Int. J. Appl. Math. Comput. Sci., 2016, Vol. 26, No. 2, 439–450
DOI: 10.1515/amcs-2016-0031

A MODIFIED K3M THINNING ALGORITHM

MAREK TABEDZKI [a], KHALID SAEED [a,b,*], ADAM SZCZEPAŃSKI [a]

[a]Faculty of Computer Science
Białystok University of Technology, ul. Wiejska 45 A, 15-351 Białystok, Poland
e-mail: {m.tabedzki,k.saeed}@pb.edu.pl, aszczepa@agh.edu.pl

[b]Faculty of Mathematics and Information Sciences
Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warsaw, Poland

The K3M thinning algorithm is a general method for image data reduction by skeletonization. It had proved its feasibility in most cases as a reliable and robust solution in typical applications of thinning, particularly in preprocessing for optical character recognition. However, the algorithm had still some weak points. Since then K3M has been revised, addressing the best known drawbacks. This paper presents a modified version of the algorithm. A comparison is made with the original one and two other thinning approaches. The proposed modification, among other things, solves the main drawback of K3M, namely, the results of thinning an image after rotation with various angles.

Keywords: skeletonization, thinning, K3M algorithm, digital image processing.

1. Introduction

Thinning is an important stage in the image processing procedure of many systems. Mainly, it is one of the possible means of creation of the description of a shape. Thinning equalizes the deficiencies created during preprocessing, such as an uneven width of the lines or ragged edges. The resulting skeleton of the object is usually one- or two-pixel wide and preserves the distinctive features of the described object. Characteristic points of it, such as endings, joints and central points of the curves, are usually sufficient to distinguish, for example, one letter from another during optical character recognition (OCR), or describe fingerprints or veins patters. The skeleton also helps, for example, in splitting the shape into its characteristic sections, like in the work of Xie *et al*. (2011), where it is used to distinguish body parts.

Thinning procedures originate in the 1950s, when using the averaging operation conducted by employing a square window with a high threshold produced almost a skeleton from an image (Dinneen, 1955). An extended survey of the approaches to thinning prior to the year 1991 is given by Lam *et al*. (1991). In general, thinning algorithms can be split into parallel, which produce a skeleton of the shape in one step, or sequential, which analyze border pixels of the shape in each iteration removing the ones corresponding to given rules up until the skeleton is extracted. Examples of the most widely used algorithms from the first group include the following:

- Rutovitz algorithm (Rutovitz, 1966),
- Zhang–Suen algorithm (Zhang and Suen, 1984),
- Guo–Hall algorithm (Guo and Hall, 1989),
- Chen algorithm (Chen and W.H., 1993),
- Deng algorithm (Deng *et al*., 2000),
- Prakash algorithm (Prakash *et al*., 2015),

and from the second group we can list the following:

- Arcelli–Sanniti di Baja algorithm (Arcelli and Sanniti di Baja, 1978),
- Arcelli improved algorithm (Arcelli, 1981),
- KMM algorithm (Saeed *et al*., 2001),
- Kardos algorithm (Kardos *et al*., 2009),

[*]Corresponding author

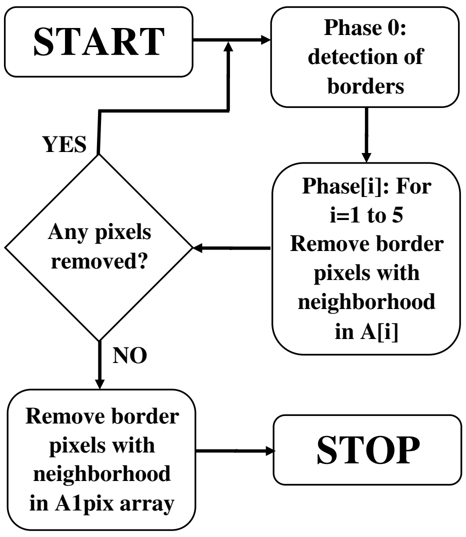

START
Phase 0: detection of borders
Phase[i]: For i=1 to 5 Remove border pixels with neighborhood in A[i]
Any pixels removed?
YES
NO
Remove border pixels with neighborhood in A1pix array
STOP

Fig. 1. Flowchart of the original K3M.

- K3M algorithm (Saeed *et al.*, 2010),
- Abu-Ain algorithm (Abu-Ain *et al.*, 2013).

The improved algorithm is planned to be used in the human posture recognition and monitoring system. The main purpose of such a system will be the monitoring of people in crowded places to detect unusual behavior.

Another application is to identify security issues and people whose strange behavior may indicate the intention of committing a crime. This may work in conjunction with other means of people identification which can be applied using a security camera, for example, face recognition which is also in the spectrum of interests of the authors (Misztal *et al.*, 2013). On the other hand, human body posture recognition might also be used in monitoring babies during their sleep.

The paper consists of four main sections. The next one contains the description of the original K3M approach and its drawbacks. Afterwards, the proposed changes are described, followed by the results of experiments and a comparison of the improved approach with the original one as well as with the KMM (Saeed *et al.*, 2001) and Zhang–Suen (Zhang and Suen, 1984) approaches. The conclusions are contained in the last section.

2. Original K3M algorithm

The K3M algorithm was described by Saeed *et al.* (2010). Its basic flowchart is presented in Fig. 1.

K3M is a sequential iterative algorithm. Each iteration involves six phases, repeated in sequence until no modification to the image can be made. Another phase is added to produce a one-pixel-width skeleton. The input to the algorithm is a binary image, where the background pixels are encoded as 0s whilst the image (object) pixels are encoded as 1s. Thinning decisions are made for border pixels (those image pixels that stick to the background), based on 3×3 neighborhood templates of the tested pixel. The thinning decision is either to delete the pixel or to retain it. The possible neighborhood configurations are encoded using values named neighbor weights. The weight of the neighborhood of the pixel $W(x, y)$ is calculated using the mask presented in Fig. 2.

The weights of the pixels that must be removed in each phase are stored in lookup arrays A0–A5 and A1pix for an additional phase:

- A0: 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,
- A1: 7, 14, 28, 56, 112, 131, 193, 224,
- A2: 7, 14, 15, 28, 30, 56, 60, 112, 120, 131, 135, 193, 195, 224, 225, 240,
- A3: 7, 14, 15, 28, 30, 31, 56, 60, 62, 112, 120, 124, 131, 135, 143, 193, 195, 199, 224, 225, 227, 240, 241, 248,
- A4: 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,
- A5: 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,
- A1pix: 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.

