the constraint of maintaining minimum distance between them. In [105], elongated ribbon-like simply connected objects without protrusions are tracked by approximate trapezoids using two pointers to follow the two contours of P. The midpoint of the base of the next trapezoid is considered to be skeletal, where the next base is either the side opposite the present base or a diagonal, depending on whether the diagonals are almost equal in length. This algorithm is extended [106] to objects containing protrusions by generating skeletons of the protrusions and backbone separately and then joining them together.

The above algorithm considers a simply connected pattern P to be a many-sided polygon, and this approach is also adopted in [72] to construct a median line. For every vertex v of P, its opposite line segment L is determined as the side of P closest to v. Then, a "projecting" line L_v is drawn from v to L, where L_v is the following:

1) The bisector of the interior angle θ at v if $\theta \leq 180°$
2) the two normal lines of the vectors forming v, otherwise.

If L_v intersects L within a threshold distance, its midpoint is added to the median line. We note that the lines of 1) are called pseudonormals in [19], where a discrete version of the symmetric axis of an object is theoretically derived using them.

Lines of elongated objects are followed by rectangular windows of variable size in [15], where the window can shrink or grow in size according to the width of the line at the location of the window. The skeleton is the unit width line connecting the centers of successive windows. In [82], a combination of thinning and stroke tracking is used on Chinese characters. Contour pixels are first sequentially examined, and p is deleted if

O1: $N(p)$ has at least one interior pixel,
O2: $N(p)$ has, at most, three contour pixels, and
O3: the contour pixels in $N(p) \cup \{p\}$ are traced consecutively.

When no more pixels can be deleted, the remaining pattern is traced using a 2×2 square, and the skeleton is the loci of these squares. The tracing is repeated if necessary.

Apart from these contour tracking methods, skeletal pixels have also been determined by a heuristic approach [59]. The deletion of a pixel p is determined by the local pattern density $d(p)$ and the density $d^*(p)$ of $d(p)$, where $d(p) = \sum_{i=1}^{8} x_i$ and $d^*(p) = \sum_{i=1}^{8} d(x_i)$. p would be retained if $d(p) \leq t_1$ and $d^*(p) \leq t_2$, where t_1 and t_2 are arithmetically consistent thresholds that can be set in a learning process.

Similar density functions based only on the 4-neighbors are considered in [138], where the values of an addition matrix are determined iteratively on black pixels p by

$$A_{(1)}(p) = \sum_{i=1}^{4} x_{2i-1}, \text{and}$$

$$A_{(n)}(p) = \sum_{i=1}^{4} A_{(n-1)}(x_{2i-1}) \text{ if } A_{(n-1)}(p) = 4(n-1),$$

when $n > 1$.

The number of directions (out of four) in which the pixel has a maximum value (according to the final addition matrix) is its comparative degree, and the matrix of these degrees is thinned according to the sequential procedure of [137], together with the added condition of having a small degree. The addition matrix also contains information that allows for an approximate reconstruction of the pattern.

Using a different approach, skeletons can also be obtained from Fourier descriptors, at least for patterns that are not closed or overlapping and that have constant width [90]. For such patterns, the contour is traced to obtain a closed curve from which Fourier descriptors can be extracted. Fourier descriptors of the skeleton can then be determined, and the skeleton can be constructed from a finite set of harmonics. However, it would appear that highly mathematical methods can be used to obtain skeletons of only rather idealized patterns.

VI. Conclusions

In the literature, there appears to be a general agreement about the requirements that should be met by a thinning algorithm. These include preservation of topological and geometric properties, isotropy, reconstructibility, and high processing speed. Whereas the nonpixel-based thinning methods of Section V are efficient in terms of the number of operations required, it would seem impossible for them to preserve more detailed features of patterns since they are usually based on locating certain critical points and connecting them. These procedures are useful in applications where the detection of such points would suffice, one possible example being feature extraction in OCR. In general, however, the emphasis should be placed on the development of parallel algorithms for processing speed, especially when parallel image processing structures become increasingly available.

Reconstructibility, or the ability to regenerate the original pattern from the skeleton, is one objective measure of the accuracy with which a skeleton is representing the pattern. This criterion is generally satisfied by algorithms based on medial axis and other distance transforms by virtue of the fact that their skeletal pixels are also the centers of maximal blocks with known radii, but this is usually not the case with thinning algorithms. (Some exceptions are [86], [88], [138] and [77] to a certain extent, all of which use labeling schemes to retain the distances of skeletal pixels from the background).

