Int. J. Appl. Math. Comput. Sci., 2010, Vol. 20, No. 2, 317–335
DOI: 10.2478/v10006-010-0024-4

K3M: A UNIVERSAL ALGORITHM FOR IMAGE SKELETONIZATION AND A REVIEW OF THINNING TECHNIQUES

KHALID SAEED [*], MAREK TABĘDZKI [**], MARIUSZ RYBNIK [***], MARCIN ADAMSKI [**]

[*] Faculty of Physics and Applied Computer Science
AGH University of Science and Technology, Al. Mickiewicza 30, 30–059 Cracow, Poland
e-mail: saeed@novell.ftj.agh.edu.pl

[**] Faculty of Computer Science
Białystok Technical University, ul. Wiejska 45 A, 15–351 Białystok, Poland
e-mail: {m.tabedzki,m.adamski}@pb.edu.pl

[***] Faculty of Mathematics and Informatics
University of Białystok, ul. M. Skłodowskiej-Curie 14, 15–097 Białystok, Poland
e-mail: mariuszrybnik@wp.pl

This paper aims at three aspects closely related to each other: first, it presents the state of the art in the area of thinning methodologies, by giving descriptions of general ideas of the most significant algorithms with a comparison between them. Secondly, it proposes a new thinning algorithm that presents interesting properties in terms of processing quality and algorithm clarity, enriched with examples. Thirdly, the work considers parallelization issues for intrinsically sequential algorithms of thinning. The main advantage of the suggested algorithm is its universality, which makes it useful and versatile for a variety of applications.

Keywords: skeletonization, thinning, digital image processing, parallelization, iteration, thinning methodologies, sequential thinning, parallel thinning.

1. Introduction

As a digital image processing technique, thinning has been around for many years. Thinning (also termed skeletonization) is an image pre-processing technique that is important in a number of applications like pattern recognition, data compression and data storage (Gonzalez and Woods, 2007). The history of thinning covers over fifty years. In the 1950s Dinnen (1955) found that an averaging operation over a square window with a high threshold resulted in a thinning of the input image. Then Kirsch *et al.* (1958) mentioned that the “custer” operation was an early attempt to obtain a thinline representation of certain character patterns. However, still there is no ultimate solution suitable for each individual area of application (Jaisimha *et al.*, 1994; Jang and Chin, 1992; Lam *et al.*, 1992; Lee *et al.*, 1991). This is due to the complicated nature of the processing as well as various evaluation measures that should be considered.

There are more than one thousand algorithms that have been published on this topic. The thinning algorithms mentioned in this paper have been selected from among the best known published works. These selected works are given here for comparison to show and evaluate the efforts of other researchers and authors in working out and implementing a number of algorithms and approaches for a variety of applications.

The first trials conducted in this field by the authors of this paper date back to 1999 (Saeed and Niedzielski, 1999). This was to aid recognition algorithms of cursive-character letters, which required thinning before classification. This work is an attempt to propose a new criterion and employ it to various applications. The presented method leads to a fast recognition algorithm of both typewritten and handwritten texts with the possibility of applying it to thin images of a variety of objects. The basic idea of thinning according to this approach lies in the generation of a skeleton with a one-pixel width. However, to serve more complicated cases, some modifications

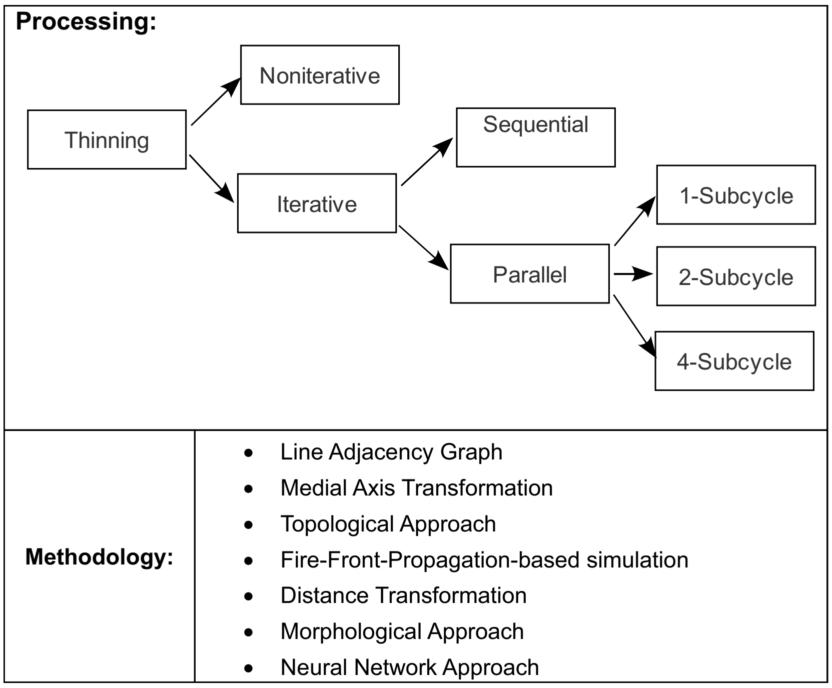


Fig. 1. Thinning algorithms taxonomy using two criteria: processing and methodology.

have been made to the whole criterion. The method suggested by the authors is universal and simple. It consists of several fast and easy-to-implement deleting phases. The obtained experimental results have also shown a successful application of the method in image pre-processing as well as image preparation for feature extraction and classification. The method proved its universality—not only does it work for printed words, but it has also shown a remarkable success rate on thinning handwritten scripts, signatures, numbers, characters of non-Latin-based alphabets (Arabic, Japanese) and even biometric images such as faces or fingerprints. The paper is organized as follows. Section 2 presents the state of the art in the following order: the proposed taxonomy of thinning algorithms, a description of general ideas of the most significant ones and a short comparison in terms of efficiency. Section 3 presents the thinning algorithm proposed by the authors in detail. Section 4 presents the results of the K3M algorithm and the comparison with other algorithms. Section 5 describes the concepts for the parallelization of intrinsically sequential thinning algorithms. Finally, Section 6 concludes the work and shows perspectives for future development.

2. State of the art

The survey presented in this section introduces a historical review covering over 40 years of works on thinning algorithms, together with their basic principles and the state of the art.

This survey is mainly based on algorithms mostly known to the authors, available either as papers in international conference proceedings, periodical and journal transactions or book chapters. The central role in the survey was played by Pavlidis (1982a), Lam *et al.* (1992), Malina *et al.* (2002) and, finally, Klette and Rosenfeld (2004). This is in addition to lots of individual works showing different methods and algorithms of thinning.

Generally, each of the presented algorithms has its own advantages and disadvantages, and each has its applications where it performs better than others. Therefore, it is often difficult to directly compare the results.

2.1. Taxonomy. All thinning algorithms can be classified as either iterative or non-iterative. In iterative methods, thinning algorithms produce a skeleton by examining and deleting contour pixels through an iterative process in either sequential or parallel way.

Parallel algorithms may also be further classified according to their performance, i.e., in 4-, 2-, or 1-subcycle manners. The latter (1-subcycle parallel algorithms) have always received more considerable attention in the research area of parallel thinning as they have reduced the computation time in a number of iterations, and that is why they are sometimes called one-pass or fully parallel algorithms (Chen and Hsu, 1989b; Chen, 1996; Guo and Hall, 1992).

In sequential thinning algorithms, contour points are examined for deletion in a predetermined order, and this can be accomplished by either raster scanning or following the image by contour pixels. In parallel thinning algorithms, pixels are examined for deletion on the basis of results obtained only from the previous iteration. That is why parallel thinning algorithms are suitable for implementation in parallel processors.

Non-iterative (non-pixel based) thinning algorithms produce a certain median or centre line of the pattern to be thinned directly in one pass, without examining all the individual pixels.

The methods used in the works presented in this section may also be categorized as one of the following methodologies (Fig. 1).

2.2. General ideas and examples. In this section a detailed description of most of the basic algorithms of thinning is presented in order to show how others attack the problem of skeletonization. Some of them are explained in detail in (Saeed, 2004).

Rutovitz (1966) proposed a parallel algorithm that formed a basis for many thinning algorithms. Image pixels are examined for deletion in an iterative process (not divided into subcycles). Contour pixels with at least two black neighbours, and where all its neighbours are 4-connected (sticking together with its edge), are marked for deletion. This may result in excessive erosion, it does not reduce diagonal lines to a unit pixel width and the skeleton is not located centrally (due to the asymmetric nature of its conditions).

Blum (1967) presented *skeletonization* as transformation for the aim of shape description. He called this transformation MAT—*Medial Axis Transformation*. Figure 2 shows examples of obtaining skeletons from some objects by this method.

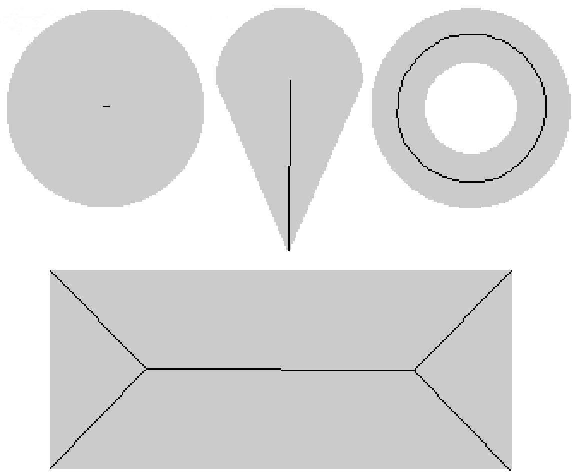


Fig. 2. Illustration of the result of thinning shapes by Blum's algorithm.

Hilditch (1968; 1969) worked out a sequential approach, in which he defined the 'crossing number'—the number of times one crosses over from a white point to a black point when the eight neighbours of a tested pixel are traversed in sequence. The image is scanned from left to right and from top to bottom, and pixels are marked for deletion under additional conditions to prevent erosion, maintain connectivity and preserve two-pixel wide lines. This method was also used with gray-scale images.

Rosenfeld (1975) showed many new aspects and solved serious problems commonly met in thinning algorithms. He was the first to evaluate the necessary and sufficient conditions for preserving topology while deleting border points in parallel process. Thereby, he solved the problem of erosion in the thinning of diagonal lines. In his algorithms, he considered a 3×3 local neighbourhood. He proved that '*a border pixel is removed only if its neighbourhood has only one black component (i.e., the Hilditch crossing number is 1) and has at least two black neighbours.*'

Arcelli's parallel behaviour algorithm (Arcelli and di Baja, 1978) uses two 3×3 thinning windows together with their $90°$ rotations as masks for pixel deletion. It removes pixels from eight borders in the following order: north-west, west, and so forth. However, it does not remove all deletable pixels.