K3M is fast, robust and reliable, as proven by the authors (Saeed *et al.*, 2010).

2.1. Drawbacks of the original approach. The original algorithm was developed mainly for preprocessing binary images for biometric purposes such

128	1	2
64	X	4
32	16	8

Fig. 2. Mask used to calculate the weight of a pixel.

as optical character recognition (OCR) or acquisition of a fingerprint or a vein pattern. One of the main assumptions of the algorithm was to maintain skeleton continuity, which was successfully achieved. As was proved in the authors' former publications, pixels that could lead to a disruption of the skeleton are not removed. The other assumptions of the algorithm were simplicity and short computing time. The fast processing speed was acquired with some compromises in the skeleton shape which are typically unnoticeable during the thinning of already relatively thin objects such as letters.

The main problem arises when considering more complicated shapes than those of the standard type. As a sequential algorithm, K3M suffers from the order-dependency problem—different visiting orders of the detected border pixels may yield different skeletons. This was examined by rotating input images—the rotation is the equivalence of changing the visiting order. The drawbacks of K3M manifest themselves mainly during the thinning of wider objects. This refers to the following:

1. *Leftover spurs in discoid objects*—this may appear occasionally on various sides of the skeleton depending on the actual shape of the disc. The spurs may or may not appear during the processing of the same object, depending on its rotation, as presented in Fig. 3.

2. *Leftover diagonal line in the bottom right corner of the skeleton*—these are the artifacts generated due to the way in which K3M analyzes the image: from the top left corner of it to the bottom right. This means that, if the corner pixel is with the weight of 193 (bottom-right corner), the neighboring pixels are typically analyzed and removed earlier during Phase 2, leaving the pixel with the actual weight of 128 when its analysis starts. This value is not included in any neighborhood lookup array and hence the pixel is preserved, and with the cumulative effect in every next step the artifact is generated. Examples of such artifacts are presented in Fig. 4. It is noticeable that, depending on the rotation angle the artifact is generated in a different corner of the original shape. Also, it is visible that using the original A1pix array produces the result with 2-pixel-wide diagonal lines. The mechanism of diagonal lines occurrence is described in Appendix A.

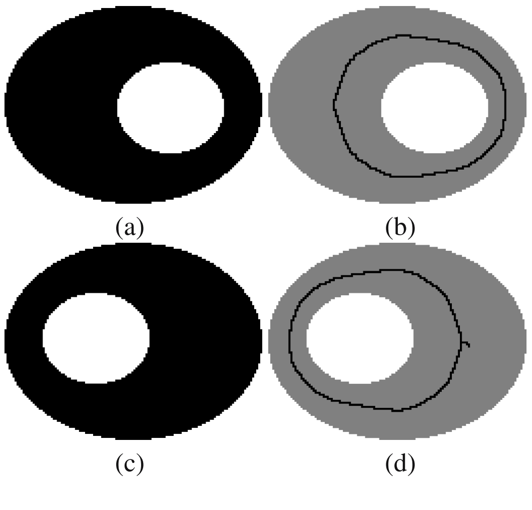


Fig. 3. Example of spur artifact: original figure (a), skeleton of the original image (b), image rotated by 180° (c), skeleton of the rotated image containing a spur artifact (d).

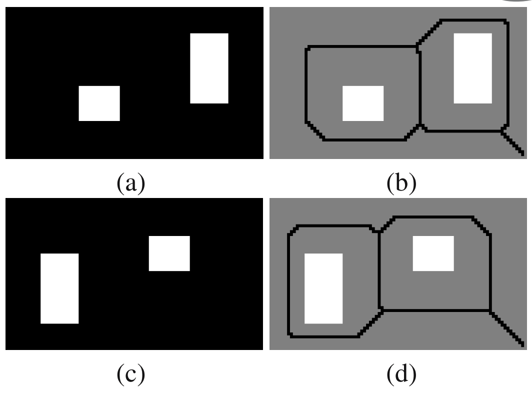


Fig. 4. Example of diagonal line artifact: original figure (a), skeleton of the original image (b), image rotated by 180° (c), skeleton of the rotated image (d).

3. *Different handling of circular objects depending on their symmetry*—it is hard to obtain an ideal circle during the preprocessing of a noisy image, and it is only for the ideal one that the K3M algorithm generates the same skeleton regardless of the angle of rotation (Fig. 5). It can also be noticed that K3M produces a cross-shaped skeleton from a ideal circle rather than a dot-shaped one, and it produces some additional horizontal and vertical lines in the other two shapes.

4. *Different skeleton shapes for the same object if it is rotated by different angles*—this is clearly noticeable in figures describing previous problems.

The proposed modifications address all aforementioned drawbacks.

3. Proposed modifications

An overall schematic of the modified solution has only one change relative to the original solution (Fig. 1), as presented in Fig. 6. The additional Phase 0a is executed after Phase 0.

The altered algorithm is split into seven phases inside the main loop and one phase outside the loop (Fig. 6).

3.1. Modification of neighborhood lookup arrays. From the original arrays only A2 was expanded (the added values are in bold) and array A1pix was replaced completely:

- A0: 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,
- A1: 7, 14, 28, 56, 112, 131, 193, 224,
- A2: 7, 14, 15, **24**, 28, 30, **48**, 56, 60, 112, 120, 131, 135, **192**, 193, 195, 224, 225, 240,
- A3: 7, 14, 15, 28, 30, 31, 56, 60, 62, 112, 120, 124, 131, 135, 143, 193, 195, 199, 224, 225, 227, 240, 241, 248,
- A4: 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,
- A5: 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,
- A1pix: 2, 5, 13, 20, 21, 22, 32, 48, 52, 54, 65, 67, 69, 80, 81, 84, 88, 97, 99, 128, 133, 141, 208, 216.