Complete isotropy or invariance under rotation seems almost impossible to achieve in iterative algorithms. In sequential algorithms, the result depends on the order in which pixels are examined, and in parallel algorithms that remove one or two types of border points in each subiteration, the resulting skeleton depends on the order of the subiterations. Examples of nonisotropic results from these algorithms are shown in [8]. At the same time, medial axis transforms are not invariant under rotation due to a lack of algorithms based on true Euclidean distance maps so that very different results can be obtained from right-angled corners with one side parallel to an axis and at 45° to it. The use of these transforms simply transfers the problem from the thinning algorithm to the distance function, and the result can be just as idiosyncratic [53].

Maintaining connectedness and topology in thinning appears to have been resolved through various means. In sequential algorithms, it is sufficient to examine 3×3 local neighborhoods from the viewpoint of crossing number, for example. Parallel algorithms resolve the problem by dividing each cycle into subiterations or by considering a larger neighborhood in one subiteration.

Preservation of geometric properties, however, appears to be a more difficult problem. The main difficulty is that to achieve simplicity of algorithm and/or processing hardware, it is desirable to consider only small local neighborhoods, but these neighborhoods are incapable of providing global, structural information of the kind that is needed (for example) to distinguish between noise spurs and genuine end points. To prevent excessive erosion and creation of spurious end points at the same time, various attempts have been made to eliminate the end-point condition [18], make the condition more broadly applicable [70], [50], or apply the condition only at the later stages of thinning [118]. However, every modification of this type involves a tradeoff when uniformly applied, and additional information is really needed to make finer distinctions between cases. For this reason, various criteria using distance transforms have been introduced ([101], [8], and [27], for example). Some such means would be needed to propagate more global information to contour pixels, and it may allow for faster deletion of pixels in the initial stages.

Ultimately, the particular geometric properties a skeleton should preserve may be problem or application dependent. Algorithms based on medial axis transforms that possess the reconstruction capability would be well suited to applications such as data compression for storage and facsimile transmission [29]. However, the skeletons obtained by these methods are very sensitive to contour noise since branches can originate to many convexities on the boundary, depending on the distance transform used. Of course, this property is what renders them capable of exactly recreating the original pattern, but they would not be useful for pattern recognition where patterns with a wide variety of insignificant local contour differences can belong to the same class. For this latter application, it is much more important for the pattern to be represented by a collection of arcs lying along the center lines of the main curves of the pattern, whereas small perturbations of the contour should be ignored. At the same time, it is recognized that the thinning of blob-like patterns may result in skeletons that cannot preserve the original shape. In this respect, two types of skeletons are actually obtained from the different methods of thinning and medial axis transforms, and the choice should depend on the application.

In conclusion, it should be mentioned that comparison of the quality of skeletons remains a largely subjective, visual decision since the concepts of (connected) medial line, noisy branch, and excessive erosion have not been precisely defined. In addition, it is not objectively or quantitatively clear as to how a skeletal branch should accurately reflect the shapes of the two contours it represents. Comparisons are inevitably made when a new (or a modification of an existing) algorithm is proposed, but it is often based on the inconclusive evidence of one or two patterns. More general comparisons have been made from a digital geometry point of view [120], and in [92] and [93], comparisons have been made between skeletons obtained from thinning algorithms and reference skeletons prepared by human subjects. However, a more objective framework for the measurement of skeleton quality remains to be developed.

Acknowledgment

The authors are grateful to Associate Editor C. Dyer and the three reviewers; their detailed comments have been very helpful in the revision of this article.

References

[1] W. H. Abdulla, A. O. M. Saleh, and A. H. Morad, "A preprocessing algorithm for handwritten character recognition," *Pattern Recogn. Lett.*, vol. 7, no. 1, pp. 13–18, 1988.

