ELEKTRİK, VOL.5, NO.1, 1997, ©TÜBİTAK EMO

A Fast Parallel Skeletonization Algorithm on Block Represented Binary Images

Iraklis M. Spiliotis and Basil G. Mertzios
Automatic Control Systems Laboratory
Department of Electrical and Computer Engineering
Democritus University of Thrace
Xanthi 67100 HELLAS
Fax: +3054126473
email: spiliot@demokritos.cc.duth.gr
mertzios@demokritos.cc.duth.gr

Abstract

This paper describes a binary image representation scheme, which is called Image Block Representation and presents a new skeletonization algorithm, which is fast implemented on block represented binary images. The main purpose of the Image Block Representation is to provide an efficient binary image representation that permits the execution of operations on image areas instead of image points. The skeletonization algorithm operates in four subiterations: each subiteration deletes the north, the south, the west and the east boundary points, respectively. Due to the substitution of the boundary points by the blocks' boundary points, the relevant operations are performed fast, while preserving the end points and the object connectivity.

Keywords: *Image Block Representation, Skeletonization, Thinning.*

1. Introduction

The most common image representation format is the two-dimensional (2-D) array. However, many research efforts for deriving alternative image representations have been motivated by the need for fast processing of huge amount of data. Such image representations aim to provide machine perception of images in pieces larger than a pixel and are separated into two categories: boundary based methods and region based methods. The region based representations include quadtree [1], run length encoding [2],[3] and interval coding representation [4]. The boundary based representations include chain code [5], contour control point models [6] and autoregressive models [7]. In the context of binary images, the run length and interval coding representations are identical. Recently, a region based method, called Image Block Representation has been presented [8] - [12]. In the image block representation process the whole binary image is decomposed into a set of rectangular areas with object level, which are called *blocks*, so the fact that many compact areas of a given binary image have the same value is exploited.

As in every other region based method, the most important characteristic of the image block representation is that a perception of image parts greater than a pixel, is provided to the machine and therefore,

all the operations on the pixels belonging to a block may be substituted by a simple operation on the block. Taking this feature into account, the implementation of new algorithms for binary image processing and analysis tasks, leads to a substantial reduction of the required computational complexity. The image block representation method is superior to run length coding and to quadtree representation.

A number of approaches for thinning and skeletonization have appeared in the literature. These approaches may be separated in two classes: the iterative and the noniterative methods. The iterative methods [13]-[17] are based on successive removal of the outermost layers of the object until a unit width connected skeleton remains. The *Medial Axis Transform* (MAT) introduced by Blum in 1967 [13], is a classical way to obtain skeleton, where the number of the required iterations is proportional to the width of the object. Because of their repetitive nature the above algorithms appear to have a high time complexity. The noniterative methods [18], [19] produce a center line of the object in a single pass of the image. A survey of thinning algorithms is provided in [20].

This paper presents a skeletonization algorithm, which is characterized by a low computational cost and it is suitable for fast processing rates, due to the substitution of image pixels with blocks. The algorithm operates in four subiterations: each subiteration deletes the north, the south, the west and the east boundary points, respectively. Due to the substitution of the boundary points by the blocks boundary points, the relevant operations are performed very fast.

2. Image Block Representation

A bilevel digital image is represented by a binary 2-D array. Due to this kind of representation, there are rectangular areas of object value, in each image. These rectangular areas, which are called *blocks*, have their edges parallel to the image axes and contain an integer number of image pixels. At the extreme case, one pixel is the minimum rectangular area of the image. It is always feasible to represent a binary image with a set of all the nonoverlapping blocks with object level and this information lossless representation is called *Image Block Representation (IBR)*. According to the above discussion, two useful definitions are formulated:

Definition 1

A *block* is a rectangular area of the image, with edges parallel to the image axes, that contains pixels of the same value. ■

Definition 2

A binary image is called *block represented*, if it is represented as a set of blocks with object level, and if each pixel of the image with object value belongs to one and only one block. ■