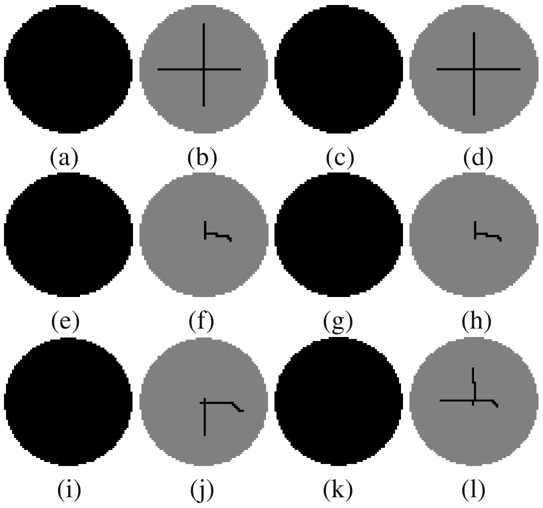


Fig. 5. Example of different circle handling: ideal circle (a), skeleton of this circle (b), 180° rotation of this circle (c), skeleton of the rotated image (d), slightly wider than higher circle (e), skeleton of this circle (f), 180° rotation of this circle (g), skeleton of the rotated image (h), asymmetrical circle (i), skeleton of this circle (j), 180° rotation of this circle (k), skeleton of the rotated image (l).

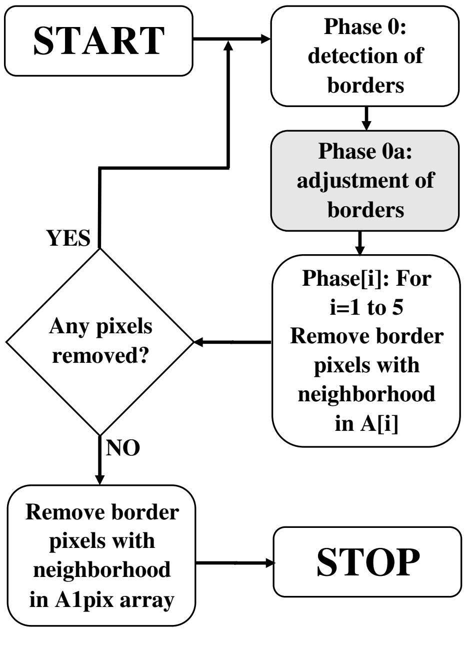

START
Phase 0: detection of borders
Phase 0a: adjustment of borders
Phase[i]: For i=1 to 5 Remove border pixels with neighborhood in A[i]
Any pixels removed?
YES
NO
Remove border pixels with neighborhood in A1pix array
STOP

Fig. 6. Flowchart of the modified K3M. The additional phase is highlighted with a gray background.

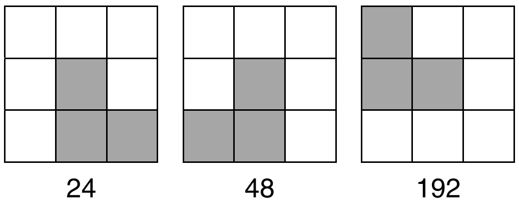


Fig. 7. Additional mask values in array A2.

The masks represented by the values added to A2 are presented in Fig. 7.

These masks were added to clear the shape of some leftover spurs during Phase 2 and as a part of the solution for the problem of the diagonal line artifacts in the original K3M (see Fig. 4).

The change in the A1pix array is proposed as a way to produce one-pixel-wide diagonal lines, as opposed to the original array which produced two-pixel-wide ones (see Fig. 4).

3.2. Modification of Phase 0. The flowchart of modified Phase 0 is presented in Fig. 8.

The main addition in this phase is the comparison of the pixel weight to four additional masks if its weight is not in the border neighborhood lookup array A0. These masks are presented in Fig. 9. The condition C1 is $(W(x,y) == 95$ *and* $(x-2,y) ==$ *white) or* $(W(x,y) == 125$ *and* $(x,y-2) ==$ *white) or* $(W(x,y) == 215$ *and* $(x,y+2) ==$ *white) or* $(W(x,y) == 245$ *and* $(x+2,y) ==$ *white)*. If it is fulfilled, the pixel is also marked as a border one. This

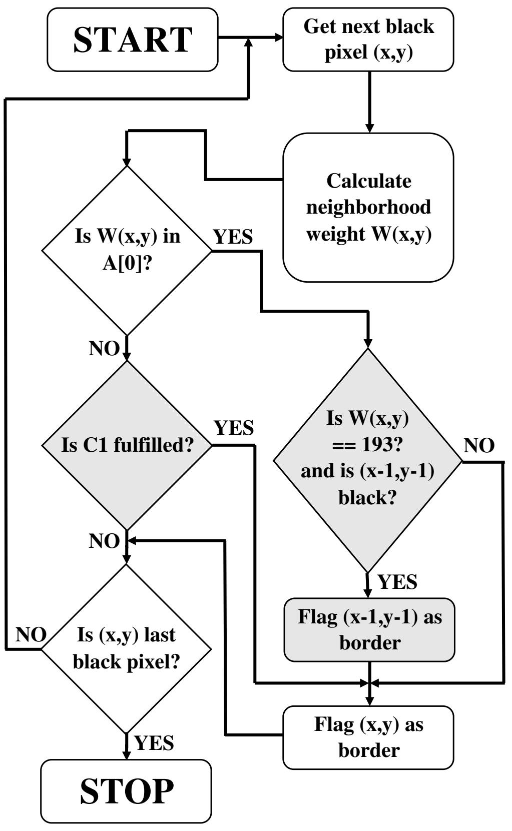

START
Get next black pixel (x,y)
Calculate neighborhood weight W(x,y)
Is W(x,y) in A[0]?
YES
NO
Is C1 fulfilled?
YES
NO
Is W(x,y) == 193? and is (x-1,y-1) black?
NO
YES
Flag (x-1,y-1) as border
Is (x,y) last black pixel?
NO
YES
Flag (x,y) as border
STOP

Fig. 8. Flowchart of Phase 0.

solution addresses the issue of circle handling (Fig. 5) and disks (Fig. 3) by the original K3M.

The second addition to Phase 0 is the marking of the top-left neighboring pixel of a bottom-right corner pixel (the weight value of 193; see Fig. 10) as a border pixel. This change is made to help deal with the diagonal line artifacts (see Fig. 4).

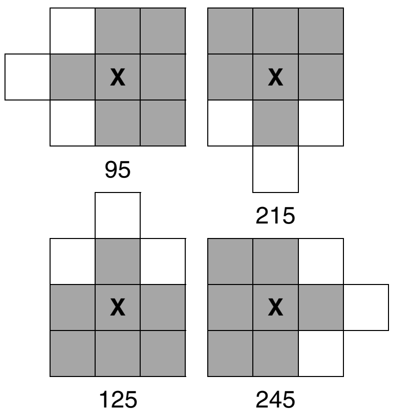


Fig. 9. Additional masks for Phase 0.

