УДК 621.391

FAST FULLY PARALLEL ONE-SUBITERATION THINNING ALGORITHM

JUN MA, V.Yu. TSVIATKOU, V.K. KONOPELKO

Belarusian State University of Informatics and Radioelectronics, Republic of Belarus

Submitted 30 October 2019

Abstract. Fast thinning algorithm with one-subiteration was proposed. Results of implementing the proposed algorithm, on a variety of binary images and comparison with ZS algorithm and OPTA algorithm show better results in terms of thinning speed, thinning rate and visual quality.

Keywords: Thinning algorithm; Binary images; Thinning Rate; Thinning speed.

Introduction

Thinning algorithm is generally deemed as a fundamental preprocessing algorithm that plays a critical role in many pattern recognition applications. Thinning algorithm tend to remove edge contour layer by layer and only leave the skeleton which has same topology with the original image. There is a large amount of literatures on thinning algorithms [1–6].

The mainly structure of the classification of the thinning algorithm as shown in Fig. 1.

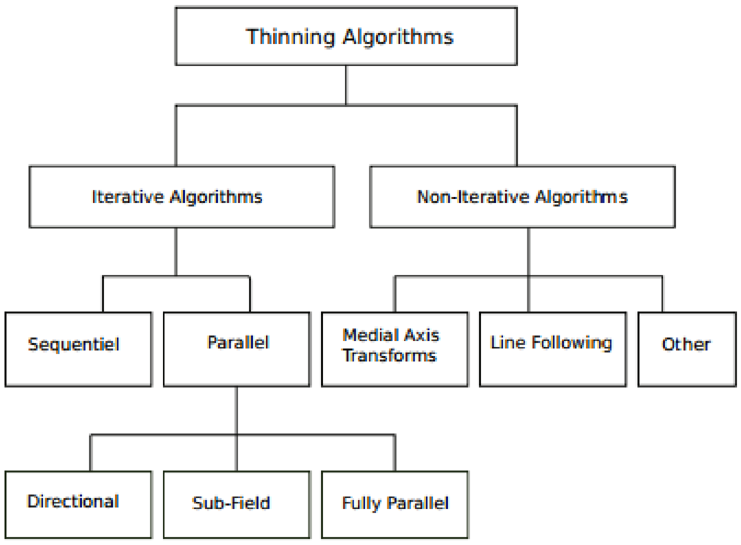

Fig. 1. Classification of the thinning algorithm

Thinning algorithm can be divided into iterative algorithm and non-iterative algorithm at first. Iterative algorithm works on the pixel by pixel based on thinning, however, non-iterative algorithm is just opposite to iterative. Non-iterative algorithm can further divide in medial axis transforms, line following and others.

According to the approach of examine pixels, iterative thinning algorithm can be classified as sequential or parallel [3]. In a sequential algorithm, each deletion pixel in the n-th iteration depend on all the operations that have been performed, however, in a parallel algorithm, the deletion of pixels in the n-th iteration only depend on the result that remains after the $(n-1)$-th; therefore, all pixels can be examined independently in a parallel manner in each iteration.

Parallel thinning algorithms can be further classified in two categories according to used approach. There are mainly three parallel methods: the directional approach, the subfield approach and the fully parallel approach. The directional approach consists of breaking the thinning iteration into several sub – iteration of thinning of which based on a combination of direction. The subfield approach consists of breaking down the figure into sub – field according to some criterion such as the parity of pixels. As for the full parallel approach is a method without any sub − iteration, in this way all the pixels are examined by the same criterion.

A good thinning algorithm should be satisfied with the following requirements [7]. First of all, the output of the thinning algorithm should maintain the original topology and geometric properties of the input image, and at the same time have one-pixel width. Second, the skeleton of the output should be located at the medial line of the original image. Besides, thinning algorithm should not be sensitive to the boundary noise. And the last, thinning algorithm should be proven to be high efficiency, which means it should process a picture as fast as possible.

Basic notations

The binary image *I* is denoted by a matrix **M** with the size $R \times C,$ 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 build up the foreground of *I*, and the white pixels form the background which is the complement of the foreground.

The set of four pixels which called 4 – neibourhood of the pixel p is noted as $N_4(p)$ as shown in Fig. 2.

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

	$(i-1, j)$	
$(i, j-1)$	$P(i, j)$	$(i, j+1)$
	$(i+1, j)$	

Fig. 2. The 4 – Neighborhood of a pixel p

$(i-1, j-1)$		$(i-1, j+1)$
	$P(i, j)$	
$(i+1, j-1)$		$(i+1, j+1)$

Fig. 3. The D – Neighborhood of a pixel p

The $N_4(p)$ and $N_D(p)$ together formed the concept of the 8-neighborhood, which noted as $N_8(p).$ This set of points define on the other hand the 8-connectivity.