A block represented binary image $f(x, y)$ is comprised of a set of nonoverlapping blocks that completely cover the image areas with object level and it is denoted as:

$$f(x, y) = \{b_i : i = 0, 1, \ldots, k-1\} \tag{1}$$

where k is the number of the blocks. Each block is described by the coordinates of two corner points, i.e.:

$$b_i = (x_{1,b_i}, x_{2,b_i}, y_{1,b_i}, y_{2,b_i}) \tag{2}$$

where for simplicity it is assumed that: $x_{1,b_i} \leq x_{2,b_i}$ and $y_{1,b_i} \leq y_{2,b_i}$, as shown in Fig. 1. In Fig. 2, the blocks that represent an image of the character d are illustrated.

The block representation concept leads to a simple and fast algorithm, which requires just one pass of the image and a simple bookkeeping process. In fact, considering a $N_1 \times N_2$ binary image $f(x,y), x = 0,1,,\ldots,N_1-1, \quad y = 0,1,..,N_2-1$, the block extraction process requires a pass from each line y of the image. In this pass all object level intervals are extracted and compared with the extracted blocks that have pixels in the previous line. The extracted blocks are denoted by $b_i : i = 0,1,\ldots,k$, the indices of the extracted blocks that have pixels in the previous line at each stage are denoted by $p_i : i = 0,1,\ldots,k_p$ and the indices of the extracted blocks that have pixels in the current line are denoted by $c_i : i = 0,1,\ldots,k_c$. Then the first block that has been extracted at the previous line is the b_{p_0} and the last block that has been extracted at the current line is the $b_{c_{k_c-1}}$. In the sequel the algorithm for the Image Block Representation is given.

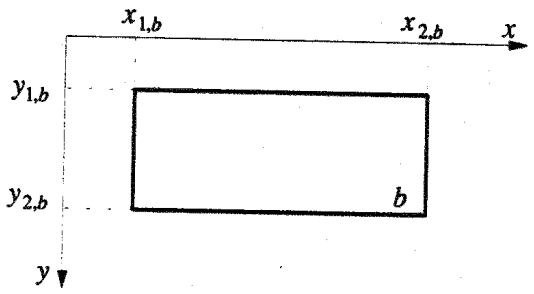

Figure 1. Each block b is described by the co-ordinates of its two corner points.

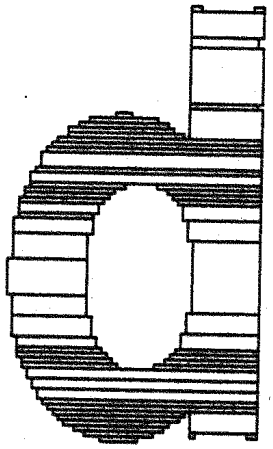

Figure 2. Image of the character d and the blocks.

Algorithm 1: Image Block Representation

Step 1: Consider each line y of the image f and find the object level intervals in line y.

Step 2: Compare intervals and blocks that have pixels in line $y-1$.

Step 3: If an interval does not match with any block, this is the beginning of a new block.

Step 4: If a block matches with an interval, the end of the block is in the line y. ■

In many operations it is important to have information concerning not only the location of the blocks but also information concerning the neighbor and connected blocks. The following definition provides a template for block connectivity:

Definition 3

Two blocks are defined as connected, if their projections on both the x or y axis are overlapped or they are neighbors. ■

The information concerning block connectivity requires a suitable data structure for storage. Therefore, each block b_i is represented as the ordering:

$$b_i = (x_{1,b_i}, x_{2,b_i}, y_{1,b_i}, y_{2,b_i}, nc_i, \mathbf{c}_i) \tag{3}$$

where x_{1,b_i}, x_{2,b_i} are the coordinates of the i-th block according to the horizontal axis, y_{1,b_i}, y_{2,b_i} are the coordinates of the block according to the vertical axis, nc_i is the number of the connected blocks and $\mathbf{c}_i$ is a list with the indexes of these connected blocks.