[2] T. M. Alcorn and C. W. Hoggar, "Pre-processing of data for character recognition," *Marconi Rev.*, vol. 32, pp. 61–81, 1969.
[3] C. J. Ammann and A. G. Sartori-Angus, "Fast thinning algorithm for binary images," *Image Vision Comput.*, vol. 3, no. 2, pp. 71–79, 1985.
[4] C. Arcelli, L. Cordella, and S. Levialdi, "Parallel thinning of binary pictures," *Electron. Lett.*, vol. 11, no. 7, pp. 148–149, 1975.
[5] C. Arcelli and G. Sanniti di Baja, "On the sequential approach to medial line transformation," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-8, no. 2, pp. 139–144, 1978.
[6] C. Arcelli, "A condition for digital points removal," *Signal Processing*, vol. 1, no. 4, pp. 283–285, 1979.
[7] C. Arcelli and G. Sanniti di Baja, "Medial lines and figure analysis," in *Proc. 5th Int. Conf. Pattern Recogn.*, 1980, pp. 1016–1018.
[8] ——, "A thinning algorithm based on prominence detection," *Pattern Recogn.*, vol. 13, no. 3, pp. 225–235, 1981.
[9] C. Arcelli, L. P. Cordella, and S. Levialdi, "From local maxima to connected skeletons," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-3, no. 2, pp. 134–143, 1981.
[10] C. Arcelli, "Pattern thinning by contour tracing," *Comput. Graphics Image Processing*, vol. 17, pp. 130–144, 1981.
[11] C. Arcelli and G. Sanniti di Baja, "Finding multiple pixels," in *Image Analysis and Processing* (V. Cantoni, S. Levialdi, and G. Musso, Eds.). New York: Plenum, 1986, pp. 137–144.
[12] ——, "On the simplicity of digital curves and contours," in *Proc. 8th Int. Conf. Patt. Recogn.* (Paris, France), 1986, pp. 283–285.
[13] ——, "A contour characterization for multiply connected figures," *Patt. Recogn. Lett.*, vol. 6, no. 4, pp. 245–249, 1987.
[14] ——, "A one-pass two-operation process to detect the skeletal pixels on the 4-distance transform," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 11, no. 4, pp. 411–414, 1989.
[15] O. Baruch, "Line thinning by line following," *Patt. Recogn. Lett.*, vol. 8, no. 4, pp. 271–276, 1988.
[16] H. Beffert and R. Shinghal, "Skeletonizing binary patterns on the homogeneous multiprocessor," *Patt. Recogn. Artificial Intell.*, vol. 3, no. 2, pp. 207–216, 1989.
[17] A. Bel-Lan and L. Montoto, "A thinning transform for digital images," *Signal Processing*, vol. 3, no. 1, pp. 37–47, 1981.
[18] M. Beun, "A flexible method for automatic reading of handwritten numerals," *Philips Tech. Rev.*, vol. 33, no. 5, pp. 89–101; 130–137, 1973.
[19] F. L. Bookstein, "The line-skeleton," *Comput. Graphics Image Processing*, vol. 11, pp. 123–137, 1979.
[20] N. G. Bourbakis, "A parallel-symmetric thinning algorithm," *Patt. Recogn.*, vol. 22, no. 4, pp. 387–396, 1989.
[21] Y. -S. Chen and W. -H. Hsu, "A modified fast parallel algorithm for thinning digital patterns," *Pattern Recogn. Lett.*, vol. 7, no. 2, pp. 99–106, 1988.
[22] ——, "A systematic approach for designing 2-subcycle and pseudo 1-subcycle parallel thinning algorithms," *Patt. Recogn.*, vol. 22, no. 3, pp. 267–282, 1989.
[23] ——, "A 1-subcycle parallel thinning algorithm for producing perfect 8-curves and obtaining isotropic skeleton of an L-shape pattern," in *Proc. Int. Conf. Comput. Vision Patt. Recogn.* (San Diego, CA), 1989, pp. 208–215.
[24] ——, "A comparison of some one-pass parallel thinnings," *Patt. Recogn. Lett.*, vol. 11, no. 1, pp. 35–41, 1990.
[25] R. T. Chin, H. -K. Wan, D. L. Stover, and R. D. Iverson, "A one-pass thinning algorithm and its parallel implementation," *Comput. Vision Graphics Image Processing.*, vol. 40, pp. 30–40, 1987.
[26] Y. K. Chu and C. Y. Suen, "An alternative smoothing and stripping algorithm for thinning digital binary patterns," *Signal Processing*, vol. 11, no. 3, pp. 207–222, 1986.
[27] N. Chuei, T. Y. Zhang, and C. Y. Suen, "New algorithms for thinning binary images and Chinese characters," *Comput. Processing Chinese Oriental Languages*, vol. 2, no. 3, pp. 169–179, 1986.
[28] C. H. Cox, P. Coueignoux, B. Blesser, and M. Eden, "Skeletons: A link between theoretical and physical letter descriptions," *Pattern Recogn.*, vol. 15, no. 1, pp. 11–22, 1982.
[29] E. R. Davies and A. P. N. Plummer, "A new method for the compression of binary picture data," in *Proc. 5th Int. Conf. Patt. Recogn.*, 1980, pp. 1150–1152.
[30] ——, "Thinning algorithms: A critique and a new methodology," *Pattern Recogn.*, vol. 14, no. 1, pp. 53–63, 1981.
[31] J. -D. Dessimoz, "Specialized edge-trackers for contour extraction and line-thinning," *Signal Processing*, vol. 2, no. 1, pp. 71–73, 1980.
[32] E. S. Deutsch, "Preprocessing for character recognition," in *Proc. IEEE NPL Conf. Patt. Recogn.* (Teddington), 1968, pp. 179–190.