Dyer and Rosenfeld (1979) introduced the algorithm for thinning gray-scale pictures. It is based on a generalized concept of pixel connectivity: two pixels are "connected" if there is a path joining them with no pixel lighter than either of them. Using this idea, the thinned version on an image can be obtained by changing each pixel gray level to the minimum of its neighbor's gray levels.

Pavlidis (1980; 1981; 1982a; 1982b) introduced the definition of "multiple pixels" for the first time—points that are traversed more than once during contour tracing, points with no neighbours in the interior and points on two-pixel-wide lines. If only the "multiple pixels" from every tracing are retained, the result may not be a connected skeleton. Therefore, "multiple pixels" are called skeletal, as well as eight neighbours of skeletal pixels from a previous iteration. This does not result in a one-pixel-wide skeleton. Contour pixels are traced sequentially.

Then, Pavlidis (1982a) proposed a combination of parallel and sequential operations—a pattern could be divided into fields, which are processed independently by parallel processors, each operating sequentially on its own segment.

Arcelli's algorithm (Arcelli, 1981) is of a sequential type. It uses contour tracing to find the pixels for deletion. This approach allows reducing the computational time needed to obtain the skeleton. Contour analysis is performed to find regions to be represented by the skeleton branches.

The method of Arcelli and di Baja (1981) is also a sequential algorithm—successive iterations consider the removal of contour elements. Regions that can be regarded as significant protrusions are detected before applying the removal operations. In (Arcelli and di Baja, 1987), the authors used the Pavlidis concept of 'multiple pixels' (pixels are placed where the contour self-interacts). The paper gives a necessary definition to satisfactorily detect such pixels. Then Arcelli and di Baja (1989) developed a sequential algorithm that used a 4-distance transform to find a set of skeletal pixels (two-pixel wide, at most) within one raster scan of the image. Then, another inspection of the picture removes the unnecessary pixels.

The algorithm of Favre and Keller (1983) covers one-subcycle parallel thinning using syntactic rules and a cascade of tables to determine a new pixel code, on the basis of the distance to the background. Two-scan post-processing is applied to obtain the final skeleton.

Ammann and Sartori-Angus (1985) proposed a fast thinning algorithm where the image is compressed to a reduced gray-scale representation. This is then thresholded (with connectivity constraints) to a binary image to which an existing thinning algorithm (Arcelli and di Baja, 1981) was applied. The resulting skeleton is then expanded to its original scale.

Chin *et al.* (1987) wrote a paper presenting a one-subcycle thinning algorithm and its parallel implementation. They called it a *'one-pass algorithm'*. They paid attention to the idea of bias skeletons, the basic approach used later by Chen (Chen and Hsu, 1989b; Chen, 1996) in his modified algorithm. Bias skeletons appear in the junction of lines that form angles less than 90° (Fig. 3(a)). They mentioned two reasons behind generating bias skeletons. One is that the restoring templates are of an even size and hence the pixel removal process is not symmetrical, unless we are dealing with systematic design, as restoring templates are only used in maintaining the connectivity of the final skeleton. The other factor is due to the side effect of the thinning templates, that is, they remove pixels from convex corners faster than from concave corners. A sample of thinning results using this algorithm is given in Fig. 3(a).

Baruch (1988) presented a noniterative, non-pixel based algorithm, which produces a skeleton in one pass, by line-following. The line is followed by a window of variable size, the skeleton is the line connecting centres of successive windows. The method is less sensitive to noise than conventional thinning algorithms.

The algorithm of Guo and Hall (1989; 1992) is another example of parallel thinning. The contour pixels are examined for deletion in an iterative process. The decision is based on a 3×3 neighbourhood. Each iteration is divided into two subcycles—one deleting the north and east pixels and the other for south and west. This algorithm does not give a one-pixel-wide skeleton. Although it shows very good results in thinning and provides a rather good skeleton, the algorithm and its 1-subcycle modified version (Guo and Hall, 1992) do not preserve all significant geometric features of the image (see Fig. 3(b)).

Chen and Hsu worked out an interesting algorithm modifying it through a series of papers (1989b; 1989a; 1990; 1988). Chen (1996) reached a 1-subcycle parallel thinning algorithm producing a one-pixel-wide skeleton that preserves significant geometric features of patterns (Fig. 3(c)). He developed a method embedded in a parallel algorithm (Chen and Hsu, 1989b) to produce bias-reduced skeletons using, as a major factor, what he defined the HDP—*Hidden Deletable Pixel* detected by algorithms of vector analysis. The HDP is detected and removed, so that the bias skeleton can be reduced. Despite the satisfactory results of thinning, Chen's algorithm did not show any significant application to object-images, handwritten words or at least machine-written alphabets of a cursive character. The results of thinning curved parts of machine-written English letters, for example, are not as good as those of straight lines. This is demonstrated by the thinning results of this algorithm in Fig. 3(c).

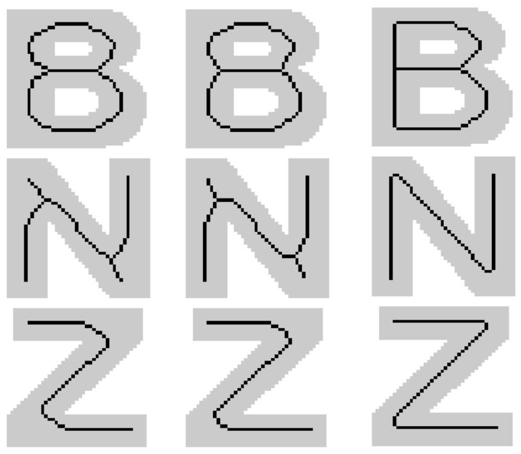


Fig. 3. Thinning by CWSI (Chin *et al.*, 1987) (a), AFP3 (Guo and Hall, 1992) (b), CYS (Chen, 1996) (c).

Parker *et al.* (1994) introduced the *force-based* approach with a new idea for thinning strategy based on a definition of a 'skeletal pixel' as being as far from the object outline as possible while maintaining basic connectivity properties. Accordingly, the basic idea is that a skeleton is a global property of a binary object, and that the boundary should be used to locate the skeleton pixels. The criterion is summarized as follows: the background pixels which are adjacent to the boundary act as if they exerted a force; the skeletal pixels lie in areas having the ridges of this force field and are located by searching where directions of the force vectors change significantly. Figure 4 shows how to compute the force at a pixel location and the vector sum of forces.

Zhang and Wang (1996) created the *2-Subiteration Parallel-Thinning Algorithm with Templates—PTA2T.*

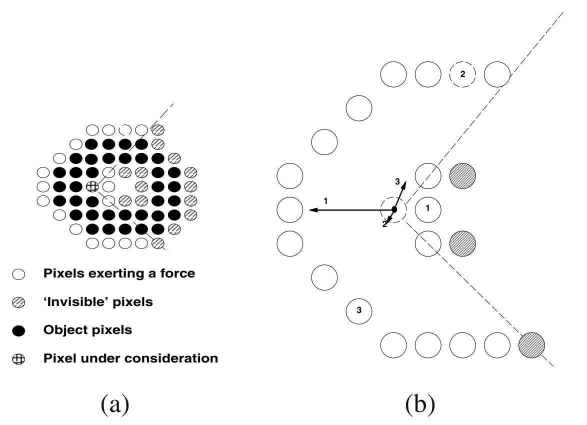


Fig. 4. Force-based thinning: computing the force acting at a given pixel (a), vector sum of forces (b).

This algorithm actually represents a modification to Guo and Hall's approach. In an iterative process the deletion decision is based on eleven 3×3 thinning templates. There are two subcycles—the second uses templates rotated by 180 degrees. The algorithm preserves the connectivity of patterns and produces one-pixel-wide skeletons (Fig. 5). In the paper, Zhang claimed this algorithm was faster than Guo and Hall's.

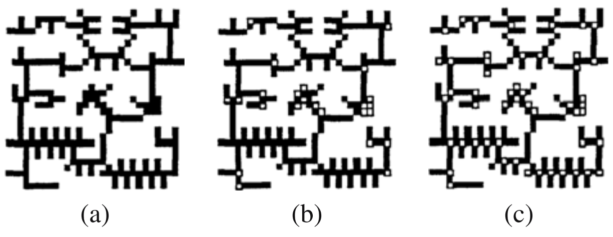


Fig. 5. Thinning results of the image in (a) using Guo and Hall (b) and Zhang (c) methods. Empty rectangles indicate the places of the removed pixels

The Altuwaijri algorithm—*ART2 NN* (Altuwaijri and Bayoumi, 1998) is a data-image-clustering based thinning algorithm that employs neural networks and adaptive resonance theory for image clustering. The image is an Arabic character (Altuwaijri and Bayoumi, 1998). Altuwaijri claims the algorithm produces skeletons which are superior to the outputs of the conventional algorithms. He compared it (Fig. 6) with Zhang's algorithm given in (Zhang and Wang, 1996). ART2 showed higher data-reduction efficiency and much simpler skeletons with less noise spurs and reduced time complexity.

The main advantage of this method, however, is its ability to remove the redundant loops appearing in the final thinned shape in many thinning algorithms. This problem may not seem particularly important for the preservation of essential geometric features in most alphabetical scripts. However, in Arabic and many other cursive

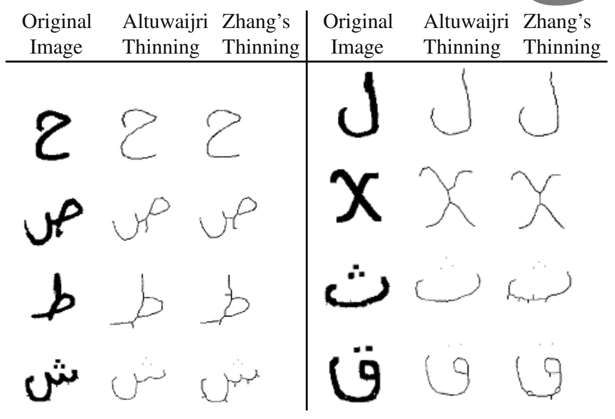


Fig. 6. Altuwaijri's thinning of images with added noise compared with thinning by Zhang's algorithm.

character alphabets such a problem is serious as the resulting letter may resemble another letter, rather than the correct one. Figure 7 shows how the redundant loop may change the letter *Seen* into the letter *Ssad*.