The Image Block Representation reduces to run length encoding or to quadtree representation of binary images at the extreme cases, where each block is comprised of pixels belonging to only one row of the image. Such a case is that of a chessboard image, where the transitions from white to black have 1 pixel length and the number of the blocks is $N^2/2$, i.e. it is exactly equal to the object run lengths or to object quadtree nodes. However, in real world situations, the image block representation is superior to run length encoding and quadtree representation, since the number of the blocks is significantly smaller than the number of runs and the number of nodes with object level, respectively.

In Fig. 3 three test images are illustrated, while in Table 1 the number of pixels with object level, the number of rows with object pixels, the number of blocks extracted from these images (using Algorithm 1), the number of object runs, the number of object quadtree nodes and the required time for the three representations are shown. It can be seen that the number of blocks generated by Algorithm 1 is significantly less than the number of rows with object pixels and therefore the IBR is superior in comparison with the run length. Also, it is obvious from Table 1 that the IBR is superior in comparison with the quadtree representation. Since the time complexity of the algorithms that are based on a region based image representation method, it is usually strongly related to the number of image areas that each specific method handles, it is expected that the algorithms based on the IBR are faster than algorithms based on the run length or the quadtree representation.

From Table 1 it is worth noting that the number of quadtree nodes with object level is significantly larger in comparison with the two other representation schemes. This is explained by the fact that in the quadtree representation, square image areas should be extracted with the additional requirement that these areas should be located at specific image positions. This results in an extended partitioning of the extracted areas and therefore in a large number of object quadtree nodes. The computational times of Table 1, as well as the times of Tables 2 and 3, have been measured in a SUN Sparcstation 4 and the relevant algorithms have been implemented using the C language.

Different IBR algorithms, which may result in a smaller number of blocks at the cost of increased time, may be implemented. Specifically, the algorithm for finding the maximal rectangle [21]-[23] in an area may be applied recursively. At each stage the points of this rectangle are labelled as background points, in order to apply the algorithm recursively to the remaining points. The time complexity of this procedure is $O(n \log n)$, where n is the number of the points of the area at each recursion. Obviously, the total complexity of this method depends on the number of the extracted blocks, i.e. is image dependent. The optimum representation is characterized by the minimum possible number of blocks. However, the selection of the optimum representation implies an additional computational cost, which may compensate the achieved savings due to the smaller number of blocks. It is pointed out that, given a specific binary image, different sets of different blocks can be formed. Actually, the nonunique block representation does not have any

implications on the implementation of any operation on a block represented image.

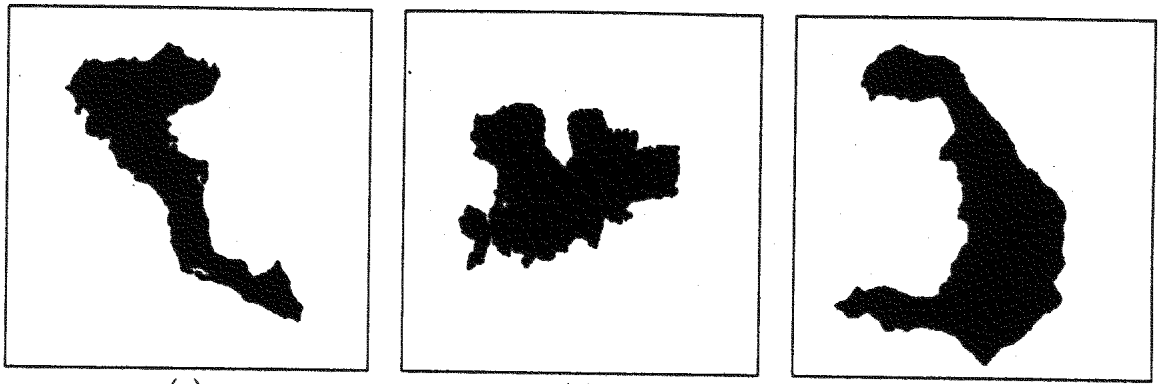

Figure 3. A set of test images. (a) Image of the island Corfu of 512×512 pixels. (b) Image of the island Mikonos of 512×512 pixels. (c) Image of the island Santorini of 512×512 pixels.