3.3. Phase 0a. Phase 0a supplements the changes introduced in Phase 0 to ensure that the discoid and circular objects will not have any unwanted spurs (see Figs. 3 and 5). The flowchart of the new phase is presented in Fig. 11.

The weight of every regular black (non-border) pixel is calculated again using the mask presented in Fig. 2, although in the calculation there are included only the neighboring border pixels found during Phase 0 and already found in Phase 0a, not all surrounding black pixels. This weight is compared to two new masks (Fig. 12) and, if the weight is equal to any of them, the pixel is also marked as border.

3.4. Modification of Phases 1–5. The flowchart of the modified Phases 1 to 5 is presented in Fig. 13.

Condition C2: *if* $W(x, y) == 241$ *and* $(x, y + 1)$ *is border and* $(x, y + 2) ==$ *white and* $(x + 1, y + 2) ==$ *white*, presented in Fig. 13, is added to address the issue of diagonal line artifacts presented in Fig. 5. It determines whether the bottom neighbor of the analyzed pixel might be a corner. If the condition is fulfilled, the bottom neighbor is analyzed for deletion before removing the currently analyzed pixel (Fig. 14). This helps avoid the situation described in Appendix A.

Condition C3: *if* $(W(x, y) == 195$ *or* $W(x, y) == 227)$ *and* $(x+1, y-1)$ *is border* (Fig. 15) is introduced to remove asymmetrical horizontal lines in circular shapes. The extra horizontal lines in the skeleton are created in a similar way as the diagonal ones—due to the fact that further columns of the image are analyzed after the removal of some pixels from an earlier column.

If these pixels are analyzed before the removal of essential neighbors, they are removed correctly. That is why during the analysis of pixel (x, y), if its weight is equal to 195 or 227, its top-right neighbor $(x - 1, y + 1)$ is also analyzed. On the other hand, this neighbor shall not be removed too early, because it would alter the calculation of the weight of top neighbor of $(x - 1, y + 1)$ pixel. That is why this pixel is only marked for deletion first, and then removed only when it becomes the analyzed pixel (x, y). These pixels are presented in Fig. 5.

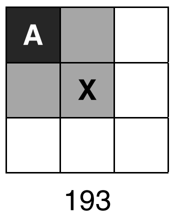


Fig. 10. Pixel that fulfills the mask 193 (X) and its neighbor which shall be marked as border (A).

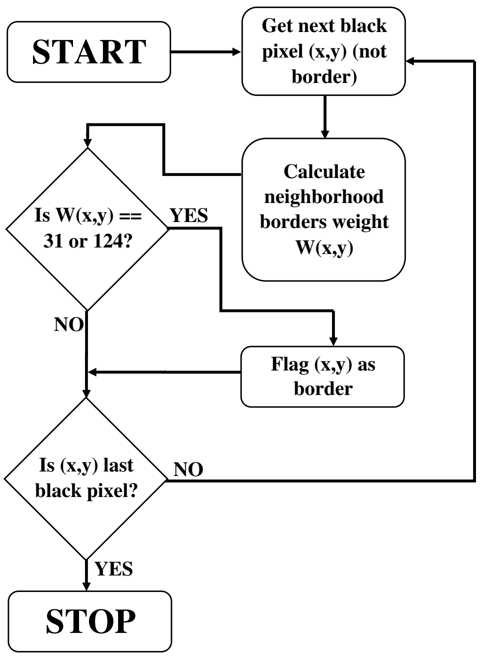

START
Get next black pixel (x,y) (not border)
Calculate neighborhood borders weight W(x,y)
Is W(x,y) == 31 or 124?
YES
NO
Flag (x,y) as border
Is (x,y) last black pixel?
NO
YES
STOP

Fig. 11. Flowchart of Phase 0a.

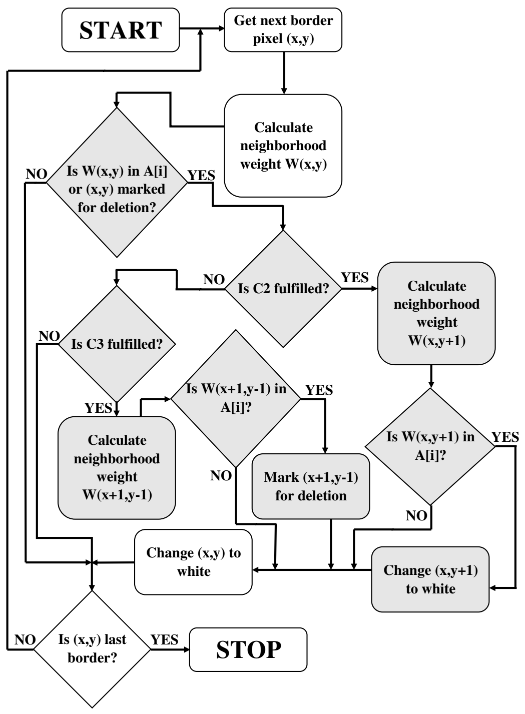

START
Get next border pixel (x,y)
Calculate neighborhood weight W(x,y)
Is W(x,y) in A[i] or (x,y) marked for deletion?
NO
YES
Is C2 fulfilled?
NO
YES
Calculate neighborhood weight W(x,y+1)
Is C3 fulfilled?
NO
YES
Is W(x+1,y-1) in A[i]?
YES
NO
Calculate neighborhood weight W(x+1,y-1)
Mark (x+1,y-1) for deletion
Is W(x,y+1) in A[i]?
YES
NO
Change (x,y) to white
Change (x,y+1) to white
Is (x,y) last border?
NO
YES
STOP

Fig. 13. Flowchart of Phase $i, i = 1, \ldots, 5$.

4. Results

Three experiments were conducted using the original and improved K3M solutions as well as the KMM (Saeed *et al.*, 2001) and Zhang–Suen (Zhang and Suen, 1984) approaches as reference algorithms. These experiments include:

- testing the algorithms on selected basic shapes—mainly the ones troublesome for the original K3M,
- checking the computing times of all algorithms,
- testing the algorithms on images presenting the human body posture.

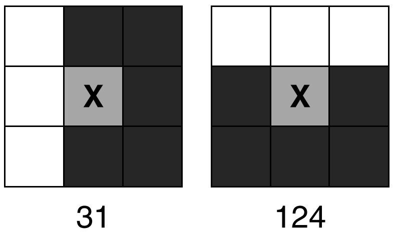


Fig. 12. Two new masks introduced for Phase 0a. In these masks, the light gray color represents a black pixel, the dark gray color represents pixels that shall be border ones and the white color represents pixels that shall not be border ones (they might be regular black or white pixels).