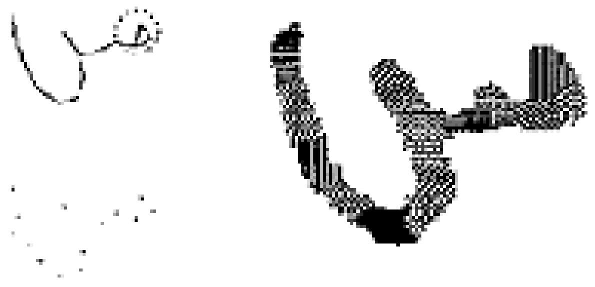


Fig. 7. Redundant loops may transform the script *Seen* into another one—*Ssad*.

Andreadis *et al.* (2000) proposed a method for obtaining a skeleton from a color image using morphological operations on vectors defined in the HSV color space. In the presented method the extraction of a skeleton is preceded by the ordering of vectors based on their individual components (hue, saturation and value).

KMM (Saeed and Niedzielski, 1999; Saeed, 2001; Saeed *et al.*, 2001) is the authors' original contribution based on an original approach described in (Saeed and Niedzielski, 1999) and modified in a series of works (Saeed, 2001; Saeed *et al.*, 2001). The algorithms acronym comes from the initials of the authors: Khalid, Marek and Mariusz. Basically, the algorithm is a sequential iterative one. It produces a one-pixel-wide continuous skeleton from an image. The details of the thinning algorithm are given in (Saeed *et al.*, 2001). The main goal of the algorithm was

1. to obtain a continuous *line of symmetry* (the name given by Blum to the skeleton (Blum, 1967)), and

2. to thin all kinds of images to their skeleton—digits,

alphabet letters, signatures, pictures and medical images in their two-dimensional views.

The ideal skeleton was not the aim of the criterion. In fact, there is no algorithm that yields an ideal skeleton in any case. The most important thing is usually the retaining of pixels that preserve the significant geometric features of the given pattern in all kinds of images. Figure 8 shows KMM thinning results when applied to the same figures given in (Malina *et al.*, 2002) and shown earlier in Fig. 2. Notice that the two circles differ from each other by one pixel-length in their diameters. The circle of a diameter with an even number of pixels is thinned to a dot, while that of a diameter with an odd number of pixels is thinned to an angular shape. Note also that a rectangle is thinned to a line and seems to be more appropriate than Blum's skeleton (Blum, 1967).

KMM was under continuous modification. Its latest version for universal use will be given in Section 3 as K3M—**K**halid, **M**arek, **M**ariusz, **M**arcin.

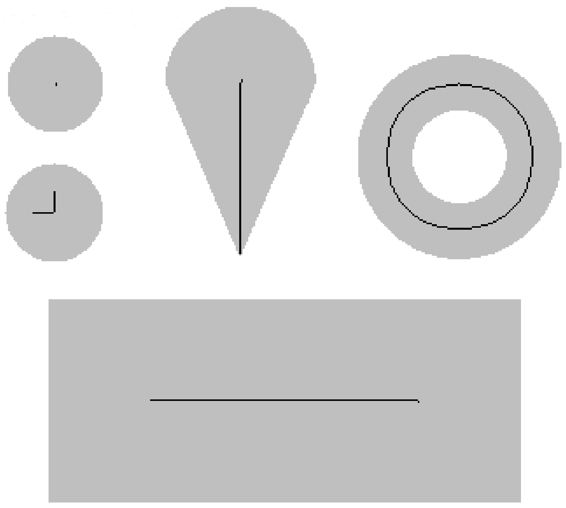


Fig. 8. Figures presented in (Malina *et al.*, 2002) thinned with KMM (Saeed, 2004) (compare with Fig. 2).

The algorithm presented by Ahmed and Ward (2002) is an example of parallel thinning with the decision of deletion made upon an algorithm based on quantity and position of neighbours of each pixel. With 20 a rules, a pixel is classified into one of four classes, and then a decision is made. Two-pixel-wide lines are treated separately.

Huang *et al.* (2003) proposed another parallel thinning algorithm. Pixel elimination rules are based on 3×3 windows considering all kinds of relations (256) formed by 8-neighbours of the object pixel. All the rules are applied simultaneously to each pixel. In order to keep the connectivity of even pixel-width lines, additional checks are applied before deleting the pixels. These additional checks involve using rules based on 4×4, 3×4 and 4×3 windows. In order to reduce the loss of information when thinning shapes like in Fig. 9(a), the authors introduce a parameter R,

$$R = \frac{A(s_1)}{A(s_2)}, \tag{1}$$

where $A(s)$ is the area of s pixels, s_1 is the image skeleton, s_2 is the image contour.

Based on its value, the algorithm returns either a thinned image (Fig. 9(b)) or its contour (Fig. 9(c)). If R is less than the threshold, the image contour is retained. The threshold value used by the authors was 0.4.

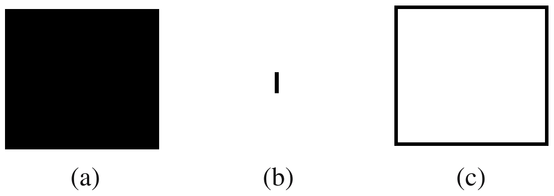


Fig. 9. Image (a), its thinned shape (b) and its contour (c).

Ji and Feng (2004) created a method that interprets the image as a 2D thermal conductor that consists of pixels, where pixel intensity represents the temperature. Pixels with a higher intensity value have higher temperature, while the background pixels have the lowest temperature. The thinning task is considered as an inverse process of heat conduction. The thinning process follows the inverse of the temperature flow, from cooler (background pixels) to warmer areas (object pixels with higher intensity), as in Fig. 10.

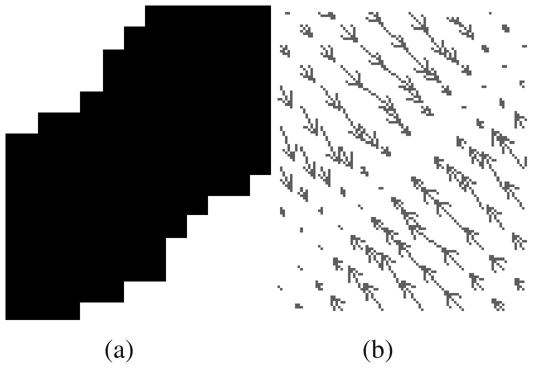


Fig. 10. Original image (a) and its heat-flow direction map (b).

The first step of the algorithm is the calculation of the Heat-Flow Direction Map (HFDM), which gives the directions and magnitude of the time-inversed heat flow for each point of the image. Then for each pixel of the image the intensity is subtracted and added to the pixel selected based on information provided by the HFDM. The latter step is iterated until a completely thinned image is obtained. The disadvantages of the proposed method lie in the fact that it does not guarantee a one-pixel width skeleton and that the computation of the right number of iterations to preserve the shape topology is a difficult task.

Quadros *et al.* (2004) proposed an algorithm to generate a disconnected, three-dimensional (3D) skeleton. A discrete skeleton is generated by propagating a wave from the boundary towards the interior on an octree lattice of an input solid model. As the wave propagates, the distance from the boundary and the direction of the wave front are calculated at the lattice-nodes (vertices) of the new front. An example of their results is shown in Fig. 11.

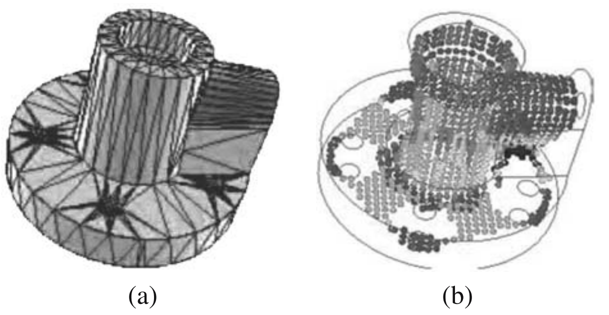


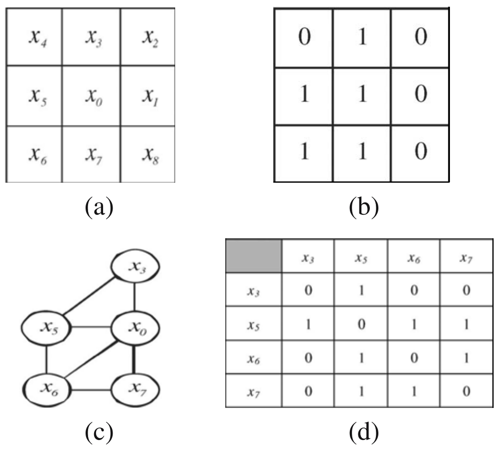
Fig. 11. Example of a 3D skeleton (Quadros *et al.*, 2004) obtained by wave propagation: graphics facets of input (a), trimmed discrete 3D skeleton (b).

Rockett's algorithm (Rockett, 2005) aimed at improving the rule based algorithm presented by Ahmed and Ward (2002). They observed the number of cases where the A-W algorithm failed to produce a centre line of a single pixel width. To eliminate such cases they adopted a two-stage thinning procedure which used the A-W rules to thin down to a skeleton which includes 2-pixel wide lines. As a second stage, they examined the 2-pixel wide lines in the skeleton produced by the first processing stage to see which pixels can be deleted without compromising the connectivity of the skeleton. In order to determine if the deletion of a pixel in a 2-pixel wide line disrupts connectivity, they built an adjacency matrix from an undirected graph of the local pixel connectivity over its eight neighbours (Fig. 12).

To retain connectivity, before deleting the pixel the algorithm checks if every row (or column) of the adjacency matrix contains at least one non-zero entry. An example of Rockett's results is presented in Fig. 13.

Pervouchine and Leedham (2005) constructed the skeleton in three steps directly from the greyscale image and represent it as a set of curves which in turn are represented as cubic B-splines. The method was designed to extract the skeleton which is very close to human perception of the original pen tip trajectory (Fig. 14).

Ju *et al.* (2007) proposed a method for computing skeletons of volumetric models by alternating thinning and a novel skeleton pruning routine. The presented method creates a family of skeletons parameterized by two user-specified numbers that determine respectively the size of the curve and surface features on the skeleton.