Table 1. The number of the pixels with object level, the number of the rows with object pixels, the number of the blocks, the number of runs with object level and the number of the quadtree nodes with object level for the set of the test images of Figure 3.

Image	Object pixels	Object rows	Image Block Representation		Run Length Coding		Quadtree Representation	
			blocks	time (sec)	object runs	time (sec)	object nodes	time (sec)
Island Corfu	41605	411	250	0.615	526	0.598	2074	0.534
Island Mikonos	47368	249	232	0.615	530	0.598	1876	0.549
Island Santorini	63203	474	257	0.619	553	0.598	2048	0.512

3. Logic Operations

Logic operations are performed fast in block represented binary images [10], [11]. In this Section the required logic algorithms for the execution of the skeletonization process, are presented. Since each image is represented by a set of nonoverlapping blocks, binary operations among the pixels of the images may be substituted by binary operations on the blocks of the images. The results are always block represented images. The implementation of the Set Difference operator which is used as an auxiliary operator for the execution of the OR, XOR and NOT operations in block represented images is also presented.

A. The SET DIFFERENCE operation

The Set Difference (SETDIF) operator is used as an auxiliary operator for the execution of the OR, XOR and NOT operations in block represented images. The execution of the SETDIF operation is

implemented easily with blocks that represent 2-D arrays. In the following the implementations of the operation SETDIF between two blocks and two images are described.

Operation SETDIF between two blocks

The execution of the operation SETDIF between two blocks b_0, b_1 results in all the points of b_0 that do not belong to b_1. In order that the output of the SETDIF operation be represented by blocks, the determination of the relative positions of the two blocks is required. In Fig. 4 all the possible relative positions between two blocks b_0, b_1 in a rectangular grid are shown.

At first the projections of the two blocks on each of the two axes are examined. A corresponding variable X that describes the relative position of the projections of the two blocks in the x-axis, is set to an appropriate value, according to the following criteria:

If $x_{1,b_0} > x_{2,b_1} + 1$ OR $x_{2,b_0} + 1 < x_{1,b_1}$ then X=-1

If $x_{1,b_0} \leq x_{1,b_1}$ AND $x_{2,b_0} \leq x_{2,b_1}$ then X=0

If $x_{1,b_0} \leq x_{1,b_1}$ AND $x_{2,b_0} > x_{2,b_1}$ then X=1

If $x_{1,b_0} > x_{1,b_1}$ AND $x_{2,b_0} \leq x_{2,b_1}$ then X=2

If $x_{1,b_0} > x_{1,b_1}$ AND $x_{2,b_0} > x_{2,b_1}$ then X=3

Using the same criteria for the projections of the two blocks in the y-axis, the corresponding variable Y is set to the appropriate value.

At the next stage the relative position of the two blocks is determined by the index function P(X,Y)=X+4Y, which is defined for X,Y $\geq$ 0. If X<0 or Y<0 then the two blocks do not intersect and the result of SETDIF operation is identical to b_0.

After the determination of the relative position of the two blocks, the appropriate resultant blocks are formulated for each value of the function P, as shown in Fig. 5. For example for the case of P=5, four blocks are formulated with coordinates $(x_{1,b_0}, x_{2,b_0}, y_{1,b_0}, y_{1,b_1} - 1)$, $(x_{1,b_0}, x_{1,b_1} - 1, y_{1,b_1}, y_{2,b_1})$, $(x_{2,b_1} + 1, x_{2,b_0}, y_{1,b_1}, y_{2,b_1})$, $(x_{1,b_0}, x_{2,b_0}, y_{1,b_1} + 1, y_{2,b_0})$, while in the case of P=10 the result is no blocks.

