A New Thinning Algorithm for Binary Images

Lynda Ben Boudaoud
LIMED Laboratory
University of Bejaia, Algeria
Email: lynda.benboudaoud@gmail.com

Abderrahmane Sider
LIMED Laboratory
University of Bejaia, Algeria
Email: ar.sider@univ-bejaia.dz

Abdelkamel Tari
LIMED Laboratory
University of Bejaia, Algeria
Email: tarikamel59@gmail.com

***Abstract*—Thinning plays a crucial role in image analysis and pattern recognition applications. It is one of the most frequently used pre-processing methods to analyze different types of images. Thinning consists basically of reducing a thick digital object into a thin skeleton. There are several thinning algorithms for getting a skeleton of a binary image in the literature. The most popular, and well proved one is the ZS algorithm proposed by Zheng and Suen. In the present paper, we propose a new thinning algorithm which combines the directional approach used by ZS and the subfield approach in order to produce a new hybrid thinning algorithm which is more efficient, produces thinner results (skeleton thickness is equal to one) than the ZS algorithm and solves the ZS's loss of connectivity problem in 2×2 squares.**

Results of applying the proposed algorithm on a variety of binary images and comparison with ZS algorithm show better results in terms of thinning rate, thinning speed, visual quality and connectivity preservation.

***Keywords*—Thinning algorithm; Binary images; Skeleton; Parallel thinning; Iterative thinning; Thinning rate; Thinning speed.**

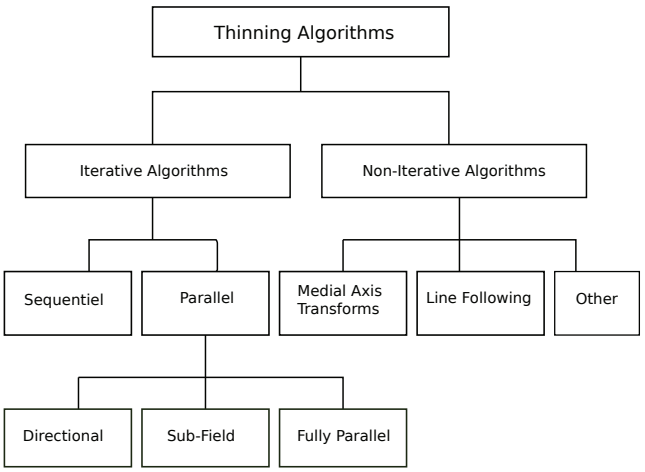

Fig. 1: Classification of Common Thinning Algorithms.

I. Introduction

In image processing, image thinning is a fundamental pre-processing step that plays a vital role in many applications such as pattern recognition, finger print classification, medical applications etc. The thinning process transforms an input binary image into a skeleton by reducing the original image which contains different thicknesses to a thin representation (a set of curves and lines). The notion of the skeleton was originally defined by Blum in 1962 [1] through an analogy with grass-fire. If we imagine an object as a grass field and we set on fire on the border of the field, the fire will start propagating inside the field and when fire fronts meet, they vanish. The meeting points of the flame fronts would constitute the skeleton of the object.

In practice, there is a need for thinning images for one or all of the following reasons [2]:

- Thinning reduces the amount of data required to be processed or transmitted over high latency networks.
- Thinning reduces the time required to process the pattern.
- Shape analysis can be more easily made on thinned patterns.

There is a large amount of literature on thinning algorithms [3], [4], [5], [6], [7], [8], [9]. Thinning algorithms can be classified in one of two broad categories [2]: iterative thinning algorithms and non iterative thinning algorithms.

Figure Fig.1 shows the major classes of thinning algorithms:

In general, iterative thinning algorithms perform pixel by pixel operations, thus they delete successive layers on the edges of a pattern until a suitable skeleton is obtained.

The criteria for a pixel deletion is a set of rules based on the $k \times k$ neighborhood chosen around the current pixel. If the rules are met then the pixel is selected for deletion. The decision for individual pixel deletion is based on the results of the previous iterations.