(a)

x_4	x_3	x_2
x_5	x_0	x_1
x_6	x_7	x_8

(b)

0	1	0
1	1	0
1	1	0

(d)

	x_3	x_5	x_6	x_7
x_3	0	1	0	0
x_5	1	0	1	1
x_6	0	1	0	1
x_7	0	1	1	0

Fig. 12. Pixel numbering convention (a), example of pixel configuration (b), resulting graph (c), and adjacency matrix (d).

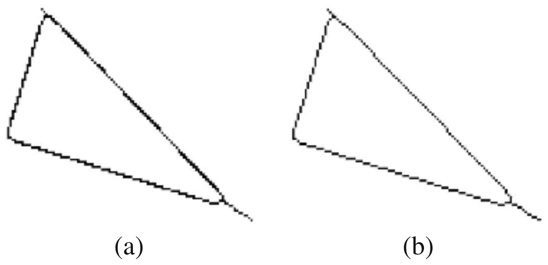


Fig. 13. Example of a skeleton obtained using Ahmed and Ward's method (a), compared with the result of Rockett's algorithm (b).

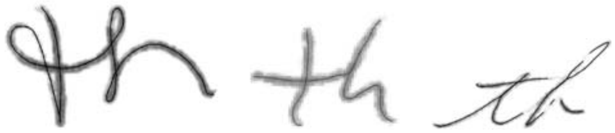


Fig. 14. Examples of greyscale images of handwritten letters and their skeletons extracted by means of the method presented in (Pervouchine and Leedham, 2005).

You and Tang (2007) extracted the skeleton of a character on the basis of the wavelet transform (Mallat, 1989). The skeletonization algorithm starts with the extraction of a primary skeleton in a regular region followed by amendment processing of the primary skeletons and connecting them in the singular region (Fig. 15).

Wan *et al.* (2008) proposed a three-stage skeletonization algorithm for shaped fiber recognition. In the first stage, the Euclidean distance transform is applied to the fiber image. In the second stage, the local maximal disc centres are determined as the true skeleton points, regar-

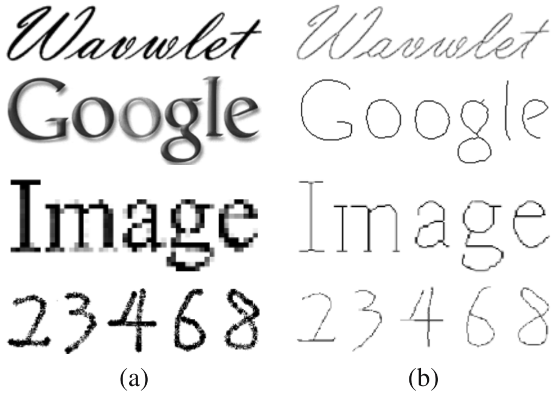


Fig. 15. Examples of images (a) and their thinned versions (b) using the algorithm described in (You and Tang, 2007).

dless of their connectivity. In the last step, a full skeleton is generated by linking isolated skeleton points, and pruning spurs arising from edge noise (Fig. 16).

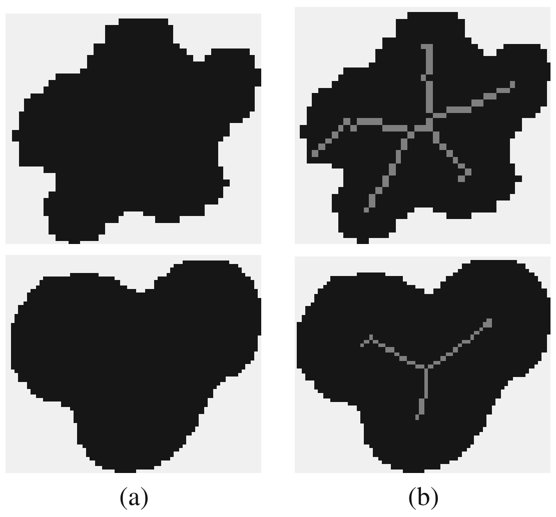


Fig. 16. Examples of star and trilobal shaped fibres (a) and their skeletons (b) obtained using the method presented in (Wan *et al.*, 2008).

Table 1 concludes the presentation of selected works by giving a summary of their most important features.

3. K3M: A modified KMM algorithm

This section presents the K3M (improved and generalized KMM) thinning algorithm in detail. First, definitions and assumptions are presented, and then a detailed algorithm description is introduced with flowcharts and neighbourhood lookup arrays.

3.1. Assumptions and definitions.

1. According to K3M, the aim of thinning is defined as the reduction of the number of pixels of an image that will preserve the structure of the image very well. As a result, after the transformation, lines that contribute to the letter are of a shape similar to the original and cross in a similar way.

2. The input image is binary, where each pixel is either an *image pixel* (encoded as 1) or a *background pixel* (encoded as 0). Therefore, colour and grey-scaled images are transformed to binary form.

3. The K3M algorithm is a sequential iterative algorithm: the iterations are repeated in sequence until no modification is made to the image during the whole iteration.

4. Each *iteration* involves seven *phases* (numbered from 0 to 6). The number of phases is determined in an empirical way.

5. *Pixel neighbours* are *image pixels* that are in the close vicinity of the tested pixel.

6. *Border pixels* are those *image pixels* that stick to the background (they have no more than seven neighbours). Therefore, they are considered potential candidates for deletion.

7. *Thinning decisions* are made for *border pixels*, based on 3×3 neighbourhood templates of the tested pixel. The 3×3 neighbourhood was chosen because of its large size capable to provide satisfactory thinning results. Larger neighbourhoods, however, would demand more computational complexity and execution time.

8. The *thinning decision* is either *to delete the pixel* (change an *image pixel* into a *background one*) or *to retain it* (the *image pixel* is not changed). The possible neighbourhood configurations are encoded using values in the range $[0, 255]$ named *neighbour weights*. The encoded neighbourhood configurations represented by *neighbour weights* are individual for each phase and are stored in *lookup arrays*.

9. After the iterations are finished, a supplemental procedure is required to produce a one-pixel width skeleton. This is because the iterative part of the algorithm is aimed at producing a fine skeleton and hence the algorithm is unable to produce a one-pixel width skeleton.

3.2. Algorithm description. K3M consists of an iterative part of seven phases and a single additional phase at the end, which is responsible for producing a one pixel-width skeleton. The overview of the algorithm is presented in Fig. 17.

Table 1. Basic algorithms of thinning.

Year	Author	Type	Remarks
1966	Rutovitz (1966)	parallel, iterative, not divided into subcycles	can result in excessive erosion, does not reduce diagonal lines to a unit pixel width, the skeleton does not lie centrally
1967	Blum (1967)	medial axis transformation, noniterative	
1969	Hilditch (1969)	sequential	maintain connectivity, preserve two-pixel wide lines
1975	Rosenfeld (1975)	parallel	solves the problem of erosion in the thinning of diagonal lines
1978	Arcelli and di Baja (1978)	parallel	does not remove all deletable pixels
1979	Dyer and Rosenfeld (1979)	interative	directly thins gray-scale images
1980	Pavlidis (1980)	sequential, combined sequential and parallel	does not result in a one-pixel-wide skeleton
1981	Arcelli (1981)	sequential	uses contour tracing to reduce computational time
1983	Favre and Keller (1983)	one-subcycle parallel	
1985	Ammann and Sartori-Angus (1985)	sequential	uses image compression before thinning to reduce computational time
1987	Chin *et al.* (1987)	1-subcycle, parallel	skeleton biased in line junctions
1988	Baruch (1988)	noniterative	produces skeleton in one pass, by line following
1989	Guo and Hall (1989)	iterative, 2-subcycles	does not give a one-pixel-wide skeleton
1989	Chen and Hsu (1989a; 1989b)	1-subcycle, parallel	bias-reduced skeletons
1994	Parker *et al.* (1994)	force-based	acquires a skeleton by using force vectors
1996	Zhang and Wang (1996)	2-subcycles, parallel	preserves connectivity, produces one-pixel-wide skeletons
1998	Altuwaijri and Bayoumi (1998)	neural networks based	redundant loops removal—disadvantage in certain alphabets
2000	Andreadis *et al.* (2000)	interative	obtains skeletons from color images
2001	KMM (Saeed *et al.*, 2001)	sequential, iterative	maintain connectivity, bias-reduced skeletons
2002	Ahmed and Ward (2002)	parallel	fails to produce a 1-pixel width line in some patterns
2003	Huang *et al.* (2003)	parallel	
2004	Ji and Feng (2004)	noniterative	does not guarantees one-pixel width skeleton, problems with choosing the right number of iterations
2004	Quadros *et al.* (2004)	3-dimensional	uses wave propagating from the boundary towards the interior
2005	Rockett (2005)	parallel	produces one-pixel-wide skeletons, uses a two-stage thinning procedure
2005	Pervouchine and Leeedham (2005)	gray-scale image thinning	represents a skeleton as cubic B-splines
2007	Ju *et al.* (2007)		creates a family of skeletons with various sizes of the curve and surface features
2007	You and Tang (2007)	wavelet based	extracted skeleton is centered inside the underlying stroke
2008	Wan *et al.* (2008)	distance map based	three-stage skeletonization algorithm for shaped fiber recognition

Iterative part overview

- Phase 0: Mark *borders* for examination.
- Phase 1: *Delete* the *borders* that have **3** neighbours sticking each other.
- Phase 2: *Delete* the *borders* that have **3** or **4** neighbours sticking each other.
- Phase 3: *Delete* the *borders* that have **3**, **4** or **5** neighbours sticking each other.
- Phase 4: *Delete* the *borders* that have **3**, **4**, **5** or **6** neighbours sticking each other.
- Phase 5: *Delete* the *borders* that have **3**, **4**, **5**, **6** or **7** neighbours sticking each other.
- Phase 6: Unmark the remaining *borders*.
- Decision: If any modification was made during the current *iteration*, return to Phase 0.

Thinning to a one-pixel width skeleton. This phase aims at producing skeletons consisting of pixels with only two neighbours unless located at a junction. This is a significant advantage of a one-pixel width skeleton as it potentially allows producing structural graphs of the examined image.