Two pixels can seem as connected if they appear in one another's 8-neighborhood.

In general, a non-zero pixel can be classified as 4 special points, which are break pixel, edge pixel, end pixel and simple pixel respectively. Break pixel: which is a non-zeros pixel for which the deletion would break the connectivity of the original pattern. Edge pixel, also called as boundary or border pixel ,has at least one zero 4-neighbor pixel. And end point has at most one non-zero 8-neighbor pixel. As for simple pixel is an edge pixel whose removal from the object does not change the topology.

Here according to the criterion, which used to describe a good thinning algorithm, it is reasonable to deduced that an algorithm can be a good algorithm if and only if following requirement can be satisfied.

1. No object in I can be completely deleted by the algorithm in any iteration;

2. It does not connect any originally disjoint objects in image $I,$ nor does it disconnect an existing hole or create any new hole in $I;$

3. It does not disconnect any object in $I;$

4. It does not connect any hole of I to another distinct hole or the background of I where a hole may be defined as a background region surrounding by a connected border of foreground pixels.

Thinning algorithm

The scan window used to find out all the deletable candidate points is expanded from the general window whose size is 3×3 in the fast fully parallels algorithm proposed in this paper, as is shown in Fig. 4.

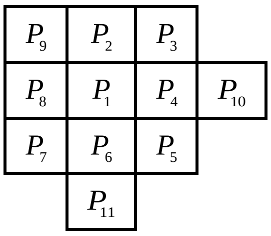

Fig. 4. Scan window

Our algorithm for extracting the skeleton of a picture consists of removing all the contour points of the picture except those points that belong to the skeleton. In each iteration, the contour point P_1 is deleted from the digital pattern if it satisfies the following conditions at the same time:

(a) $2 \le B(P_1) \le 6,$

(b) $A(P_1) = 1,$

(c) $P_4 = 0$ or $P_8 + P_{10} > 0,$

(d) $P_6 = 0$ or $P_2 + P_{11} > 0,$

where $A(P_1)$ is the number of 01 pattern in the ordered set $P_2, P_3, \ldots, P_9$ hat are the eight neighbors of P_1, and $B(P_1)$ is the number of nonzero neighbors of P_1, that is,

$$B(P_1) = \sum_{i=2}^{9} P_i.$$

By condition (a), the end points of a skeleton lines are preserved. In another word, it means there is only one 8-connected component in the neighborhood of P_1, which implies there is deletable when P_1 is a boundary pixel and the deletion will not break the connectivity of the 8 neighborhood. Besides, condition (b) prevents the deletion of those points that lie between the end points of skeleton line. Also, condition (c) and (d) can preserve the excessive erosion of the 2×2 pattern.

After iteratively remove the boundary and left the skeleton of the original image by applying operations mentioned above, an extra delete procedure is introduced to ensure the single thick of the skeleton. The additional procedure only performed once, so it will not increase the cost of the time dramatically. In this procedure, a normal 3×3 scan window is used to examine the skeleton pixel by pixel. If one of the following conditions is satisfied, current pixel can be removed

(a') $P_4 \times P_6 = 1$ and $P_9 = 0,$

(b') $P_6 \times P_8 = 1$ and $P_3 = 0,$

(c') $P_2 \times P_4 = 1$ and $P_7 = 0,$

(d') $P_2 \times P_8 = 1$ and $P_5 = 0.$

After all the pixels have been processed, the whole thinning algorithm is finished.

Experiments and compare

In this section, well known thinning algorithm ZS [6], OPTA [8] and our algorithm were implemented in Matlab2018 and subsequently tested them on a variety of binary images which involves patterns from basic shapes to animal shapes.

In order to evaluate our new algorithm and these famous thinning algorithms, comparisons are done using the following performance measures:

Thinning rate (TR): Or the degree of thinness of the image as it was proposed in [9] by the following formula:

$$TR = 1 - TM1 \div TM2,$$

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

$$TM1 = \sum_{i=0}^{n} \sum_{j=0}^{m} TC\left(P(i, j)\right),$$

where $P(i, j)$ is a foreground pixel with coordinates $i, j.$ TC is a function which counts the number of the black triangles which can be created from $P(i, j)$ and its neighboring pixels given by the following formula:

$$TC(P_1) = P_1 \times [(P_8 \times P_9) + (P_2 \times P_9) + (P_2 \times P_3) + (P_3 \times P_4)].$$

TM2 represents the largest number of black triangles that an image could have computed. This number is computed as follows:

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

where m and n denote the number of the row and number of the column of the picture respectively.