According to the way they examine pixels, these algorithms can be classified as sequential or parallel [3]. In a sequential algorithm, the deletion of a pixel in the n^{th} iteration depends on all the operations that have been performed so far, i.e on the result of the $(n-1)^{th}$ as well as on the pixels already processed in the n^{th} iteration. In a parallel algorithm, the deletion of pixels in the n^{th} iteration would depend only on the result that remains after the $(n-1)^{th}$ iteration; therefore, all pixels can be examined independently in a parallel manner in each iteration.

Parallel thinning algorithms can be further divided into three categories according to the used approach. There are mainly three parallel approaches: the directional approach, the subfield approach and the fully parallel approach.

- The directional approach: it goes back to the work of Rosenfeld in 1970. It consists of breaking the thinning

step (iteration) into several sub-steps (sub-iterations) of thinning each of which based on directions or a combination of directions (North, West, East, South). The deletion conditions are changed from one sub-iteration to another, then; all simple points belonging to the same direction are removed in parallel.

- The subfield approach: it consists of breaking down the figure into sub-fields [5], not according to the edge orientation, but according to some criterion such as the parity of pixels (please see section 4) so that simple pixels belonging to the same sub-field are removed in parallel.
- The fully parallel approach: no sub-iteration or sub-field takes place : the same thinning operator (deletion criteria) is used at each iteration, using an extended $k \times k$ neighborhood, where $k > 3$.

Non-iterative thinning algorithms unlike iterative ones are not based on examining individual pixels. Some popular non-pixel based methods include medial axis transforms, distance transforms, and determination of centerlines by line following.

A good thinning algorithm should possess many properties [10]. First, the result of thinning process should preserve the connectivity (or topology), and have one pixel thick (width). Second, the shape of the pattern should be preserved, for example, an object having the shape of the letter "b" should not be transformed into an object like "o". Third, the skeleton should lie in the middle of the original shape. Fourth, excessive erosion should be prevented (the length of lines and curves should be preserved) [11]. Finally, the algorithm should be as fast as possible (high speed of processing, low execution time). More properties and requirements may be defined for specialized tasks, but the above enumerated basic proprieties are most important features which thinning methods should have. Some of these features are contradictory so it is hard to fulfill all these requirements at once and not all algorithms may satisfy all of them; a good algorithm aims at satisfying most of them. The rest of this paper is organized as follows. Section 2 gives some basic preliminary notions about binary images. In Section 3, one of the most popular, studied and used parallel thinning algorithm, namely the ZS algorithm [9] is sketched. Section 4 presents our new hybrid thinning algorithm. Section 5 first defines two useful performance measures (thinning rate, thinning speed) and then shows results of both ZS and the proposed algorithm on a broad range of test images along with a discussion for each type of image. Finally, in Section 6, we round off the paper by some concluding remarks and perspectives.

II. Preliminary notions

The binary image S is represented by a matrix P, of size $M \times N$, where $P(i,j)$ represents the binary value of the pixel (i,j). If $P(i,j) = 1$ then the pixel (i,j) is black and if $P(i,j) = 0$, it is white. The black pixels form the foreground of S, and the white ones form the background which is the complement of the foreground.

A pixel p at location (i,j) has two horizontal and two vertical neighbors as it is shown in figure Fig.2.

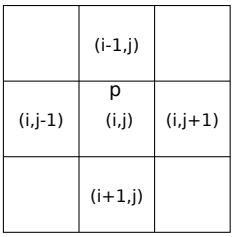

Fig. 2: The 4-Neighborhood of a Pixel p.

This set of four pixels is called 4-neighborhood of the pixel p noted $N_4(p)$. It describes on an other hand the 4-connectivity. Each of these neighbors is at a unit distance from p.

The are 4 diagonal pixels that represent the diagonal neighborhood of the pixel p noted $N_D(p)$ as shown in figure Fig.3.

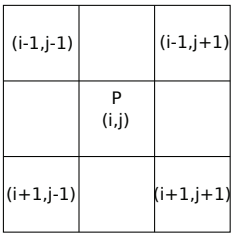

Fig. 3: The D-Neighborhood of a Pixel p.