[33] ——, "Comments on a line thinning scheme," *Comput. J.*, vol. 12, 1969, p. 412.
[34] ——, "Toward isotropic image reduction," in *Proc. IFIP Congress* (Ljubljana, Yugoslavia), 1971, pp. 161–172.
[35] ——, "Thinning algorithms on rectangular, hexagonal, and triangular arrays," *Comm. ACM*, vol. 15, no. 9, pp. 827–837, 1972.
[36] A. R. Dill, M. D. Levine, and P. B. Noble, "Multiple resolution skeletons," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-9, no. 4, pp. 495–504, 1987.
[37] G. P. Dinnen, "Programming pattern recognition," in *Proc. West. Joint Comput. Conf.* (New York), 1955, pp. 94–100.
[38] R. O. Duda, P. E. Hart, and J. H. Munson, "Graphical-data-processing research study and experimental investigation," AD650926, pp. 28–30, Mar. 1967.
[39] U. Eckhardt, "A note on Rutovitz' method for parallel thinning," *Patt. Recogn. Lett.*, vol. 8, no. 1, pp. 35–38, 1988.
[40] U. Eckhardt and G. Maderlechner, "Thinning algorithms for document processing systems," in *Proc. IAPR Workshop Comput. Vision: Spec. Hardware Ind. Applications* (Tokyo, Japan), 1988, pp. 169–172.
[41] A. Favre and H. Keller, "Parallel syntactic thinning by recoding of binary pictures," *Comput. Vision Graphics Image Processing*, vol. 23, pp. 99–112, 1983.
[42] G. Feigin and N. Ben-Yosef, "Line thinning algorithm," *Proc. SPIE*, vol. 397, pp. 108–112, 1984.
[43] G. E. Forsen, "Processing visual data with an automaton eye," in *Pictorial Pattern Recognition* (G. C. Cheng, R. S. Ledley, D. K. Pollock, and A. Rosenfeld, Eds.). Washington DC: Thompson, 1968, pp. 471–502.
[44] J. G. Fraser, "Further comments on a line thinning scheme," *Comput. J.*, vol. 13, pp. 221–222, 1970.
[45] G. Gallus and P. W. Neurath, "Improved computer chromosome analysis incorporating preprocessing and boundary analysis," *Phys. Med. Biol.*, vol. 15, no. 3, pp. 435–445, 1970.
[46] R. C. Gonzalez and P. Wintz, "The skeleton of a region," in *Digital Image Processing*. Reading, MA: Addison-Wesley, 1987, pp. 398–402.
[47] V. K. Govindan and A. P. Shivaprasad, "A pattern adaptive thinning algorithm," *Pattern Recogn.*, vol. 20, no. 6, pp. 623–637, 1987.
[48] F. C. A. Groen and N. J. Foster, "A fast algorithm for cellular logic operations on sequential machines," *Pattern Recogn. Lett.*, vol. 2, no. 5, pp. 333–338, 1984.
[49] Z. Guo and R. W. Hall, "Parallel thinning with two-subiteration algorithms," *Comm. ACM*, vol. 32, no. 3, pp. 359–373, 1989.
[50] R. W. Hall, "Fast parallel thinning algorithms: Parallel speed and connectivity preservation," *Comm. ACM*, vol. 32, no. 1, pp. 124–131, 1989.
[51] C. J. Hilditch, "An application of graph theory in pattern recognition," in *Machine Intell.* (B. Meltzer and D. Michie, Eds.). New York: Amer. Elsevier, 1968, pp. 325–347, vol. 3.
[52] ——, "Linear skeletons from square cupboards," in *Machine Intell.* (B. Meltzer and D. Michie, Eds.). New York: Amer. Elsevier,1969, pp. 403–420, vol. 4.
[53] ——, "Comparison of thinning algorithms on a parallelprocessor," *Image Vision Comput.*, vol. 1, no. 3, pp. 115–132, 1983.
[54] C. M. Holt, A. Stewart, M. Clint, and R. H. Perrott, "An improved parallel thinning algorithm," *Comm. ACM*, vol. 30, no. 2, pp. 156–160, 1987.
[55] C. Holt and A. Stewart, "A parallel thinning algorithm with fine grain subtasking," *Parallel Comput.*, vol. 10, pp. 329–334, 1989.
[56] S. H. Y. Hung and T. Kasvand, "Critical points on a perfectly 8- or 6-connected thin binary line," *Pattern Recogn.*, vol. 16, no. 3, pp. 297–306, 1983.
[57] M. I. Izutsdkiver, "Algorithm for the initial processing of an ensemble of symbols in the recognition process," *Auto. Remote Contr.*, vol. 35, no. 8, pp. 1292–1298, 1974.
[58] N. Izzo and W. Coles, "Blood-cell scanner identifies rare cells," *Electron.*, vol. 35, pp. 52–55, Apr. 1962.
[59] R. N. Jones and M. C. Fairhurst, "Skeletonisation of binary patterns: A heuristic approach," *Electron. Lett.*, vol. 14, no. 9, pp. 265–266, 1978.
[60] K. Kedem and D. Keret, "A fast algorithm for skeletonizing lines by midline technique," in *Proc. Int. Comput. Sci. Conf.*, (Hong Kong), 1988, pp. 731–735.
[61] R. A. Kirsch, L. Cahn, C. Ray, and G. J. Urban, "Experiments inprocessing pictorial information with a digital computer," in *Proc. East. Joint Comput. Conf.* (New York), 1957, pp. 221–229.
[62] T. Y. Kong and A. Rosenfeld, "Digital topology: Introduction and survey," *Comput. Vision Graphics Image Processing*, vol. 48, pp. 357–393, 1989.
[63] J. T. Kuehn, J. A. Fessler, and H. J. Siegel, "Parallel image thinning and vectorization on PASM," in *Proc. Int. Conf. Comput. Vision Patt. Recogn.*, 1985, pp. 368–374.