B. The OR operation

Considering the fact that the blocks consist of object level pixels (ones) in block represented binary images, it is seen that the OR operation is a union region operation and is easily implemented with blocks. Care has to be taken to avoid the existence of overlapping blocks in the results. The use of the SETDIF operation guarantees this requirement. The application of the OR operation between two blocks b_0 and b_1, is executed according to the following formula:

$$b_0 \text{ OR } b_1 = \text{SETDIF}(b_0, b_1) \cup b_1 \tag{4}$$

where the symbol of the set union $\cup$ means that the block b_1 is contained in the results.

C. The NOT operation

The NOT operation cannot be executed directly, since the image block representation carries only information concerning image regions with object level and not image regions belonging to the background. However, the NOT operation in block represented images is implemented by defining an image f_Θ with the same width and height as the input image that contains only one block Θ covering all the image. The execution of the operation SETDIF(Θ, f) results in the inverse of the image , i.e.

$$\text{NOT}(f) = \text{SETDIF}(f_\Theta, f) \tag{5}$$

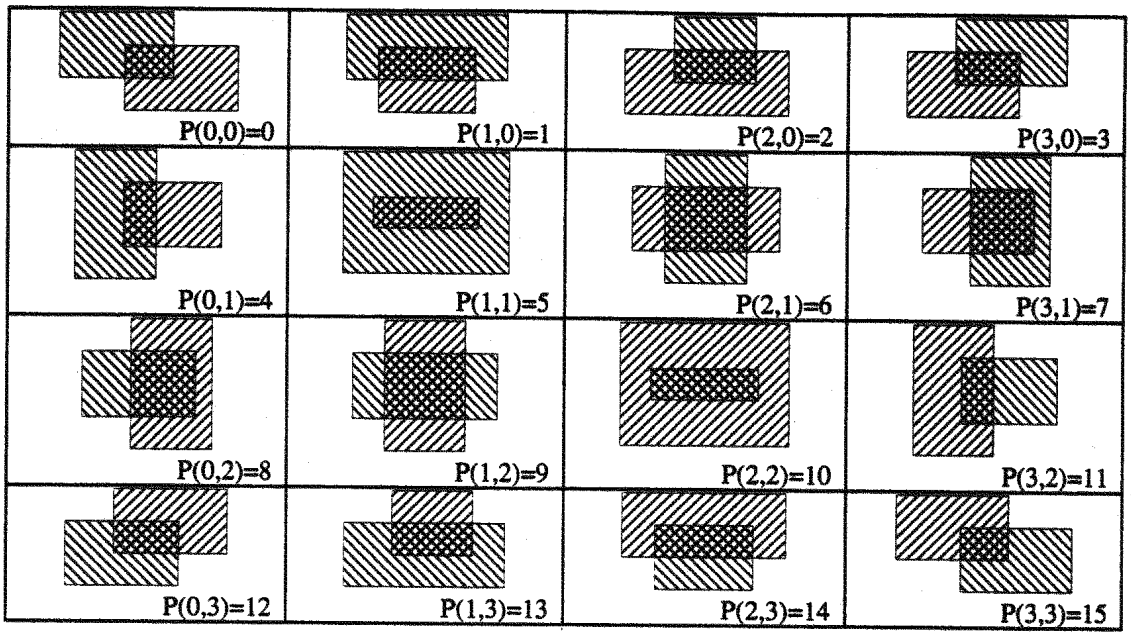

Figure 4. The relative positions of the blocks (shading) and (shading) according to a rectangular grid.