The points of $N_4(p)$ and $N_D(p)$ together are called 8-neighborhood of p noted $N_8(p)$ / $N_8(p) = N_4(p) \cup N_D(p)$ (see figure Fig. 4). This set of points define on the other hand the 8-connectivity.

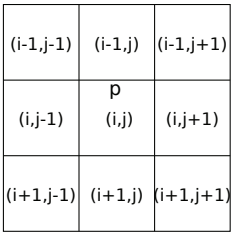

Fig. 4: The 8-Neighborhood of a Pixel p.

Two pixels are said to be connected if they are adjacent in some sense, i.e.

- they are neighbors ($N_4(p)$, $N_D(p)$ or $N_8(p)$), and
- their intensity values are the same (o or 1 for binary images).

When we apply thinning on binary images, it produces another binary image as output which is a skeleton. An example of a result of thinning the alphabet letter "L" is given in figure Fig.4 below.

In general, a non-zero (black) pixel is either a:

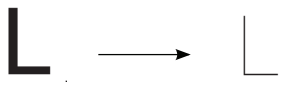

Fig. 5: Letter 'L' of Latin Alphabet (in the left) and Its Skeleton (in the right).

- break pixel: which is a non-zero pixel for which the deletion would break the connectivity of the original pattern or an
- edge pixel : also called boundary, or border pixel which is a non-zero pixel that has at least one zero (white) 4-neighbor pixel (i.e., it lies along the edge of the pattern), or an
- end pixel : which is a non-zero pixel that has at most one non-zero 8-neighbor pixel, or a
- simple pixel : which is an edge pixel whose removal from the object does not change the topology (or the connectivity).

We say that an algorithm preserves the connectivity properties [11] of any binary image if and only if :

- No object in S can be completely deleted by the algorithm in any iteration.
- It does not connect any originally dis-joint objects in S, nor does it disconnect an existing hole or create any new hole in S.
- It does not disconnect any object in S.
- And it does not connect any hole of S to another distinct hole or the background of S where a hole may be defined as a background region surrounded by a connected border of foreground pixels

III. ZS Thinning Algorithm

A very popular and well-proved thinning algorithm is the ZS algorithm proposed by Zhang and Suen in 1984 [9]. It is an iterative parallel thinning algorithm working on 3×3 neighborhood as shown in figure Fig. 6.

P8	P2	P3
P7	P1	P4
P6	P1	P5

Fig. 6: ZS 3×3 Neighborhood.

It is a directional algorithm which consists of two sub-iterations: the first is aimed at deleting the South-East boundary pixels and the North-West corner pixels while the second one is aimed at deleting the North-West boundary pixels and the South-East corner pixels that are the opposite orientations.

A. ZS Algorithm Description

The method for extracting the skeleton of a picture consists of removing all contour pixels of the picture except those belonging to the skeleton. In the first sub-iteration, the contour point p_1 is deleted from the pattern, if it satisfies the following conditions:

(a) $2 \leq B(p_1) \leq 6$
(b) $A(p_1) = 1$
(c) $p_2 \times p_4 \times p_6 = 0$
(d) $p_4 \times p_6 \times p_8 = 0$

In the second sub-iteration, the contour point p_1 is deleted from the pattern, if it satisfies the following conditions:

(a) $2 \leq B(p_1) \leq 6$
(b) $A(p_1) = 1$
(c') $p_2 \times p_4 \times p_8 = 0$
(d') $p_2 \times p_6 \times p_8 = 0$

where: $A(p_1)$ is the number of "0-1" (white-black, in this order) pairs in the clock-wise traversal of the 8-neighborhood of p_1, i.e. p_2,p_3,p_4,...,p_8, p_9. $B(p_1)$ is the number of non-zero 8-neighbors, that is:

$$B(p_1) = \sum_{i=2}^{9} P_i. \tag{1}$$

If any condition is not satisfied then p_1 will not be deleted from the foreground.