[64] P. C. K. Kwok, "A thinning algorithm by contour generation," *Comm. ACM*, vol. 31, no. 11, pp. 1314–1324, 1988.

[65] ——, "Customising thinning algorithms," in *Proc. IEEE Int. Conf. Image Processing Applications*, 1989, pp. 633–637.

[66] ——, "Thinning in a distributed environment," in *Proc. 10th Int. Conf. Patt. Recogn.* (Atlantic City, NJ), 1990, pp. 694–699.

[67] L. Lam and C. Y. Suen, "Structural classification and relaxation matching of totally unconstrained handwritten Zip-code numbers," *Patt. Recogn.*, vol. 21, no. 1, pp. 19–31, 1988.

[68] C. Lantuejoul, "Skeletonization in quantitative metallography," in *Issues in Digital Image Processing* (R. M. Haralick and J. C. Simon, Eds.). Amsterdam: Sijthoff and Noordoff, 1980, pp. 107–135.

[69] Y. Le Cun *et al.*, "Handwritten digit recognition: Applications of neural network chips and automatic learning," *IEEE Commun. Mag.*, pp. 41–46, Nov. 1989.

[70] P. S. P. Wang, "An improved fast parallel thinning algorithm for digital patterns," in *Proc. Int. Conf. Comput. Vision Patt. Recogn.*(San Francisco), 1985, pp. 364–367.

[71] ——, "A comment on a fast parallel algorithm for thinning digital patterns," *Comm. ACM*, vol. 29, no. 3, pp. 239–242, 1986.

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

[73] B. H. McCormick, "The Illinois pattern recognition computer—Illiac III," *IEEE Trans. Electron. Comput.*, vol. EC-12, no. 6, pp. 791–813, 1963.

[74] B. Moayer and K. S. Fu, "A syntactic approach to fingerprint pattern recognition," *Pattern Recogn.*, vol. 7, pp. 1–23, 1975.

[75] J. L. Mundy and R. E. Joynson, "Automatic visual inspection using syntactic analysis," in *Proc. Int. Conf. Patt. Recogn. Image Processing*, 1977, pp. 144–147.

[76] I. S. N. Murthy and K. J. Udupa, "A search algorithm for skeletonization of thick patterns," *Comput. Graphics Image Processing*, vol. 3, pp. 247–259, 1974.

[77] N. J. Naccache and R. Shinghal, "STPA: A proposed algorithm for thinning binary patterns," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-14, no. 3, pp. 409–418, 1984.

[78] ——, "An investigation into the skeletonization approach of Hilditch," *Pattern Recogn.*, vol. 17, no. 3, pp. 279–284, 1984.

[79] ——, "In response to 'A comment on an investigation into the skeletonization approach to Hilditch,'" *Patt. Recogn.*, vol. 19, no. 2, p. 111, 1986.

[80] A. Nakayama, F. Kimura, Y. Yoshida, and T. Fukumura, "An efficient thinning algorithm for large scale images based upon pipeline structure," in *Proc. 7th Int. Conf. Patt. Recogn.* (Montreal), 1984, pp. 1184–1187.