The essential property of each pixel, which is used for quick determination of its neighbourhood configuration, is the *neighbour weight*. It is an 8-bit number, where subsequent bit values correspond to neighbours, starting from the pixel above and going clockwise. It is calculated with the use of the neighbourhood bit values matrix (3), as in (2), where $w(x, y)$ is the neighbour weight of pixel (x, y) and $img(x, y)$ is the binary value of the image pixel at coordinates (x, y).

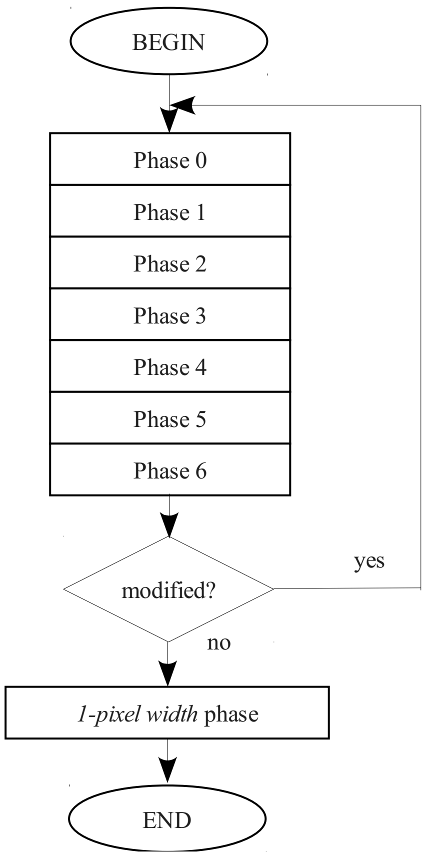


Fig. 17. K3M algorithm simplified flowchart.

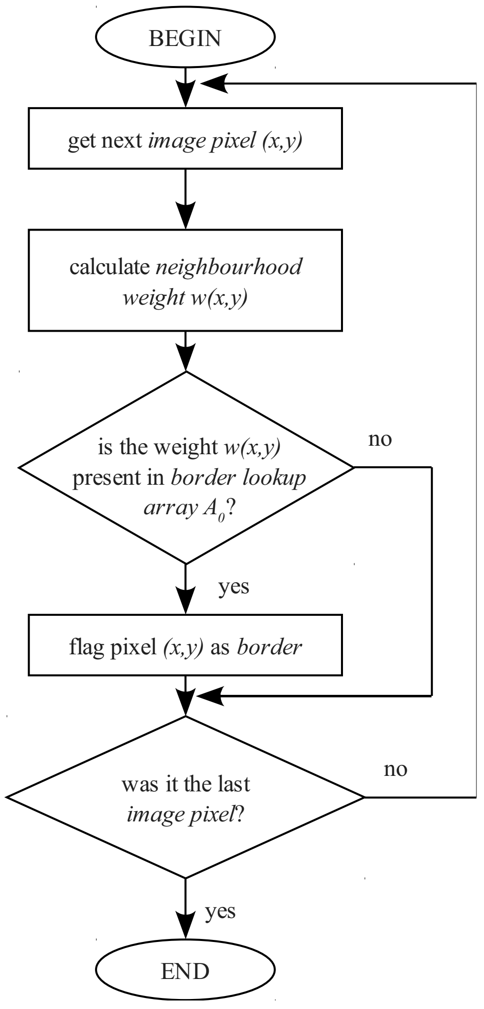

BEGIN
get next image pixel (x,y)
calculate neighbourhood weight w(x,y)
is the weight w(x,y) present in border lookup array A0?
no
yes
flag pixel (x,y) as border
was it the last image pixel?
no
yes
END

Fig. 18. Flowchart of phase 0—marking borders.

Neighbour weight calculation

$$w(x, y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} N(i + 1, j + 1) \cdot img(x + i, y + j). \tag{2}$$

Neighbourhood bit values matrix

$$N = \begin{bmatrix} 128 & 1 & 2 \\ 64 & 0 & 4 \\ 32 & 16 & 8 \end{bmatrix}. \tag{3}$$

Figure 18 presents the flowchart of Phase 0, aimed at marking borders (pixels that are candidates for deletion).

The iterative phases 1 to 5 are very similar to each other and aim at deleting pixels with a growing number of sticking neighbours. The phases may be presented using a common flowchart with parameter i that changes from 1 to 5 (see Fig. 19) and determines the employed lookup array A_i.

3.3. Neighbourhood lookup arrays. In this section some specific *neighbourhood lookup arrays* are given. They serve for marking borders and determining *thinning decisions*. The components of arrays A_i (Phases 1 to 5) follow the rules mentioned in the algorithm description concerning the number of neighbours. The contents of array A_{1pix} (thinning to a one-pixel width skeleton) was determined on the basis of several experiments and the authors' original KMM algorithm (Saeed *et al.*, 2001).

List of neighbourhood lookup arrays

- A_0: *lookup* array for Phase 0—marking *borders*,
- A_1: *lookup* array for Phase 1—deleting pixels having 3 sticking neighbours,
- A_2: *lookup* array for Phase 2—deleting pixels having 3 or 4 sticking neighbours,
- A_3: *lookup* array for Phase 3—deleting pixels having 3, 4 or 5 sticking neighbours,

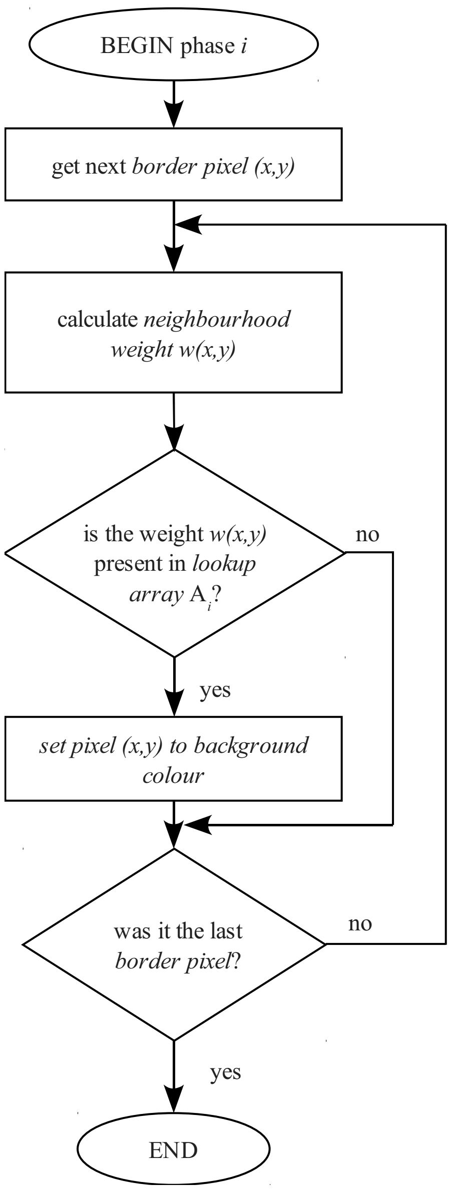

BEGIN phase *i*
get next *border pixel (x,y)*
calculate *neighbourhood weight w(x,y)*
is the weight *w(x,y)* present in *lookup array* A_i?
no
yes
set pixel (x,y) to background colour
was it the last *border pixel*?
no
yes
END

Fig. 19. Flowchart of Phase i ($i \in \{1, 2, 3, 4, 5\}$) deleting pixels.

- A_4: *lookup* array for Phase 4—deleting pixels having 3, 4, 5 or 6 sticking neighbours,
- A_5: *lookup* array for Phase 5—deleting pixels having 3, 4, 5, 6 or 7 sticking neighbours,
- A_{1pix}: *lookup* array used for thinning to a one-pixel width skeleton.

Components of neighbourhood lookup arrays

- $A_0 = \{3, 6, 7, 12, 14, 15, 24, 28, 30, 31, 48, 56, 60, 62, 63, 96, 112, 120, 124, 126, 127, 129, 131, 135, 143, 159, 191, 192, 193, 195, 199, 207, 223, 224, 225, 227, 231, 239, 240, 241, 243, 247, 248, 249, 251, 252, 253, 254\}$,
- $A_1 = \{7, 14, 28, 56, 112, 131, 193, 224\}$,
- $A_2 = \{7, 14, 15, 28, 30, 56, 60, 112, 120, 131, 135, 193, 195, 224, 225, 240\}$,
- $A_3 = \{7, 14, 15, 28, 30, 31, 56, 60, 62, 112, 120, 124, 131, 135, 143, 193, 195, 199, 224, 225, 227, 240, 241, 248\}$,
- $A_4 = \{7, 14, 15, 28, 30, 31, 56, 60, 62, 63, 112, 120, 124, 126, 131, 135, 143, 159, 193, 195, 199, 207, 224, 225, 227, 231, 240, 241, 243, 248, 249, 252\}$,
- $A_5 = \{7, 14, 15, 28, 30, 31, 56, 60, 62, 63, 112, 120, 124, 126, 131, 135, 143, 159, 191, 193, 195, 199, 207, 224, 225, 227, 231, 239, 240, 241, 243, 248, 249, 251, 252, 254\}$,
- $A_{1pix} = \{3, 6, 7, 12, 14, 15, 24, 28, 30, 31, 48, 56, 60, 62, 63, 96, 112, 120, 124, 126, 127, 129, 131, 135, 143, 159, 191, 192, 193, 195, 199, 207, 223, 224, 225, 227, 231, 239, 240, 241, 243, 247, 248, 249, 251, 252, 253, 254\}$.

3.4. Modifications to KMM. K3M may be seen as an evolved KMM algorithm as both of them are based on iterative sequential thinning. K3M is more systematic. KMM, however, was essentially based on a single lookup array while K3M uses many of them in an understandable pattern. Therefore, K3M involves more iterative phases, namely, six, in comparison with KMM, which contains only four. The main motivation to develop a new K3M algorithm comes from the fact that the previous attempt (KMM) was to develop a more systematic approach to image thinning. As was verified by experiments, the results are subjectively better so the attempt has proved its success rate.

4. K3M results and comparison

In this section the most interesting thinning results of the K3M algorithm are presented and discussed. We will therefore test the algorithm on a variety of inputs—isolated letters, handwritten words and, finally, graphics and symbols, regarding skeleton continuity, shape preservation and a 1-pixel-wide skeleton. The goal is to verify if the performance complies with the expectations.

4.1. Isolated letters. As can be observed in Fig. 20, K3M preserves the letter shapes. For numerous applications it is especially important and essential to maintain the original connectivity of the script lines as well as the straight angles. K3M does an excellent job here, as can be seen in the letter ‘B’, for example.