4.1. Thinning of basic images. To test the improvements over the original K3M approach, a set of basic shapes including all those described in Section 2.1 was prepared. The images were created by the authors, specifically for this publication, to reveal and emphasize the problems that may arise in skeletonization. The images were processed using both improved and original approach as well as the KMM and Zhang–Suen algorithms, as mentioned earlier. The resulting skeletons are presented in Table 1. Many of the shapes were handled differently by the algorithms. A brief summary of the results is listed below.

- Shapes 1, 2 and 3 were handled properly by the improved K3M, KMM and Zhang–Suen approaches, while K3M produced diagonal line artifacts.

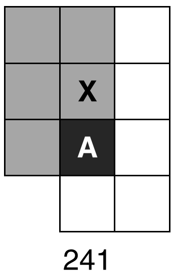


Fig. 14. Situation in which a bottom neighbor (A) of the analyzed pixel (X) will be considered a corner.

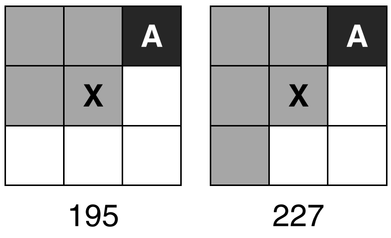


Fig. 15. Situations in which the top-right neighbor (A) of the analyzed pixel (X) will also be considered for removal.

- Shape 4 was handled properly by both the K3M and KMM approaches while the Zhang–Suen approach omitted one vertex of the triangle.
- Shape 5, the diamond, was handled properly by the original K3M while KMM omitted one vertex and the improved K3M and Zhang–Suen approaches omitted two vertexes.
- Shapes 6–8 are different circles, 7 and 8 are symmetrical and 6 is asymmetrical. The improved K3M handled all of them properly while KMM produced a correct skeleton approach only from 6 and 7, and Zhang–Suen only from 8, leaving no pixels in the other ones. The original K3M treated all the shapes differently, although arguably only 8 could be considered as proper.
- Shape 9 was handled properly by both K3M approaches, although the original K3M produced a short diagonal spur in the bottom of the skeleton and the there-pixel neighborhood in diagonal lines. Both the K3M and Zhang–Suen approaches are missing one side of the arrow.
- Shape 10 was handled properly by all approaches except KMM.
- Shape 11 was handled properly only by the original K3M, while the result of KMM is missing one side of an arrow, and the improved K3M and Zhang–Suen approaches produced straight line skeletons.
- Shape 12 was handled properly by all algorithms, although the original K3M shortened the top part of the diagonal line.
- Shape 13 was handled properly by all algorithms.
- Shape 14 was handled properly by both K3M and KMM. The Zhang–Suen approach omitted one vertex of the shape.

As can be seen, in most cases the modified K3M algorithm gives better results than the original K3M and the other tested approaches. It is the only approach which handled all the circles properly.

Table 1. Comparison of the results of thinning between the improved K3M and the original K3M, KMM and Zhang–Suen algorithms. The black lines are the skeletons of the objects and the gray silhouettes are the original shapes. The column *No.* contains the image number.

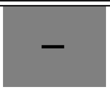
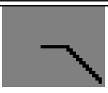
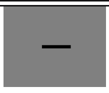
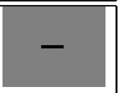
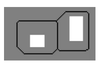
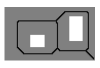
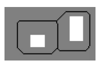
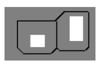
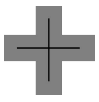
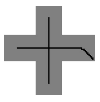
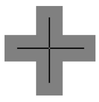
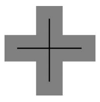
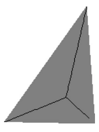
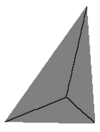
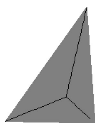
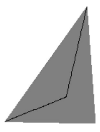
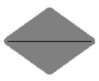
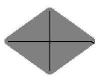
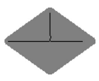
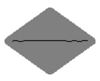
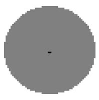
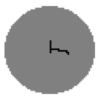
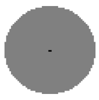
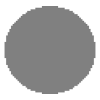
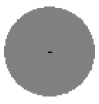
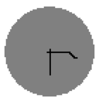
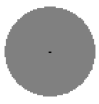
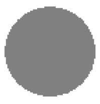
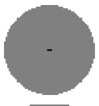
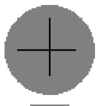
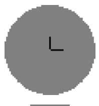
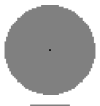
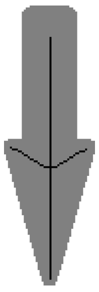
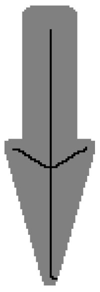
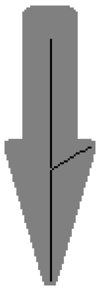
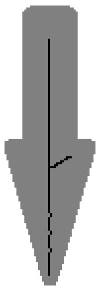
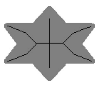
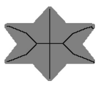
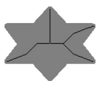
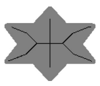
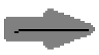
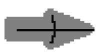
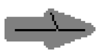
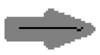
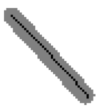
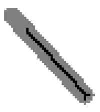
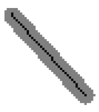
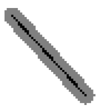
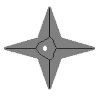
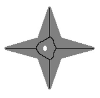
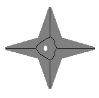
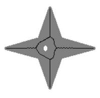
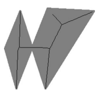
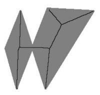
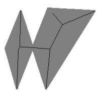
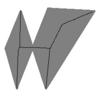
Sample	Improved K3M	Original K3M	KMM	Z–S
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				

Table 2. Comparison of computing time of thinning for the images presented in Table 1 using the improved K3M and the original K3M, KMM and Zhang–Suen algorithms. The *Sample* column gives information about the image number, *Res.* shows the image resolution in pixels, and the remaining columns show the average computing time for particular algorithms in milliseconds.