[81] T. V. Nguyen and J. Sklansky, "A fast skeleton-finder for coronary arteries," in *Proc. 8th Int. Conf. Patt. Recogn.* (Paris, France), 1986, pp. 481–483.

[82] H. Ogawa and K. Taniguchi, "Thinning and stroke segmentation for handwritten Chinese character recognition," *Patt Recogn.*, vol. 15, no. 4, pp. 299–308, 1982.

[83] J. F. O'Callaghan and J. Loveday, "Quantitative measurement of soil cracking patterns," *Patt. Recogn.*, vol. 5, pp. 83–98, 1973.

[84] L. O'Gorman, "$k \times k$ thinning," *Comput. Vision Graphics Image Processing*, vol. 51, pp. 195–215, 1990.

[85] T. Pavlidis, "A thinning algorithm for discrete binary images," *Comput. Graphics Image Processing*, vol. 13, pp. 142–157, 1980.

[86] ——, "A flexible parallel thinning algorithm," in *Proc. Int. Conf. Patt. Recog. Image Processing* (Dallas, TX), 1981, pp. 162–167.

[87] ——, *Algorithms for Graphics and Image Processing*. Rockville, MD: Comput. Sci., 1982, pp. 195–214.

[88] ——, "An asynchronous thinning algorithm," *Comput. Graphics Image Processing*, vol. 20, pp. 133–157, 1982.

[89] ——, "A vectorizer and feature extractor for document recognition," *Comput. Vision Graphics Image Processing*, vol. 35, pp. 111–127, 1986.

[90] E. Persoon and K. S. Fu, "Shape discrimination using Fourier descriptors," *IEEE Trans. Syst., Man Cybern.*, vol. SMC-7, no. 3, pp. 170–179, 1977.

[91] J. Piper, "Efficient implementation of skeletonisation using interval coding," *Patt. Recogn. Lett.*, vol. 3, no. 6, pp. 389–397, 1985.

[92] R. Plamondon and C. Y. Suen, "On the definition of reference skeletons for comparing thinning algorithms," in *Proc. Vision Interface 1988* (Edmonton, Canada), 1988, pp. 70–75.

[93] ——, "Thinning of digitized characters from subjective experiments: A proposal for a systematic evaluation protocol of algorithms," in *Computer Vision and Shape Recognition* (A. Krzyzak, T. Kasvand, and C. Y. Suen, Eds.). Singapore: World Scientific, 1989, pp. 261–272.

[94] K. Preston, "The CELLSCAN system—A leucocyte pattern analyzer," in *Proc. West. Joint Comput. Conf.* (Los Angeles, CA), 1961, pp. 173–183.

[95] K. Preston, M. J. B. Duff, S. Levialdi, P. E. Norgren, and J. -I. Toriwaki, "Basics of cellular logic with some applications in medical image processing," *Proc. IEEE*, vol. 67, no. 5, pp. 826–857, 1979.

[96] M. C. Rahier and P. G. A. Jespers, "Dedicated LSI for a microprocessor controlled hand carried OCR system," *IEEE J. Solid-State Circuits*, vol. SC-15, no. 1, pp. 14–24, 1980.

[97] S. Riazanoff, B. Cervelle, and J. Chorowicz, "Parametrisable skeletonization of binary and multi-level images," *Patt. Recogn. Lett.*, vol. 11, no. 1, pp. 25–33, 1990.

[98] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *J. ACM*, vol. 13, no. 4, pp. 471–494, 1966.

[99] A. Rosenfeld, "Connectivity in digital pictures," *J. ACM*, vol. 17, no. 1, pp. 146–160, 1970.

[100] ——, "A characterization of parallel thinning algorithms," *Inform. Contr.*, vol. 29, no. 3, pp. 286–291, 1975.

[101] A. Rosenfeld and L. S. Davis, "A note on thinning," *IEEE Trans. Syst. Man Cybern.*, vol. 25, pp. 226–228, 1976.

[102] A. Rosenfeld and A. C. Kak, *Digital Picture Processing (2nd ed.)*. New York: Academic, 1982, vol. II, ch. 11.

[103] D. Rutovitz, "Pattern recognition," *J. Roy. Stat. Soc.*, vol. 129, Series A, pp. 504–530, 1966.

[104] P. Saraga and D. J. Woollons, "The design of operators for pattern processing," in *Proc. IEEE NPL Conf. Patt. Recogn.* (Teddington), 1968, pp. 106–116.