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:

$$TS = DP \div ET,$$

$$DP = OP - SP,$$

where DP or (deleted points) is the number of black points deleted during thinning, OP is the number of black points of the image before thinning. SP or (skeleton points) are the number of the remaining points after applying the thinning algorithm or the number of pixels in the skeleton and ET denote the real execution time.

In the following, we report both visual and numerical results of ZS, OPTA and our new algorithm on test image.

First test image consists of 2×2 square, 135° lines and 45° lines on which the application of the ZS algorithm and OPTA algorithm will cause some distortion, such as excessive error or the existence of redundant pixels which cannot ensure that the skeleton is precisely one pixel thick, which mentioned in some literatures [5, 11–13]. The application results are shown in Fig. 5.

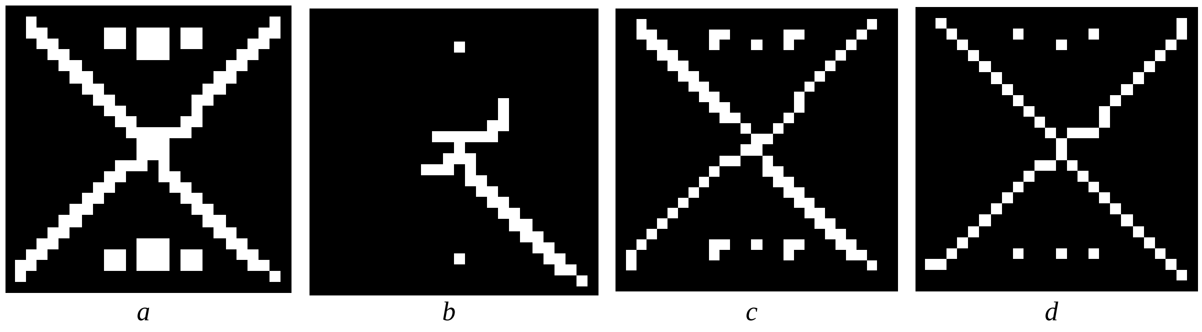

Fig. 5. Input image and its skeletons: *a* – original image; *b* – ZS; *c* – OPTA ; *d* – our algorithm

In Fig. 5, ZS algorithm both confronts serious excessive error in diagonal and in 2×2 square, what's more, it also suffered the redundant pixels. The only problem of the OPTA is the existence of the redundant pixels. Our algorithm deals successfully with these patterns.

Next, more images from the dataset kimia99 have been introduced to test, the results of the different thinning algorithm are shown below in Fig. 6.