Although ZS is a simple and an efficient algorithm, it has some problems, mainly:

1) The loss of connectivity due to the complete disappearance of 2×2 square patterns in the thinning process which renders the algorithm invalid from the topological (connectivity) point of view, thus, the first property of a good algorithm cited in Section 1 is broken (see figure Fig. 7).

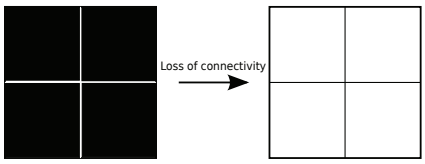

Fig. 7: ZS Thinning Result of a 2×2 Square.

2) The resulting skeleton is not one pixel thick which also breaks the first property of a good thinning algorithm. This is due to the presence of redundant pixels in the skeleton [11]. Redundant pixels are the remaining simple pixels whose deletion does not disconnect the picture, as shown in figure Fig. 8 where redundant pixels are labeled R.
3) It suffers from the excessive erosion of diagonal lines (see figure Fig. 9), thus, the fourth property of a good algorithm is not satisfied.

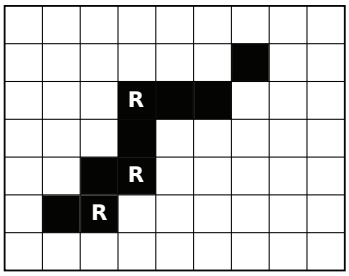

Fig. 8: Skeleton with Redundant Pixels Labeled R.

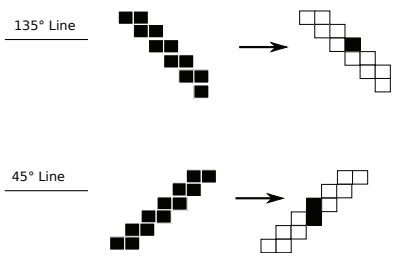

Fig. 9: ZS Thinning Result Excessively Eroded of a $135°$ Line and a $45°$ Line.

IV. New Thinning Algorithm

In this section, we propose a new hybrid thinning algorithm. It aims to settle ZS algorithm's problems cited in the last Section, to improve it, and to produce better skeletons.

Let S be the binary image to be processed, composed of elements p_i where $i \in 1, ..., 9$. Let P be the corresponding matrix of size $M \times N$ and elements $P(i, j)$. Then,

- our algorithm uses a 3×3 neighborhood as shown in figure Fig.6,
- p_1 is a black pixel selected for deletion,
- $B(p_1)$ is the number of "1" (black pixel) in the 8-neighborhood of pixel p_1 as defined in equation 1,
- $C(p_1)$ is the number of 8-connected components of "1" in the 8-neighborhood of the pixel p_1, [7] where :

$$\begin{aligned} C(p_1) = \neg p_2 \wedge (p_3 \vee p_4) + \neg p_4 \wedge (p_5 \vee p_6) \\ + \neg p_6 \wedge (p_7 \vee p_8) + \neg p_8 \wedge (p_9 \vee p_2) \end{aligned} \quad (2)$$

The new thinning algorithm is based on the directional approach used by ZS algorithm (cf. Section 3) and is combined with the sub-field approach [11] which consists of dividing an image into two sub-fields. We define 2 sub-fields according to the parity of pixels. The first sub-field is the set of odd pixels that is those for which the sum of their coordinates $i + j$ is odd and the second sub-field is similarity composed of the set of even pixels.

We describe our new algorithm as follows: In the first sub-iteration, an edge point p_1 is deletable if it satisfies the following conditions:

(a) $(i + j)mod2 = 0$
(b) $C(p) = 1$
(c) $2 \leq B(p_1) \leq 7$
(d) $p_2 \times p_4 \times p_6 = 0$
(e) $p_4 \times p_6 \times p_8 = 0$

In the second sub-iteration, an edge point p_1 is deletable if it satisfies the following conditions:

(a) $(i + j)mod2 \neq 0$
(b) $C(p) = 1$
(c) $2 \leq B(p_1) \leq 7$
(d) $p_2 \times p_4 \times p_8 = 0$
(e) $p_2 \times p_6 \times p_8 = 0$.