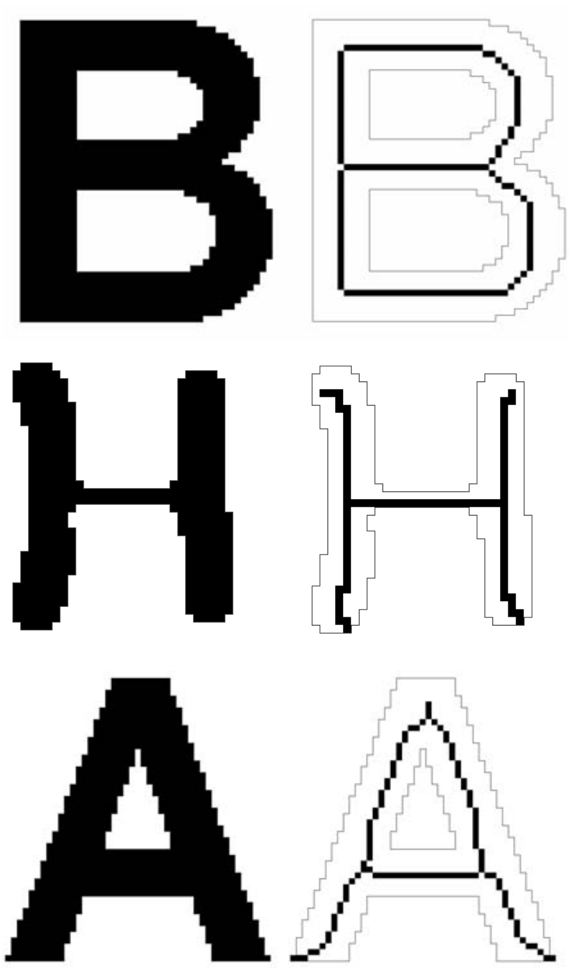


Fig. 20. Thinning isolated letters with K3M.

4.2. Handwritten words. In most cases printed words are thinned (with some exception in the case of bad quality texts) in the same way as several isolated letters. Handwritten words, however, are interesting as they contain one or more connections between individual letters, extra loops, and irregularities, which is worth studying.

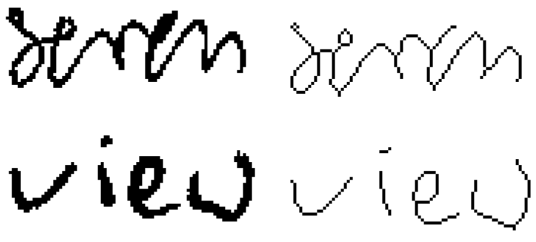


Fig. 21. Handwritten words thinned with K3M.

As seen in Fig. 21, the overall shapes of the examined words are well preserved. However, for possibly closed-loops letters, like the second 'e' in the word 'seven', it is obvious that some information is lost. It is impossible to avoid such problems without additional steps to create a real loop from such a letter. Irregular dots are transferred to short lines rather than to real dots, which is a common trait for most thinning algorithms.

4.3. Graphical symbols. In thinning graphical symbols (Fig. 22) and graphic-alike alphabets (Fig. 23), it is especially important to keep the right angles and interconnections close in terms of the shape to the original picture. One can see that these aspects are well managed by the K3M algorithm. However, occasional deformations of originally straight diagonally placed lines can be observed.

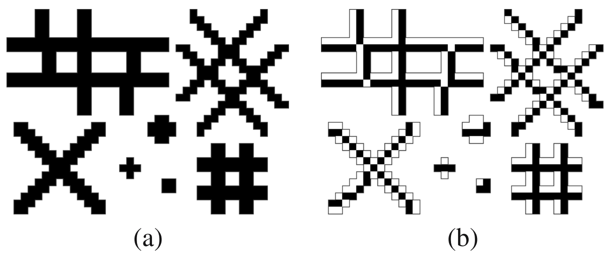


Fig. 22. Thinning graphical symbols with K3M: original images (a), their thinned shapes (b).

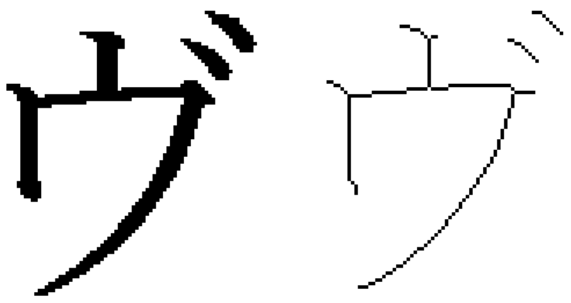


Fig. 23. Results of thinning a Japanese symbol with K3M.

4.4. Comparison of results between K3M and other algorithms. Table 2 shows the results of thinning when using the K3M approach in comparison with some of the above-mentioned algorithms.

Table 3, however, presents a comparison between K3M and other algorithms considering several important aspects, like the output width, angles preservation and topology preservation.

4.5. Advantages and drawbacks of K3M. K3M has the following feasible advantages:

Table 2. Results of thinning by Guo and Hall's, Zhang's, A-W, KMM and K3M algorithms.

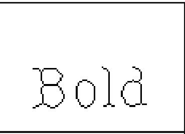
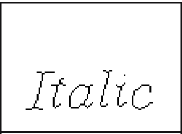
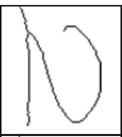
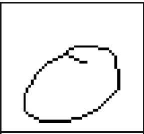
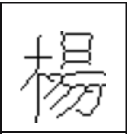
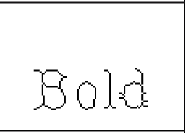
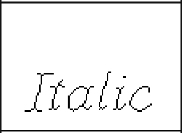
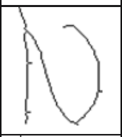
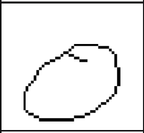
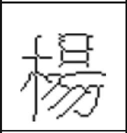
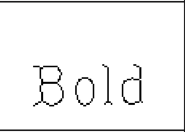
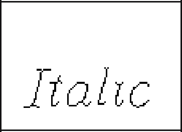
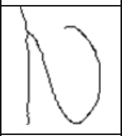
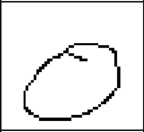
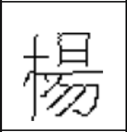
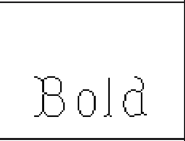
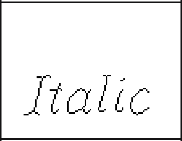
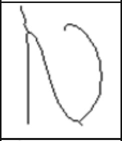
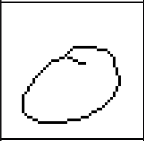
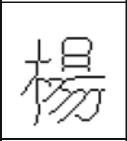
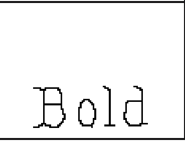
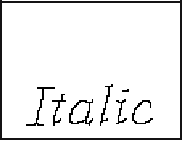
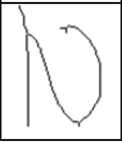
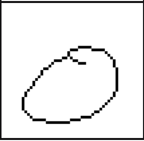
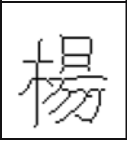
Method	Image				
Guo and Hall's (Guo and Hall, 1992)					
Zhang's (Zhang and Wang, 1996)					
A-W (Ahmed and Ward, 2002)					
KMM (Saeed *et al.*, 2001)					
K3M					

Table 3. Comparison of results: K3M and others.

Algorithm	Guo and Hall (1992)	Zhang and Wang (1996)	Ahmed and Ward (2002)	KMM (Saeed *et al.*, 2001)	K3M
1-pixel wide	–	+	–	+	+
bias-reduced skeleton (preserves right angles)	–	–	–	–	+
preserves topology (no disappearances)	–	+	–	+	+

1. The algorithm preserves right angles at the lines interconnections, which result in better correspondence between the original and the modified image.
2. It produces a one-pixel-wide skeleton.
3. The iterative phases are clearly aimed at deleting pixels in a specific neighbourhood, therefore a general idea of this thinning method is easily found.
4. The universality of the algorithm allows wide and diversified applications.

Although the K3M algorithm has successfully proved its universal performance over many other algorithms, it is intrinsically iterative in nature, and hence it requires more computing power than other, non-iterative algorithms. Moreover, each iteration contains seven sequential phases. This increases the computational complexity and makes the parallelization attempts difficult.

Despite the fact that these drawbacks do not affect the shape preservation, the authors are currently working on reducing the algorithm complexity and computational power requirements. The new results will be the matter of a future publication after proving their validity.

5. Parallelizing sequential algorithms

Non-one-pass iterative thinning algorithms (sequential algorithms with one or more subcycles) are not intrinsically suitable for parallel processing. Therefore, they tend to have difficulties with successful utilisation of modern parallel data processing techniques like multi-core processors or processor grids (hardware or network implemented). With the growing of the usage and popularity of parallel processing hardware and its techniques, it is basic to develop solutions that would allow such algorithms to profit from the use of multi-processors.

In Fig. 24 one can see *scanline*-type sequential processing, where rows are processed sequentially. For each pixel, a decision is made based on the state of the neighbours.

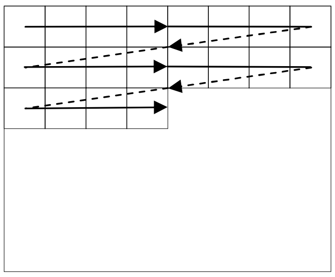


Fig. 24. Sequential *scanline*-type processing.

The steps of the iterative thinning algorithm are repeated several times in their consecutive iterations. Each processing step may contain several sub-cycles with operations such as flagging some pixels (for example, as candidates for deletion or as non-deletable pixels). Furthermore, sub-cycles perform their operations depending, among other things, on that information. The final decision depends on the flags of neighbours set in the previous subcycles, making the algorithm impossible to perform in a parallel way as there exists information dependence between neighbouring pixels (information on the actual state of neighbours is required at the specified subcycle).

The information dependence area grows with each subcycle included in the iteration, as the state of neighbours depends on the states of their neighbours, and so on. The growth of the effective neighbourhood is presented in Fig. 25.