[105] B. Shapiro, J. Pisa, and J. Sklansky, "Skeletons from sequential boundary data," in *Proc. Int. Conf. Patt. Recog. Image Processing* (Chicago, IL), 1979, pp. 265–270.

[106] ——, "Skeleton generation from x, y boundary sequences," *Comput. Graphics Image Processing*, vol. 15, pp. 136–153, 1981.

[107] H. Sherman, "A quasitopological method for the recognition of line patterns." in *Proc. Int. Conf. on Inform. Processing* (Paris, France), 1959, pp. 232–238.

[108] R. M. K. Sinha, "Primitive recognition and skeletonization via labeling," in *Proc. Int. Conf. Syst. Man Cybern.* (Halifax, Canada), 1984, pp. 272–279.

[109] R. M. K. Sinha and C. J. Ammann, "Comments on fast thinning algorithm for binary images," *Image Vision Comput.*, vol. 4, no. 1, pp. 57–58, 1986.

[110] R. M. K. Sinha, "A width-independent algorithm for character skeleton estimation," *Comput. Vision Graphics Image Processing*, vol. 40, pp. 388–397, 1987.

[111] R. W. Smith, "Computer processing of line images: A survey," *Patt. Recogn.*, vol. 20, no. 1, pp. 7–15, 1987.

[112] M. Del Sordo and T. Kasvand, "A near-neighbor processor for line thinning," in *Proc. Int. Conf. Acoust. Speech Signal Processing*, 1985, pp. 1523–1525.

[113] ——, "Neighborhood look-up tables for skeletonization," in *Proc. 4th Scand. Conf. Image Anal.* (Trondheim, Norway), 1985, pp. 663–670.

[114] J. H. Sossa, "An improved parallel algorithm for thinning digital patterns," *Patt. Recogn. Lett.*, vol. 10, no. 2, pp. 77–80, 1989.

[115] R. Stefanelli and A. Rosenfeld, "Some parallel thinning algorithms for digital pictures," *J. ACM*, vol. 18, no. 2, pp. 255–264, 1971.

[116] R. Stefanelli, "A comment on an investigation into the skeletonization approach of Hilditch," *Patt. Recogn.*, vol. 19, no. 1, pp. 13–14, 1986.

[117] C. Y. Suen, M. Berthold, and S. Mori, "Automatic recognition of handprinted characters," *Proc. IEEE*, vol. 68, no. 4, pp. 469–487, 1980.

[118] S. Suzuki and K. Abe, "Binary picture thinning by an iterative parallel two-subcycle operation," *Patt. Recogn.*, vol. 10, no. 3, pp. 297–307, 1987.

[119] T. Suzuki and S. Mori, "A thinning method based on cell structure," in *Proc. Int. Workshop Frontiers Handwriting Recogn.* (Montreal, Canada), 1990, pp. 39–52.

[120] H. Tamura, "A comparison of line thinning algorithms from a digital geometry viewpoint," in *Proc. 4th Int. Conf. Patt. Recogn.* (Kyoto, Japan), 1978, pp. 715–719.

[121] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 12, no. 8, pp. 787–808, 1990.

[122] J. -I. Toriwaki and S. Yokoi, "Distance transformation and skeletons of digitized pictures with applications," in *Progress in Pattern Recognition* (L. N. Kanal and A. Rosenfeld, Eds.). New York: North-Holland, 1981, pp. 189–264.

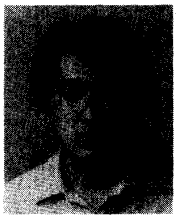
[123] E. E. Triendl, "Skeletonization of noisy hand-drawn symbols using parallel operations," *Patt. Recogn.*, vol. 2, pp. 215–226, 1970.

[124] Y. F. Tsao and K. S. Fu, "Parallel thinning operations for digital binary images," in *Proc. Int. Conf. Patt. Recog. Image Processing* (Dallas, TX), 1981, pp. 150–155.