Where: $C(p) = 1$ implies that p is simple when p is a boundary pixel and the deletion will not disconnect the 3×3 neighborhood. $2 \leq B(p_1) \leq 7$ means that we examine a larger set of border points than that considered by ZS, it implies deletion of more boundary pixels. Indeed, ZS could have removed more pixels but it was limited by the conditions (a) and (b). (cf. Section 3).

V. Implementation, Tests and results

We have implemented both ZS and our new algorithm in C Language and subsequently tested them on a variety of binary images which involves patterns ranging from alphabet of various international languages like Chinese, Arabic, English to different shapes of objects like animals (duck, horse), handwritten characters, and numbers.

In order to evaluate our new algorithm and ZS, comparisons are done using the following performance measures:

- Thinning rate (TR): Or the degree of thinness of the image as it was proposed in [12] by the following formula:

$$TR = 1 - TM1 \div TM2 \quad (3)$$

Where:
$TM1$ stands for total triangle count of the thinned image given by:

$$TM1 = \sum_{i=0}^{n} \sum_{i=0}^{m} TC(P[i][j]) \quad (4)$$

Where: $P[i][j]$ is a black pixel with coordinates i, j. TC is a function which counts the number of black triangles which can be created from $P[i][j]$ and its neighboring pixels given by:

$$TC(p) = (p_8 * p_9) + (p_9 * p_2) + (p_2 * p_3) + (p_3 * p_4) \quad (5)$$

$TM2$ represents the largest number of black triangles that an image could have computed as:

$$TM2 = 4 * [max(m, n) - 1]^2 \quad (6)$$

and n, m are dimensions of picture.
When $TR = 1$ the image is perfectly thinned, but $TR = 0$ means that the image is not thinned at all [10].

- Thinning Speed (TS): measures the number of pixels thinned per time unit (second) given by:

$$TS = \frac{DP}{ET} \quad (7)$$

$$DP = OP - SP \tag{8}$$

Where: DP or (deleted points) is the number of black points deleted during thinning, OP or (object points) is the number of black points of the image before thinning, SP or (skeletal points) is the number of black points remaining after thinning or the number of pixels in the skeleton and ET is the execution time of thinning process in second.

A. Tests and Results

In the following, we report both visual and numerical results of ZS and our new algorithm on four kind of shapes and confront the algorithms on erosion, thinness and speed.
Test 1: 2×2 square

First we test our algorithm on the 2×2 square. The application of the proposed algorithm is shown in figure Fig.10.

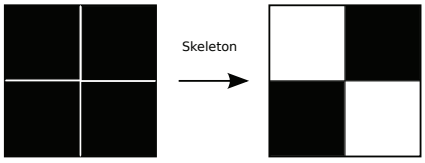

Fig. 10: Thinning Result of a 2×2 Square by the Proposed Algorithm.

As we can see, there are two skeletal pixels in the proposed thinning result, that are not redundant, whereas, ZS makes the structure completely disappear (Fig.7). Thus, by preserving the connectivity of the 2×2 patterns, the new thinning algorithm successfully solves the first ZS's problem (the loss of topology).

Test 2: diagonal lines

Second, we test our algorithm on the same diagonal lines that ZS tends to excessively erode (cf. Fig.9). The result of the application of our new algorithm over the diagonal lines is shown in figure Fig.11. We see that our algorithm deals successfully with the diagonal patterns. There are more skeletal pixels than those thinned by ZS, thus our algorithm solves the excessive erosion problem of ZS. So, it preserves more information about the original pattern.

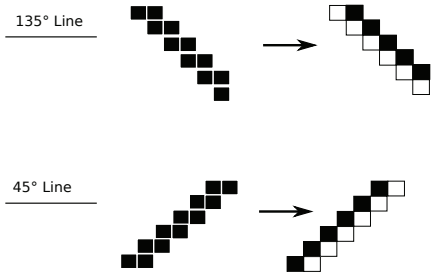

Fig. 11: Thinning Result of $135°$ Line and $45°$ Line by the New Algorithm.