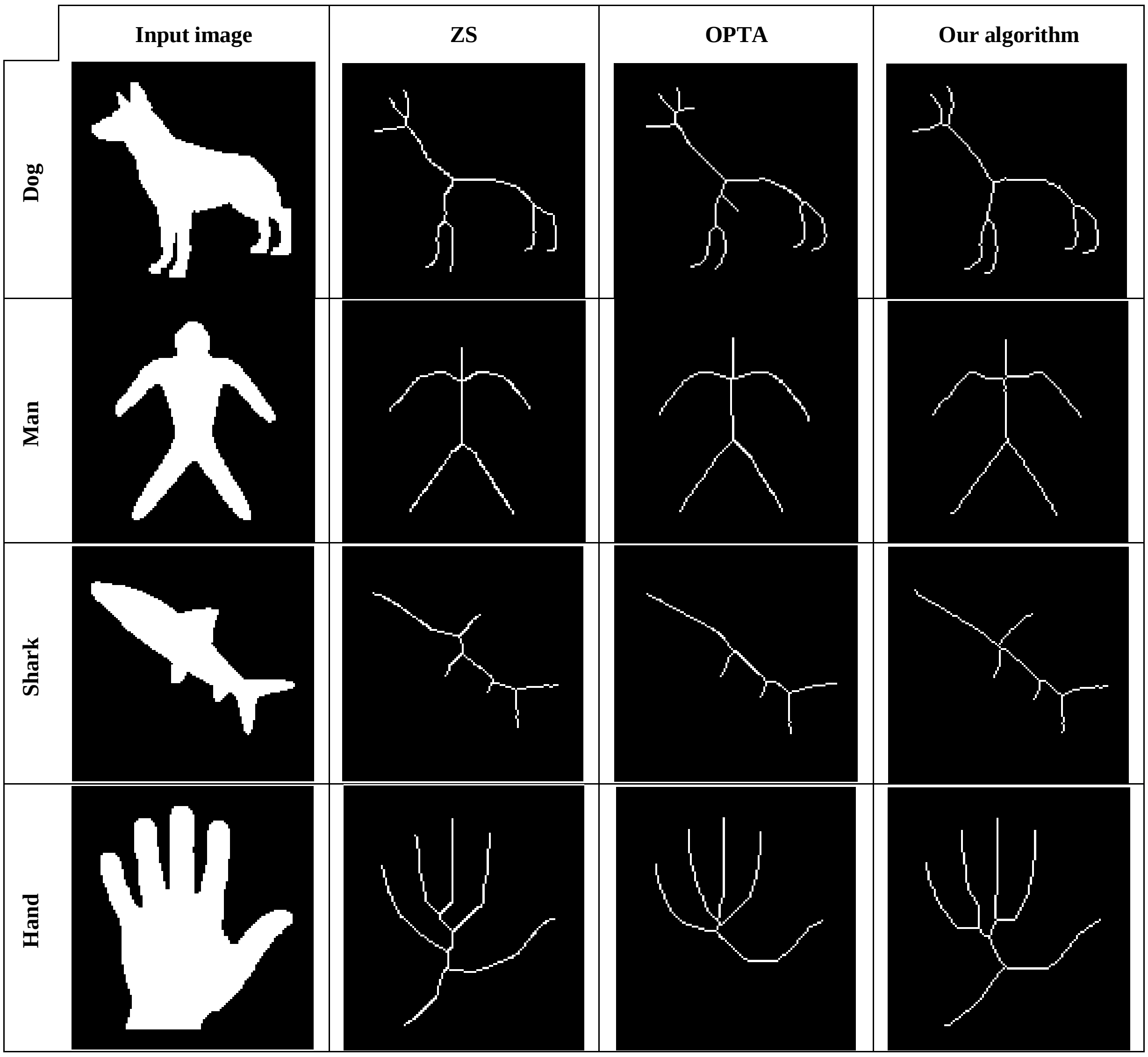

Fig. 6. Input images and its skeletons

Tabl. 1 gives a summary of performance details for all skeletons extracted from the binary image above.

Table 1. **Comparison results of thinning performance for ZS, OPTA and our algorithm**

Images	Object points	ZS			OPTA			Our algorithm		
		*TR*1	*TS*1 (p/s)	*SP*1	*TR*2	*TS*2 (p/s)	*SP*2	*TR*3	*TS*3 (p/s)	*SP*3
Dogs	3345	0,9991	426954	262	0,9996	191358	253	0,9999	601456	251
People	2699	0,9986	644012	232	0,9992	273963	227	1	909690	203
Shark	2402	0,9991	613532	187	0,9993	267551	167	1	810920	169
Hand	5270	0,9987	442326	333	0,9999	179904	250	1	641850	303

The performance evaluation confirms that our algorithm is better than ZS and OPTA in terms of: thinness, since our algorithm produces thinner skeletons of one width thick such $TR3 > \max(TR, TR2)$ in all test cases. Second, our results without excessive in diagonal lines. Moreover, indicator of the thinning speed much higher than both ZS and OPTA, with about 1,5 times and 3 times higher than the ZS algorithm and OPTA algorithm respectively. Finally, our algorithm produces better visual quality (middle of the shape, preserving the original shape).

Conclusion

In this paper we have proposed a new fully fast parallel one-subiteration thinning algorithm. After compare with the famous thinning algorithm ZS and OPTA, the results of performance evaluation and experiments show that new algorithm is faster and produces better skeleton without any excessive erosion.

References

1. Lam L., [et. al.] // Pattern Analysis and Machine Intelligence, IEEE Transactions. 1992. Vol. 14. P. 869–885.
2. Jagna. A. // International Journal of Advanced Research in Computer and Communication Engineering. 2014. Vol. 3. P. 8309–8311.
3. Brener N.E., Weian Deng S.S.I. // International Journal of High Performance Computing Application. 2000. Vol. 14. P. 65–81.
4. Rosenfeld A., Kak A. Digital Picture Processing. Elsevier. 1976.
5. Tarabek P. // 7th IEEE International Symposium on Applied Computational Intelligence and Informatics. 2012. P. 75–79.
6. Zhang. T.Y., Suan C.Y. // Commun. ACM. 1984. Vol. 27. P. 236–239.
7. Hastings E. // Pattern analysis and Machine Intelligence, IEEE Transactions. 1992. Vol. 14. P. 869–885.
8. Chen C.S., Tsai W.H. // Pattern Recognition Lett. 1990. P. 471–477.
9. Quek C., Ng R.G.S. // IEEE Transcation on System Man and Cybernetics. 1994.
10. Tarabek P. // Journal of Information. Control and Management Systems. 2008. Vol. 6. P. 125–132.
11. Abdulla W.H., Saleh A.O.M., Morad A.H. // Pattern Recognition Letters 7. 1988. P. 13–18.
12. Juan H.S. // Pattern Recoginition Letters 10. 1989. P. 77–80.
13. Boudaoud L.B., Sider A., Tari A. // Vis Comput. 2018. P. 689–706.