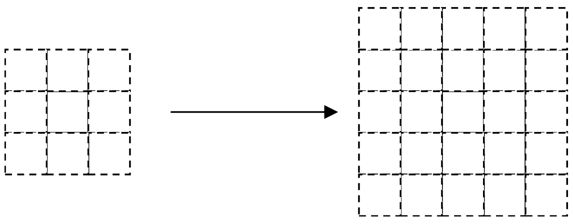


Fig. 25. Dependence area growth due to information dependence.

There may be several solutions that allow parallel processing of such problematic thinning algorithms:

- *Modification of the thinning algorithm to diminish or eliminate subcycles.* In the case of elimination, it leads to the possibility of fully parallel processing. In the case of decreasing the number of subcycles, it diminishes the information dependence area.
- *Division of the image in such a way that information dependence is non-existent or irrelevant.* This is not always possible.
- *Division with overlapping regions*—regular division while providing indispensable information (information dependence areas from the area of division) for each parallel processor.

5.1. Division with overlapping regions. In the division process, the image is divided into some overlapping parts to be processed in parallel. These parts are called *regions*. Figure 26 shows how an image space is divided into two regions P1 and P2. Figure 27, however, shows how they are overlapped.

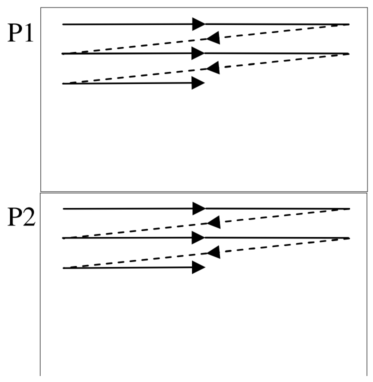


Fig. 26. Parallel *scanline*-type processing on different image regions P1 and P2.

Obviously, in the region edge each pixel has its definite number of neighbours. We can then deduce that the neighbouring pixel weights have a basic role in the decision taking process during the thinning procedure. The edge pixels of P1 in the *crossing border* between P1 and P2 actually stick the neighbours of other pixels belonging to P2. That is why the edge pixels from two regions share neighbours and hence some useful data are available to both parts.

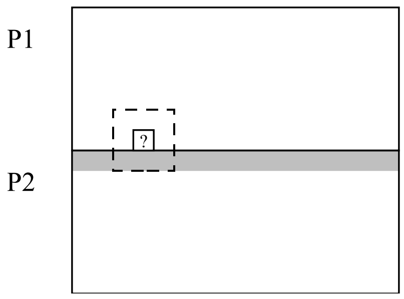


Fig. 27. Pixel in the region edge.

In most algorithms (including KMM and K3M) the neighborhood is of a 3×3 character, but the approach is also valid for 3×4 or 5×5 neighborhood basis algorithms.

The overlapping area serves as the container of information needed for neighbourhood checking. The width of the region depends on the number of subcycles in the thinning algorithm and the requested number of independent parallel cycles of the algorithm. Such a configuration may be presented as read/write privileges of processes (as seen in Fig. 28) assuming that the processes operate on a shared memory.

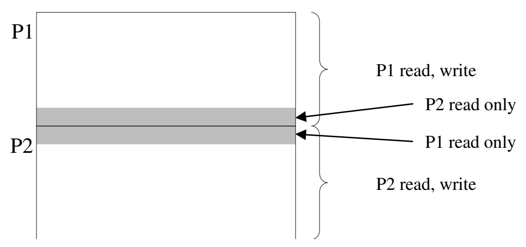


Fig. 28. Overlapping regions.

Pseudocode of the non-parallelized thinning algorithm

```
while not thinned do
    foreach pixel in image
        check pixel neighbours
        do thinning routines
    end foreach
end do
```

Pseudocode of the parallelized thinning algorithm

```
iteration := 1
while not thinned do
  foreach pixel in image
    if neighbour in border band then
      while neighbour.iteration < iteration-1 do
        wait for synchronization
      end while
    end if
    check pixel neighbours
    do thinning routines
    if pixel in border band then
      send synchronization info
    end if
  end foreach
  iteration := iteration + 1
end do
```

5.2. Prediction of the neighbouring processor decision. In some cases it is possible to predict the decision that a neighbouring processor should make. In particular, if a pixel in the border area is “white,” no change can be made in conventional thinning (with a possible exception of thinning algorithms that perform thinned shapes smoothing). That may particularly simplify the synchronization task as presented above.

6. Conclusions and future work

The first aspect presented in the paper was the survey presentation of the state of the art in the thinning area. A very large number of algorithms in this field have been proposed and due to the complicated nature of skeletonization it is often unclear how the approaches are related to each other in terms of processing quality and the general idea. A number of important works ranging from the earliest to the latest publications on thinning were discussed and compared. Short summaries with the available results of those algorithms were also presented. For the sake of general comparison, essential and basic traits of the algorithms were shown. The main points of view were the type of methodology used and the parallelization possibility (intrinsic or secondary) with final general remarks on the obtained results or the dedicated application of the studied algorithm. The survey is considered a background for the authors’ proposal of the thinning algorithm named K3M.

The second important aspect of the paper is the presentation of the authors’ algorithm evolved from the original KMM algorithm. The proposed algorithm introduces its advantages in terms of several important characteristics: thinning quality, a large range of possible applications and the clarity of processing stages. The examples shown in Section 4 for isolated letters, words and graphical symbols demonstrate the processing quality of K3M. The examined examples showed that the K3M algorithm presents constant quality for printed words, handwritten scripts, numbers and characters of both Latin and non-Latin alphabets. This fact proves the universal character of the K3M algorithm compared with other algorithms known to the authors, which can be considered the most important advantage.

The next task, which has been proved to be true in practical applications, is that the idea of the algorithm is clear and simple to follow. This was mainly demonstrated in the detailed description in Section 3. The clarity and simple nature of the algorithm make it easy to study and use, which is significant when compared with other, often empirically obtained, algorithms.

The last important aspect shown and discussed in the paper regards parallelization issues for sequential thinning algorithms. This matter was considered in Section 5 and could certainly benefit in improving many sequential iterative thinning algorithms. It is particularly useful in the context of the recent popularity of parallel processing hardware and software techniques.

Future work is mainly aimed at two directions: the first is related to fine-tuning of the algorithm in order to eliminate the slight deformation of some lines currently occurring. The introduced changes, however, should not hassle the universality of the applications of the algorithm. The second development direction is the reduction of the number of phases in the iterative part of the K3M algorithm. That would result in lowering the computational effort as well as improving the parallelization possibility of the algorithm.

Acknowledgment

This work is supported by the Rector of Białystok Technical University (grant no. W/WI/10/07).

The authors are indebted to the anonymous referees for their constructive and valuable comments, and for their extremely helpful suggestions that improved the quality of the presented work.

References

Ahmed, M. and Ward, R. (2002). A rotation invariant rule-based thinning algorithm for character recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(12): 1672–1678.

Altuwaijri, M.M. and Bayoumi, M.A. (1998). A thinning algorithm for arabic characters using art2 neural network, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* **45**(2): 260–264.

Ammann, C.J. and Sartori-Angus, A.G. (1985). Fast thinning algorithm for binary images, *Image Vision and Computing* **3**(2): 71–79.

Andreadis, I., Vardavoulia, M., Louverdis, G. and Papamarkos, N. (2000). Colour image skeletonisation, *Proceedings of the 10th European Signal Processing Conference, Tampere, Finland*, Vol. 4, pp. 2389–2392.

Arcelli, C. (1981). Pattern thinning by contour tracing, *Computer Graphics Image Processing* **17**(2): 130–144.

Arcelli, C. and di Baja, G.S. (1978). On the sequential approach to medial line transformation, *IEEE Transactions on Systems, Man and Cybernetics* **8**(2): 139–144.

Arcelli, C. and di Baja, G.S. (1981). A thinning algorithm based on prominence detection, *Pattern Recognition* **13**(3): 225–235.

Arcelli, C. and di Baja, G.S. (1987). A contour characterization for multiply connected figures, *Pattern Recognition Letters* **6**(4): 245–249.

Arcelli, C. and di Baja, G.S. (1989). A one-pass two-operation process to detect the skeletal pixels on the 4-distance transform, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**(4): 411–414.

Baruch, O. (1988). Line thinning by line following, *Pattern Recognition Letters* **8**(4): 271–276.

Blum, H. (1967). A transformation for extracting new descriptors of shape, *in* W.W. Dunn (Ed.), *Models for the Perception of Speech and Visual Form*, MIT Press, Cambridge, MA, pp. 362–380.

Chen, Y.-S. (1996). The use of hidden deletable pixel detection to obtain bias-reduced skeletons in parallel thinning, *ICPR '96: Proceedings of the 13th International Conference on Pattern Recognition*, *Washington, DC, USA,* pp. 91–95.

Chen, Y.-S. and Hsu, W.-H. (1988). A modified fast parallel algorithm for thinning digital patterns, *Pattern Recognition Letters* **7**(2): 99–106.

Chen, Y.-S. and Hsu, W.-H. (1989a). A 1-subcycle parallel thinning algorithm for producing perfect 8-curves and obtaining isotropic skeleton of an l-shape pattern, *Proceedings of the International Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA*, pp. 208–215.

Chen, Y.-S. and Hsu, W.-H. (1989b). A systematic approach for designing 2-subcycle and pseudo 1-subcycle parallel thinning algorithms, *Pattern Recognition* **22**(3): 267–282.

Chen, Y.-S. and Hsu, W.-H. (1990). A comparison of some one-pass parallel thinnings, *Pattern Recognition Letters* **11**(1): 35–41.

Chin, R.T., Wan, H.-K., Stover, D.L. and Iverson, R. D. (1987). A one-pass thinning algorithm and its parallel implementation, *Computer Vision, Graphics, and Image Processing* **40**(1): 30–40.

Dinnen, G. (1955). Programming pattern recognition, *Proceedings of the Western Joint Computer Conference, Los Angeles, CA, USA*, pp. 94–100.

Dyer, C. and Rosenfeld, A. (1979). Thinning algorithms for gray-scale pictures, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1**(1): 88–89.

Favre, A. and Keller, H. (1983). Parallel syntactic thinning by recoding of binary pictures, *Computer Vision, Graphics, and Image Processing* **23**(1): 99–112.

Gonzalez, R.C. and Woods, R.E. (2007). *Digital Image Processing, 3rd Edition*, Prentice Hall, Upper Saddle River, NJ.