Test 3: real life image

Third, we give an example of a 256×256 binary image representing a real life pattern which is a duck. After application of ZS, we obtain the superimposed white skeleton over the black original pattern (duck) with a thinning rate $TR1 = 0.999612$ as shown in figure Fig.12.

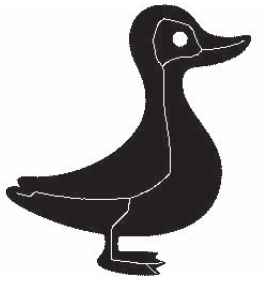

Fig. 12: White Skeleton Superposed on the Original Pattern Produced by ZS.

Figure Fig.13 shows the result of the application of the new thinning algorithm over the same duck image, for which the thinning rate is $TR2 = 0.999981$.

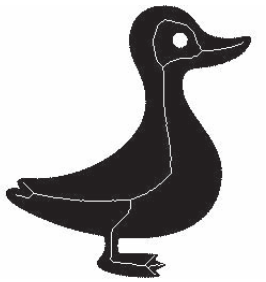

Fig. 13: White Skeleton Superposed on the Original Pattern Produced by the New Thinning Algorithm.

The thinning rate for the image duck achieved by our algorithm is superior to that achieved by ZS $(0.999981 > 0.999612)$, which reflects the fact that the skeleton does not contain redundant pixels (it is one width) and therefore constitutes a better thinned skeleton. Moreover, one can observe that the geometric characteristics are completely preserved despite that our algorithm removes more pixels. Indeed, the paw and tail, well thinned, better reflect the original duck shape which ZS algorithm fails to.

Test 4: digital life images

This class of images is produced by computer software or hardware as scanners and cameras. Figures Fig.14 and Fig.15 show superposed original and thinned images of different Latin Alphabet letters and Arabic numbers, respectively.

Table.I gives a summary of performance details for all the experiments shown before, for both algorithms.

The performance evaluation confirms that the new algorithm is better than ZS in terms of: First, thinness, since our new algorithm produces thinner skeletons of one width such $TR2 > TR1$ in all test cases. Second, avoid the excessive erosion for the too little patterns like diagonal lines (like $135°$

TABLE I: Comparing Thinning Performance of ZS and the New Algorithm

Images	Object points	ZS Algorithm			New Algorithm		
		TR1	TS1(p/s)	SP1	TR2	TS2(p/s)	SP2
line135°	11	1, 00	248517, 47	1	1, 00	2080218, 73	5
line45°	12	1, 00	440547, 98	2	1, 00	1441506, 86	6
duck	21643	0, 999612	212157, 99	570	0, 999981	256628, 65	538
alphabet	26090	0, 999432	494780, 92	3612	0, 999985	537456, 30	3307
numbers	18558	0, 999619	470721, 00	1889	0, 999994	518760, 61	1732

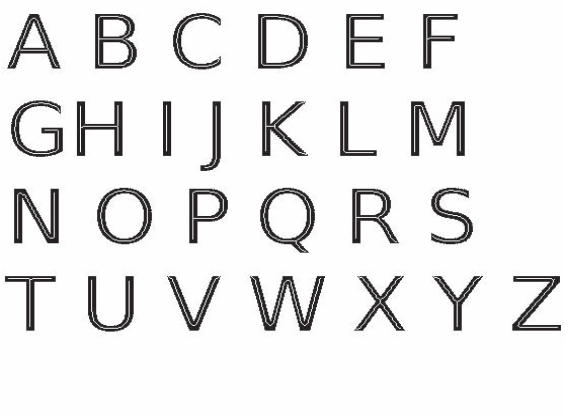

Fig. 14: White Skeleton Superposed on the Original Alphabet Pattern Produced by the New Thinning Algorithm.