[125] ——, "A general scheme for constructing skeleton models," *Inform. Sci.*, vol. 27, no. 1, pp. 53–87, 1982.
[126] L. J. Vliet and B. J. H. Verwer, "A contour processing method for fast neighborhood operations," *Patt. Recogn. Lett.*, vol. 7, no. 1, pp. 27–36, 1988.
[127] A. M. Vossepoel, J. P. Buys, and G. Koelewijn, "Skeletons from chain-coded contours," in *Proc. 10th Int. Conf. Patt. Recogn.* (Atlantic City), 1990, pp. 70–73.
[128] P. S. P. Wang, L.-W. Hui, and T. Fleming, "Further improved fast parallel thinning algorithm for digital patterns," in *Computer Vision, Image Processing and Communications—Systems and Applications* (P. S. P. Wang, Ed.). Singapore: World Scientific, 1986, pp. 37–40.
[129] P. S. P. Wang and Y. Y. Zhang, "A fast serial and parallel thinning algorithm," in *Proc. Eighth Euro. Meeting Cybern. Syst. Res.* (Vienna, Austria), 1986, pp. 909–915.
[130] , "A fast and flexible thinning algorithm," *IEEE Trans. Comput.*, vol. 38, no. 5, pp. 741–745, 1989.
[131] Y. Xia, "A new thinning algorithm for binary images," in *Proc. 8th Int. Conf. Patt. Recogn.* (Paris, France), 1986, pp. 995–997.
[132] 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, pp. 847–851, 1987.
[133] Q. -Z. Ye and P. E. Danielsson, "Inspection of printed circuit boards by connectivity preserving shrinking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 5, pp. 737–742, 1988.
[134] S. Yokoi, J. -I. Toriwaki, and T. Fukumura, "Topological properties in digitized binary pictures," *Syst. Comput. Contr.*, vol. 4, no. 6, pp. 32–39, 1973.
[135] ——, "An analysis of topological properties of digitized binary pictures using local features," *Comput. Graphics Image Processing*, vol. 4, pp. 63–73, 1975.
[136] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Comm. ACM*, vol. 27, no. 3, pp. 236–239, 1984.
[137] Y. Y. Zhang and P. S. P. Wang, "A modified parallel thinning algorithm," in *Proc. 9th Int. Conf. Patt. Recogn.* (Rome, Italy), 1988, pp. 1023–1025.
[138] ——, "A maximum algorithm for thinning digital patterns," in *Proc. 9th Int. Conf. Patt. Recogn.* (Rome, Italy), 1988, pp. 942–944.

Louisa Lam received the B. A. degree from Wellesley College, Wellesley, MA, where she was elected to Phi Beta Kappa. She received the Ph.D. degree in mathematics from the University of Toronto, Toronto, Canada.

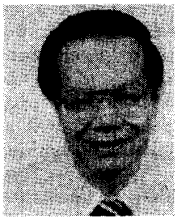
She is currently teaching mathematics at Vanier College and conducting research at Concordia University, Montreal, Canada. Her research interests include character recognition and skeletonization algorithms.



Seong-Whan Lee (M'91) was born in Beolgyo, Korea, in 1962. He received the B. S. degree in computer science and statistics from Seoul National University, Seoul, Korea, in 1984 and the M. S. and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology in 1986 and 1989, respectively.

In 1987, he worked as a visiting researcher at the Pattern Recognition Division, Delft University of Technology, Delft, the Netherlands. He was a visiting scientist at the Centre for Pattern Recognition and Machine Intelligence, Concordia University, Montreal, Canada, during the winter of 1989 and the summer of 1990. Since 1989, he has been an Assistant Professor in the Department of Computer Science, Chungbuk National University, Chungbuk, Korea. His research interests include pattern recognition, computer graphics, and intelligent man-machine interfaces.

Dr. Lee was awarded a best paper prize from the Korea Information Science Society in 1986. He is a member of the governing board of the Special Interest Group on Artificial Intelligence of Korea. He is also a member of the Korea Information Science Society, the Pattern Recognition Society, the Association for Computing Machinery, and the IEEE Computer Society.



Ching Y. Suen (F'86) received the M.Sc. (Eng.) degree from the University of Hong Kong and the Ph.D. degree from the University of British Columbia, Vancouver, Canada.

In 1972, he joined the Department of Computer Science at Concordia University, Montreal, Canada, where he became Professor in 1979 and served as Chairman from 1980 to 1984. Presently, he is the Director of the new Center for Pattern Recognition and Machine Intelligence (CENPARMI) at Concordia University. During the past 15 years, he has been appointed to visiting positions at several institutions in different countries. He is the author/editor of several books including *Computer Vision and Shape Recognition, Frontiers in Handwriting Recognition*, and *Computational Analysis of Mandarin and Chinese.* His latest book is entitled *Operational Expert System Applications in Canada*, which is published by Pergamon Press. He is the author of many papers, and his current interests include pattern recognition and machine intelligence, expert systems, optical character recognition and document processing, and computational linguistics.

An active member of several professional societies, Dr. Suen is an Associate Editor of several journals related to his areas of interest. He is the Past President of the Canadian Image Processing and Pattern Recognition Society, Governor of the International Association for Pattern Recognition, and President of the Chinese Language Computer Society.