Guo, Z. and Hall, R.W. (1989). Parallel thinning with two-subiteration algorithms, *Communications of the ACM* **32**(3): 359–373.

Guo, Z. and Hall, R.W. (1992). Fast fully parallel thinning algorithms, *CVGIP: Image Understanding* **55**(3): 317–328.

Hilditch, C.J. (1968). An application of graph theory in pattern recognition, *Machine Intelligence* **3**: 325–347.

Hilditch, C.J. (1969). Linear skeletons from square cupboards, *Machine Intelligence* **4**: 403–420.

Huang, L., Wan, G. and Liu, C. (2003). An improved parallel thinning algorithm, *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, *Washington, DC, USA,* pp. 780–783.

Jaisimha, M.Y., Haralick, R.M. and Dori, D. (1994). Quantitative performance evaluation of thinning algorithms under noisy conditions, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA*, pp. 678–683.

Jang, B.K. and Chin, R.T. (1992). One-pass parallel thinning: Analysis, properties and quantitative evaluation, *Transactions on Pattern Analysis and Machine Intelligence* **14**(11): 1129–1140.

Ji, X. and Feng, J. (2004). A new approach to thinning based on time-reversed heat conduction model, *International Conference on Image Processing (ICIP), Singapore*, Vol. 1, pp. 653–656.

Ju, T., Baker, M.L. and Chiu, W. (2007). Computing a family of skeletons of volumetric models for shape description, *Computer-Aided Design* **39**(5): 352–360.

Kirsch, R.A., Cahn, L., Ray, C. and Urban, G.H. (1958). Experiments in processing pictorial information with a digital computer, *IRE-ACM-AIEE '57 (Eastern): Papers and Discussions Presented at the December 9-13, 1957, Eastern Joint Computer Conference: Computers with Deadlines to Meet*, *New York, NY, USA,* pp. 221–229.

Klette, R. and Rosenfeld, A. (2004). *Digital Geometry: Geometric Methods for Digital Image Analysis*, Morgan Kaufmann, San Francisco, CA.

Lam, L., Lee, S.-W. and Suen, C.Y. (1992). Thinning methodologies—A comprehensive survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(9): 869–885.

Lee, S.-W., Lam, L. and Suen, C. Y. (1991). Performance evaluation of skeletonizing algorithms for document image processing, *Proceedings of the First International Conference on Document Analysis and Recognition, Saint-Malo, France,* pp. 260–271.

Malina, W., Ablemeyko, S. and Pawlak, W. (2002). *Fundamentals of Digital Image Processing*, Akademicka Oficyna Wydawnicza EXIT, Warsaw, (in Polish).

Mallat, S. (1989). A theory for multiresolution signal decomposition: The wavelet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**(7): 674–693.

Parker, J., Jennings, C. and Molaro, D. (1994). A force-based thinning strategy with sub-pixel precision, *Vision Interface Conference, Banff, Alberta, Canada*.

Pavlidis, T. (1980). A thinning algorithm for discrete binary images, *Computer Graphics Image Processing* **13**(2): 142–157.

Pavlidis, T. (1981). A flexible parallel thinning algorithm, *Proceedings of the International Conference on Pattern Recognition and Image Processing, Dallas, TX, USA*, pp. 162–167.

Pavlidis, T. (1982a). *Algorithms for Graphics and Image Processing*, Springer, Computer Science Press, Rockville, MD.

Pavlidis, T. (1982b). An asynchronous thinning algorithm, *Computer Graphics Image Processing* **20**(2): 133–157.

Pervouchine, V. and Leedham, G. (2005). Document examiner feature extraction: Thinned vs. skeletonised handwriting images, *Proceedings of the IEEE Region 10 Technical Conference (TENCON05), Melbourne, Australia,* pp. 1–6.

Quadros, W. R., Shimada, K. and Owen, S. J. (2004). 3d discrete skeleton generation by wave propagation on pr-octree for finite element mesh sizing, *SM '04: Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications*, *Aire-la-Ville, Switzerland*, pp. 327–332.

Rockett, P. I. (2005). An improved rotation-invariant thinning algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(10): 1671–1674.

Rosenfeld, A. (1975). A characterization of parallel thinning algorithms, *Information and Control* **29**(3): 286–291, `http://theory.lcs.mit.edu/~iandc/ic75.html`

Rutovitz, D. (1966). Pattern recognition, *Journal of Royal Statistical Society* **129**(4): 504–530.

Saeed, K. (2001). Text and image processing: Non-interrupted skeletonization, *Proceedings of the 1st International IEEE Conference on Circuits, Systems, Comunications and Computers—IEEE-CSCC'01, Crete, Greece,* Advances in Scientific Computing, Computational Intelligence and Applications, World Scientific Engineering Society Press, pp. 350–354.

Saeed, K. (2004). *Image Analysis for Object Recognition*, Białystok Technical University Press, Białystok.

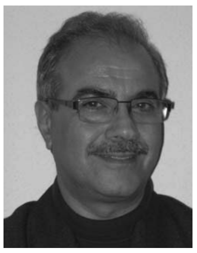
Saeed, K. and Niedzielski, R. (1999). Experiments on thinning of cursive-style alphabets, *International Conference on Information Technologies for Education, Science and Business ITESB'99*, Minsk, Belarus, pp. 45–49.

Saeed, K., Rybnik, M. and Tabedzki, M. (2001). Implementation and advanced results on the non-interrupted skeletonization algorithm, *in* W. Skarbek (Ed.) *Computer Analysis of Images and Patterns*, Lecture Notes in Computer Science, Vol. 2124, Springer-Verlag, Heidelberg, pp. 601–609.

Wan, Y., Yao, L., Xu, B. and Zeng, P. (2008). A distance map based skeletonization algorithm and its application in fiber recognition, *International Conference on Audio, Language and Image Processing*, Shanghai, China, pp. 1769–1774.

You, X. and Tang, Y. Y. (2007). Wavelet-based approach to character skeleton, *IEEE Transactions on Image Processing* **16**(5): 1220–1231.

Zhang, Y. Y. and Wang, P. P. (1996). A parallel thinning algorithm with two-subiteration that generates one-pixel-wide skeletons, *International Conference on Pattern Recognition, Vienna, Austria,* Vol. 4, pp. 457–461.



Khalid Saeed received the B.Sc. degree in electrical and electronic engineering in 1976 from Baghdad University, the M.Sc. (electronics) and Ph.D. (telecommunications) degrees from the Wrocław University of Technology, Poland, in 1978 and 1981, respectively. He received his D.Sc. degree (habilitation) in computer science from the Polish Academy of Sciences in Warsaw in 2007. From 1992 to 2008 he was with Białystok Technical University. Since 2008 he has been a professor of computer science at the AGH University of Science and Technology in Poland. He has published more than 120 works, about 20 edited books, and seven text and reference books. His areas of interest are image analysis and processing, biometrics and computer information systems.

Marek Tabędzki received his M.Sc. degree in computer science from Białystok Technical University (Poland) in 2001. He is presently with the same university, at the Faculty of Computer Science. His research interests include information processing systems, particularly digital image processing and pattern recognition.

Mariusz Rybnik received his M.Sc. degree in computer science from Białystok Technical University (Poland) in 2001, and the Ph.D. degree in engineering in biology and medicine from the University of Paris XII—Val de Marne (France) in 2004. He is currently an assistant professor at the Faculty of Mathematics and Computer Science of the University of Białystok. His research interests include image processing, pattern recognition, biometrics and artificial intelligence.

Marcin Adamski received his M.Sc. degree in computer science from Białystok Technical University (Poland) in 2001, where he currently works at the Faculty of Computer Science. His research interests concern image processing and image recognition methods, and their applications in biometric systems.

Appendix

Pseudocodes of the implemented computer programs

{creating N-lookup arrays}

```
{alternate to weight calculation}
function lookup(N:integer) : boolean
   vector = [pixel.N,  pixel.NE, pixel.E,
             pixel.SE, pixel.S,  pixel.SW,
             pixel.W, pixel.NW]
   changes := 0
   for i := 1 to 7
      if vector[i] <> vector[i+1] then
         changes := changes + 1
      end if
   end for
   if changes <= 2 then
      ones := 1
      for i := 1 to 8
         if vector[i] = 1 then
            ones := ones + 1
         end if
      end for
      if ones = N then
         return true
      end if
   end if
   return false
end function

pixel.N - north neighbour of actual pixel
pixel.NE - northeast neighbour of actual pixel
and so on...
param N - number of sticking neighbours needed
```

{classic 1-stage}

```
mark image pixels as 1
mark background pixels as 0
repeat
   thinned = true
   for each pixel in image
       if pixel = 1 then
           if (pixel.N mod 2 = 0) or
              (pixel.S mod 2 = 0) or
              (pixel.W mod 2 = 0) or
              (pixel.E mod 2 = 0) then
             pixel := 3
           end if
       end if
   end for
   for each pixel in image
       if pixel = 3 then
          { DLA - deletion lookup array }
          if weight(pixel) in DLA then
              pixel := 0
              thinned = false
          else
              pixel := 3
          end if
       end if
   end for
   for each pixel in image
       if pixel = 3 then
           pixel := 1
       end if
   end for
until thinned
mark even pixels as background
mark odd pixels as image
```

{neighbour weight calculation}

```
function weight(pixel) : integer
  return 1*(pixel.N mod 2) + 2*(pixel.NE mod 2)+
      64*(pixel.W mod 2) + 4*(pixel.E mod 2)+
      32*(pixel.SW mod 2) + 16*(pixel.S mod 2)+
      8*(pixel.SE mod 2) + 128*(pixel.NW mod 2)
end function
```

{single-for 1-stage}

```
mark image pixels as 1
mark background pixels as 0
phase := 1
repeat
  thinned = true
  for each pixel p in image
    if p mod 2 = 1 then
      { checking if border pixel }
      if (p.N <> 2*phase and p.N mod 2 = 0) or
        (p.S <> 2*phase and p.S mod 2 = 0) or
        (p.W <> 2*phase and p.W mod 2 = 0) or
        (p.E <> 2*phase and p.E mod 2 = 0) then
        { DLA - deletion lookup array }
        if weight(p) in DLA then
           p := 2 * phase
           thinned = false
        else
           p := 2 * phase + 1
        end if
      end if
    end if
  end for
  phase := phase + 1
until thinned
mark even pixels as background
mark odd pixels as image
```

Received: 14 June 2009
Revised: 18 October 2009