Sample	Res.	Improved K3M	Original K3M	KMM	Z–S
1	32x25	6	<1	<1	<1
2	88x52	46	15	15	<1
3	72x71	31	12	9	<1
4	127x164	265	78	78	16
5	107x78	78	31	24	<1
6	56x54	31	15	13	<1
7	64x64	47	16	15	<1
8	57x57	31	15	15	<1
9	46x143	47	16	15	<1
10	127x104	156	47	47	15
11	44x20	<1	<1	<1	<1
12	41x44	<1	<1	<1	<1
13	151x138	94	31	15	31
14	216x196	562	172	172	31

4.2. Computing times.

All the test shapes were processed 51 times by the original and the improved K3M, KMM as well as the Zhang–Suen approaches. Processing times from these trials were sorted in ascending order and 11 median values were averaged. These results are presented in Table 2.

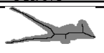
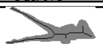
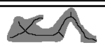
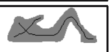
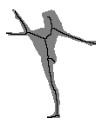
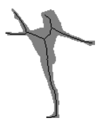
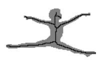
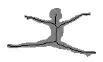
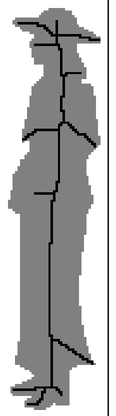
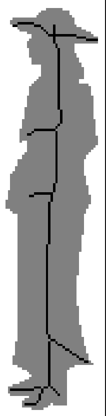
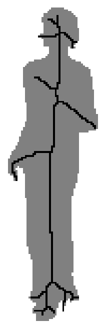
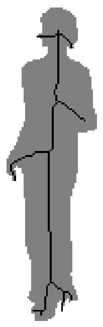
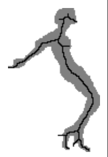
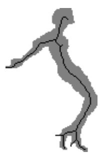
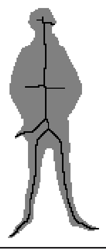
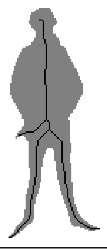
It is worth noticing that pictures 11 and 12 are very small, which accounts for the low execution time, below the timer resolution. The analysis was conducted on an Intel Core i7-4790 3.6 GHz CPU with 4 GB of RAM, running Microsoft Windows 8.1, and all the implementations were single-threaded, written in Java.

All the algorithms processed each of the images in less than a second, in some cases even in less than a millisecond. The Zhang–Suen approach, due to its parallel character, seems to be the fastest algorithm amongst the tested ones. The other approaches are somehow similar in their implementation process and sequential character. Notice that the time differences shown in Table 2 are certainly higher for larger images.

4.3. Thinning of human silhouettes.

The last experiment was conducted to check whether the improvements of the original K3M have any impact on the thinning of sample human silhouettes. Overall, the tests were performed on 79 samples; the eight exemplary results are presented in Table 3.

Table 3. Comparison of the results of thinning of human silhouettes using the original and improved K3M approaches. The column *Sample* contains image numbers, the column *O. K3M* contains images processed using the original K3M approach and the columns *I. K3M* contain images processed using the improved K3M approach.

Sample	Original K3M	Improved K3M	Sample	Original K3M	Improved K3M
1			2		
3			4		
5			6		
7			8		

As seen in the sample images, the improved K3M leaves fewer unwanted artifacts than the original approach, although the skeletons of the silhouettes are still im perfect and need some extra denoising before further analysis.

5. Conclusions

The first experiment shows that modifications of the K3M algorithm eliminated all of the drawbacks of the original approach described in Section 2.1. The improved K3M generally gives better thinning results than the original K3M, KMM or other known algorithms, like the Zhang–Suen approach.

As shown in the second experiment, better results come at a cost—the computing time increases, though it is still fast enough to process small images in real time. The cost is partially due to the fact that the improved K3M was implemented on the basis of an optimal K3M implementation which includes some solutions that accommodate well for the latter but hamper the optimal implementation of the former. The authors are currently concentrating on the improvement of the quality of the source code of the proposed solution.

One of the important aspects is reduction in the number of phases without any loss of skeleton quality. A five-scan procedure is more time-consuming than one. However, it is worth noting that, in our algorithm, the computing time does not scale linearly, and hence five phases do not take five times more time than one. This is because the most time-consuming stage—looking for border pixels—happens only once per iteration step. Therefore, each of the five phases takes into account only a small subset of original image pixels. Moreover, there are fewer pixels to analyze after each phase as some of them are removed. In an optimal implementation, border pixels are probably buffered, so there is no need to search for them in each stage.

The third experiment was conducted to prove the suitability of the improved solution for a human posture recognition system. The results are promising, although still reveal some problems. Such problems are solved by using the black pixel distance from the nearest white pixel in the initial image preprocessing stage. This allows us to narrow the number of pixels left for thinning by determining which pixels are unnecessary for proper skeleton acquisition. This procedure is explained in Appendix B. The initial experiments of the authors proved that this approach may provide even more regularity and repeatability of similar shape skeletons.

Conclusively, the most important features of thinning algorithms are connectivity and shape preservation. The latter is particularly important when thinning wide shapes, and, as revealed by the experiments, the proposed modification performs significantly better than the original K3M or any other tested algorithm in the paper. By connectivity (or continuity) we mean topology preservation—the goal of the thinning algorithm is to reduce the image (understood as a set of black pixels) in such a way that each 8-connected black component in the original picture contains exactly one 8-connected black component of the produced picture whilst each 4-connected white component in the output picture contains exactly one 4-connected white component of the input picture. In other words, a topology preserving thinning algorithm does not split, nor does it delete completely, the 8-connected black components, and it does not merge nor creates 4-connected white components (Kong and Rosenfeld, 1989). This is guaranteed by the selection of pixels for deletion.

Pixel weights, introduced in the deletion arrays, represent specific configurations of the tested pixel neighbors. There is a finite number of such configurations, and they have been carefully selected to ensure that no pixel will be removed if it causes skeleton disruption. The weights representing configurations where a tested pixel is the only pixel connecting its neighbors are never included in any deletion array. In Steps 1–5 of the algorithm we consider only pixels whose neighbors are interconnected, and hence there is no possibility that their deletion can cause a loss in skeleton continuity. For the final step, the deletion array was created with only one principle in mind—to never remove a pixel if it disrupts skeleton continuity. The sequential nature of these steps ensures that the pixels are deleted in accordance with a defined sequence. This takes place in such a way that when testing any pixel, there is previous knowledge about the decisions made for prior pixels, and we always make our decision before testing the following pixels, although this comes at some additional cost (an order-dependency problem).