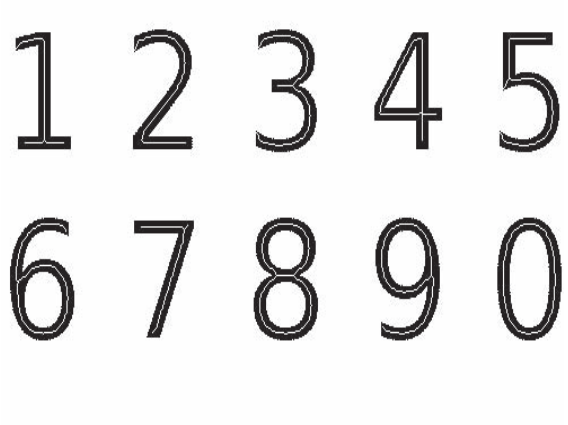

Fig. 15: White Skeleton Superposed on the Original Arabic Numbers Pattern Produced by the New Thinning Algorithm.

line). Third, it preserves the connectivity of the 2×2 square as the number of skeletal pixels is different from zero. Fourth, our algorithm runs in polynomial time ($O(n^4)$), moreover, it is faster, as the thinning speed is higher in all the tests where the speed result is based on an execution time computed as the mean of 100 different experiments. Finally, our algorithm produces better visual quality (middle of the shape, preserving the original shape). Hence, the efficiency of the new proposed algorithm.

VI. Conclusion

In this paper we have proposed a new hybrid thinning algorithm based on the directional approach used by ZS, combined with a subfield approach. After implementation, we have compared the two algorithms. The results of performance evaluation and experiments show that the new algorithm is faster and produces better skeletons in terms of thinness and visual quality, solves the problem of loss of connectivity due to the complete removal 2×2 square incurred in ZS and adequately avoids excessive erosion. In perspective, we are going to compare our algorithm with other much more recent algorithms but no less used, such as the one in [4]. On the other hand, it will be interesting to study other performance metrics such as connectivity and noise sensitivity among others.

References

[1] H. Blum, "An associative machine for dealing with the visual field and some of its biological implications," *Biological Prototypes and Synthetic Systems*, pp. 244–260, 1962.

[2] E. Hastings, "A survey of thinning methodologies," *Pattern analysis and Machine Intelligence,IEEE Transactions*, vol. 4, pp. 869–885, 11 1992.

[3] L. Lam and Al., "Thinning methodologies-a comprehensive survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 14, pp. 869–885, 9 1992.

[4] A. Jagna, "An efficient image independent thinning algorithm," *International Journal of Advanced Research in Computer and Communication Ingineering*, vol. 3, 10 2014.

[5] C. Louhou, "Contribution à l'analyse topologique des images : Etude d'algorithmes de squelettisation pour images 2d et 3d,selon une approche topologie digitale ou topologie discrète," Ph.D. dissertation, 2001.

[6] S. S. I. Weian Deng and N. E. Brener, "A fast parallel thinning algorithm for the binary image skeletonization," *International Journal of High Performance Computing Applications*, vol. 14, no. 1, pp. 65–81, 2000.

[7] A. Rosenfeld and A. Kak, *Digital Picture Processing*, elsevier ed., New York, 1976.

[8] P. Tarabek, "A robust parallel thinning algorithm for pattern recognition," *7th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pp. 75–79, 5 2012.

[9] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM*, vol. 27, no. 3, pp. 236–239, Mar. 1984. [Online]. Available: http://doi.acm.org/10.1145/357994.358023

[10] P. Tarabek, "Performance measurements of thinning algorithms," *Journal of Information, Control and Management Systems*, vol. 6, no. 2, 2008.

[11] Z. Guo and R. W. Hall, "Parallel thinning with two-subiteration algorithms," *Commun. ACM*, vol. 32, no. 3, pp. 359–373, Mar. 1989. [Online]. Available: http://doi.acm.org/10.1145/62065.62074

[12] R. G.S. Ng and C. Quek, "A novel single pass thinning algorithm," *IEEE Transaction on System Man and Cybernetics*, 1994.