As the decision made for previous pixels has impact on the decision made for the tested one, the visiting order is significant and can always affect the results. There are two consequences of this: first, it is important to keep a proper sequence of iteration; deletion arrays were selected for the specific iteration direction, and will not work for others (for example, some lines may be completely deleted). Second, a rotation of the thinned shape affects results. This is because, as a matter of fact, the rotation means changing the order in which we iterate through pixels. In effect, when thinning rotated shapes, we can obtain different output images, as was described in Sections 2.1 and 4.1. The authors now work on a new algorithm that combines the topology preservation features of K3M+ with the advantages of parallel algorithms.

Acknowledgment

This work was supported by the grants S/WI/1/2013 and S/WI/2/2013 from the Białystok University of Technology and funded with resources for research by the Ministry of Science and Higher Education in Poland.

References

Abu-Ain, W., Abdullah, S.N.H.S., Bataineh, B., Abu-Ain, T. and Omar, K. (2013). Skeletonization algorithm for binary images, *Procedia Technology* **11**(0): 704–709.

Arcelli, C. (1981). Pattern thinning by contour tracing, *Computer Graphics and Image Processing* **17**(2): 130–144, DOI: 10.1016/0146-664X(81)90021-6.

Arcelli, C. and Sanniti di Baja, G. (1978). On the sequential approach to medial line transformation, *IEEE Transactions on Systems, Man and Cybernetics* **8**(2): 139–144, DOI: 10.1109/TSMC.1978.4309914.

Chen, Y. and W.H., H. (1993). Parallel thinning algorithm for binary digital patterns, *in* C. Chen *et al.* (Eds.), *Handbook of Pattern Recognition; Computer Vision*, World Scientific Publishing, River Edge, NJ, pp. 457–490, DOI: 10.1007/978-3-642-33564-8_78.

Deng, W., Iyengar, S.S. and Brener, N.E. (2000). A fast parallel thinning algorithm for the binary image skeletonization, *International Journal of High Performance Computing Applications* **14**(1): 65–81.

Dinneen, G. (1955). Programming pattern recognition, *Proceedings of the 1955, Western Joint Computer Conference, New York, NY, USA*, pp. 94–100, DOI: 10.1145/1455292.1455311.

Guo, Z. and Hall, R. (1989). Parallel thinning with two-subiteration algorithms, *Communications of the ACM* **32**(3): 359–373, DOI: 10.1145/62065.62074.

Kardos, P., Nemeth, G. and Palagyi, K. (2009). An order independent sequential thinning algorithm, *in* P. Wiederhold and R. Barneva (Eds.), *Combinatorial Image Analysis,* Lecture Notes in Computer Science, Vol. 5852, Springer, Berlin/Heidelberg, pp. 162–175, DOI: 10.1007/978-3-642-10210-3_13.

Kong, T.Y. and Rosenfeld, A. (1989). Digital topology: Introduction and survey, *Computer Vision, Graphics, and Image Processing* **48**(3): 357–393, DOI: 10.1016/0734-189X(89)90147-3.

Lam, L., Lee, S. and Sueni, C. (1991). Thinning methodologies—a comprehensive survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(9): 869–885, DOI: 10.1109/34.161346.

Misztal, K., Szczepański, A., Kocjan, P., Saeed, K. and Tabor, J. (2013). Distribution estimation applied to face recognition as a simple and robust solution, *2013 International Conference on Biometrics and Kansei Engineering (ICBAKE), Tokyo, Japan,* DOI: 10.1109/ICBAKE.2013.19.

Prakash, R., Prakash, K.S. and Binu, V. (2015). Thinning algorithm using hypergraph based morphological operators, *2015 IEEE International Advance Computing Conference (IACC), Benglore, India,* pp. 1026–1029.

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

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, Berlin/Heidelberg, pp. 601–609.

Saeed, K., Tabędzki, M., Rybnik, M. and Adamski, M. (2010). K3M: A universal algorithm for image skeletonization and a review of thinning techniques, *International Journal of Applied Mathematics and Computer Science* **20**(2): 317–335, DOI: 10.2478/v10006-010-0024-4.

Xie, F., Xu, G., Cheng, Y. and Tian, Y. (2011). Human body and posture recognition system based on an improved thinning algorithm, *IET Image Processing* **5**(5): 420–428, DOI: 10.1049/iet-ipr.2009.0303.

Zhang, T. and Suen, C. (1984). A fast parallel algorithm for thinning digital patterns, *Communications of the ACM* **27**(3): 236–239.

Marek Tabedzki received an M.Sc. degree in 2001 and a Ph.D. degree in 2009, both in computer science from the Białystok University of Technology (Poland). He is presently with the same university, at the Faculty of Computer Science. His research interests include information processing systems, particularly digital analysis image processing, pattern recognition and biometrics.

Khalid Saeed received a B.Sc. degree in electrical and electronics engineering in 1976 from Baghdad University, and then M.Sc. and Ph.D. degrees from the Wrocław University of Technology in 1978 and 1981, respectively. He received his D.Sc. degree (habilitation) in computer science from the Polish Academy of Sciences in 2007. He had been a visiting professor of computer science with the Białystok University of Technology, where he is now working as a full professor. He was a professor of computer science at the AGH University of Science and Technology in the years 2008–2014. He also works at the Faculty of Mathematics and Information Sciences of the Warsaw University of Technology. His areas of interest are biometrics, image analysis and processing, and computer information systems.

Adam Szczepański received his M.Sc. degree in computer science from the Białystok Technical University (Poland) in 2008. He is a former employee of the Faculty of Physics and Applied Computer Science of the AGH University of Science and Technology in Kraków, Poland. Currently he works as a software developer in Eniro Group Poland. His research interests include digital image and signal processing, and pattern matching and recognition. He is registered as a Ph.D. student with the Faculty of Computer Science at the Białystok University of Technology under the supervision of Prof. Khalid Saeed.

Appendix A
Genesis of diagonal line artifacts

To present the genesis of diagonal line artifacts, a square figure will be used (Fig. A1). This is due to the fact that this example has four corners and thus is a good demonstrator of how the top-left, bottom-left and top-right corners are removed and the bottom-right corner is kept in the skeleton. Figure A1 contains also a description explaining how regular, border and removed black pixels will be marked in all the figures in this appendix.

It is important to remember that the K3M algorithm analyzes the image column-by-column from the top-left corner of the image to the bottom-right one, so the analysis of the shape in Fig. A1 is performed starting from pixel 1, then going to 2 and ending with pixel 16.

The decisions for pixels in Phase 0 are as follows:

- Pixel 1 – Weight = 28: is border,
- Pixel 2 – Weight = 31: is border,

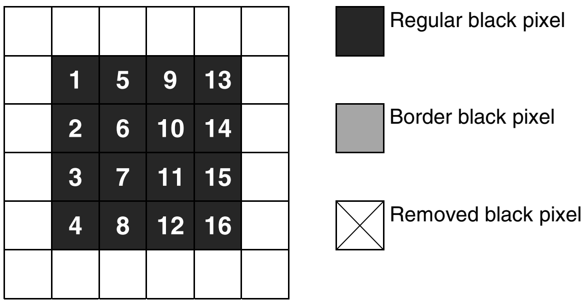


Fig. A1. Sample square.

- Pixel 3 – Weight = 31: is border,
- Pixel 4 – Weight = 7: is border,
- Pixel 5 – Weight = 124: is border,
- Pixel 6 – Weight = 255: is not border,
- Pixel 7 – Weight = 255: is not border,
- Pixel 8 – Weight = 199: is border,
- Pixel 9 – Weight = 124: is border,
- Pixel 10 – Weight = 255: is not border,
- Pixel 11 – Weight = 255: is not border,
- Pixel 12 – Weight = 199: is border,
- Pixel 13 – Weight = 112: is border,
- Pixel 14 – Weight = 241: is border,
- Pixel 15 – Weight = 241: is border,
- Pixel 16 – Weight = 193: is border.

The results of Phase 0 are presented in Fig. A2.

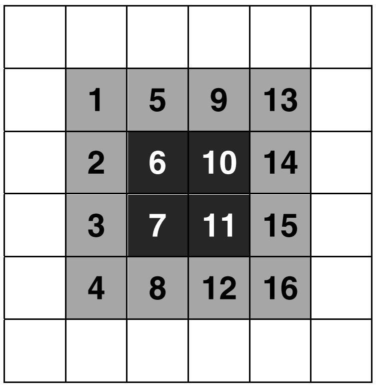


Fig. A2. Result of Phase 0 on the sample shape.

The decisions for pixels in Phase 1 are as follows:

- Pixel 1 – Weight = 28: changed to white,
- Pixel 2 – Weight = 30: no change,
- Pixel 3 – Weight = 30: no change,
- Pixel 4 – Weight = 7: changed to white,
- Pixel 5 – Weight = 60: no change,
- Pixel 8 – Weight = 135: no change,
- Pixel 9 – Weight = 124: no change,
- Pixel 12 – Weight = 199: no change,
- Pixel 13 – Weight = 112: changed to white,
- Pixel 14 – Weight = 240: no change,
- Pixel 15 – Weight = 241: no change,
- Pixel 16 – Weight = 193: no change.

The results of Phase 1 are presented in Fig. A3.

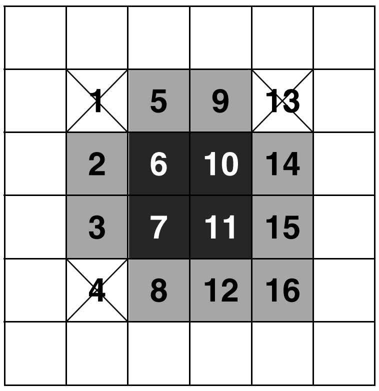


Fig. A3. Result of Phase 1 on the sample shape.

The decisions for pixels in Phase 2 are as follows:

- Pixel 2 – Weight = 30: changed to white,
- Pixel 3 – Weight = 14: changed to white,
- Pixel 5 – Weight = 28: changed to white,
- Pixel 8 – Weight = 7: changed to white,
- Pixel 9 – Weight = 56: changed to white,
- Pixel 12 – Weight = 135: changed to white,
- Pixel 14 – Weight = 112: changed to white,
- Pixel 15 – Weight = 208: no change,
- Pixel 16 – Weight = 129: no change.

The results of Phase 2 are presented in Fig. A4.

After Phase 2 there are no changes in the shape in Phases 3 to 5, and at the beginning of the next Phase 0 the shape is as presented in Fig. A5. Pixels 15 and 16 are the beginning of the diagonal line artifact and would not be changed during a further analysis, even if the original shape were bigger.

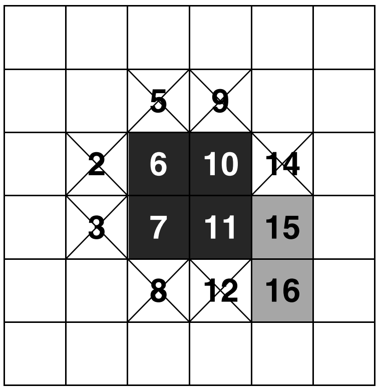


Fig. A4. Result of Phase 2 on the sample shape.

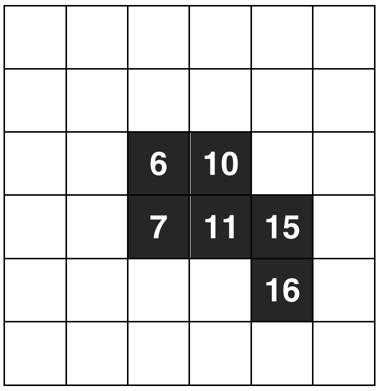


Fig. A5. Shape at the beginning of the next Phase 0. The diagonal line artifact is clearly visible.

Appendix B

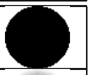
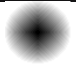
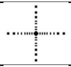
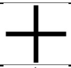
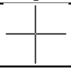
Determining the pixels necessary for the thinning of an object

This appendix contains a very brief description of the authors' current work to improve the algorithm further. The idea consists of 5 steps:

1. calculation of the distances of each black pixel from nearest white pixel,
2. calculation of the weights of each black pixel depending on the distances of its black neighbors,
3. determination of the leaving or removing of each black pixel based on of its distance and weight,
4. dilatation of the remaining black pixels,
5. thinning using the improved K3M algorithm.

The results of these steps are presented in Table B1. It is noticeable that with this approach the circular shape is thinned differently than using regular algorithms—not to a dot, but to a cross skeleton.

Table B1. Graphical representation of the proposed processing steps.

Description	Image
Original image	
Gray-scale representation of the distances of black pixels from the nearest white pixel	
Pixels left after the third step	
Image after dilatation	
Final image after thinning	

Received: 4 April 2015
Revised: 27 August 2015
Re-revised: 28 October 2015
Accepted: 2 December 2015