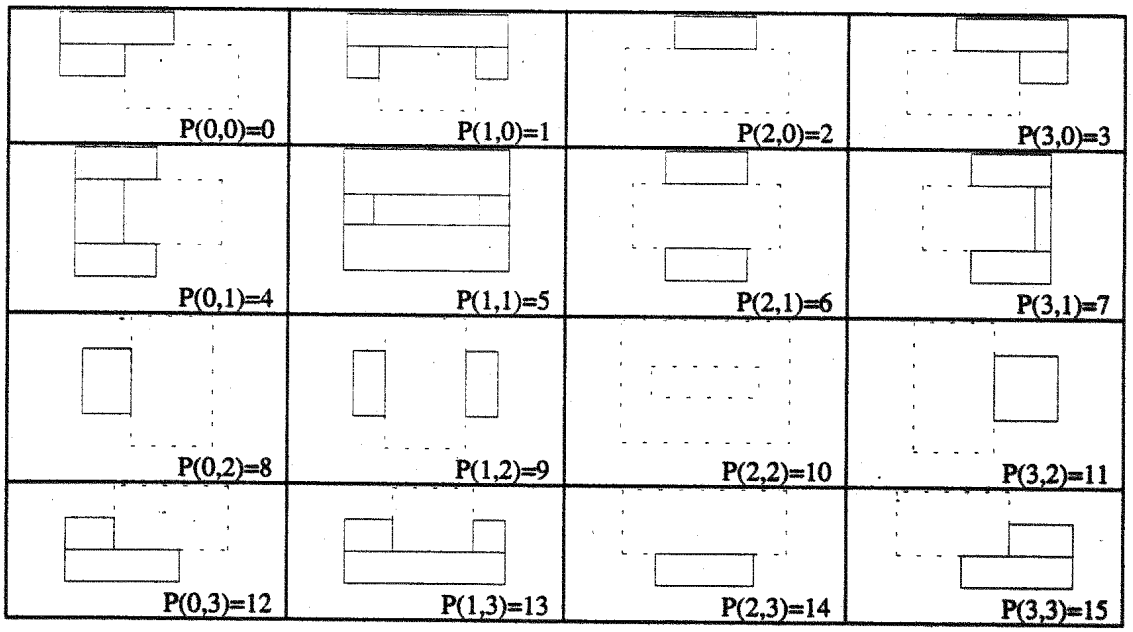

Figure 5. The blocks that result from the operation SETDIF (b_0, b_1) for each specific relative position of the blocks b_0, b_1.

Table 2 shows the required time for the execution of the logic operations, considering block represented images and 2-D array represented images respectively, for the test images of Fig. 3. The pixel based method, which has been used for the time comparisons of Table 2, consists of the execution of the logic operation between each pixel of the first image and the corresponding pixel of the second image. The fast implementation of the AND and XOR operations [10], [11] also permitted using block represented images.

Table 2. The required time for the execution of the logic operation OR among the binary images of the islands Corfu and Mikonos and the NOT operation for the image of the island Corfu, using the pixel and block based algorithms. The last column gives the time reduction factor when the image block representation based algorithm is used in comparison with the pixel based algorithm.

Logic Operation	Pixel based method	Block based method	Reduction factor
OR	2.308 sec	0.328 sec	7.0
NOT	1.408 sec	0.276 sec	5.1

4. Fast Parallel Skeletonization Algorithm

The following criteria should be valid, for any skeletonization algorithm:

(i) does not remove end points

(ii) does not break connectivity

(iii) does not cause excessive erosion of the region.

A skeletonization algorithm is sequential, if the deletion of a pixel in the nth iteration depends on the result of the $(n-1)$th iteration and on the pixels already processed in the nth iteration. A skeletonization algorithm is parallel, if the deletion of a pixel in the nth iteration depends only on the result of the $(n-1)$th iteration.

In the proposed algorithm each iteration is divided into four subiterations. In the first, second, third and fourth subiteration, the north, west, south and east pixels of the object, are removed respectively. The main advantage of the proposed algorithm is that it operates in blocks and therefore it permits the deletion of areas of pixels, instead of a single pixel.

Consider the pixel p and their 8 neighbors, as defined in Fig. 6. $B(p)$ is defined as the number of nonzero neighbors of p, i.e. $B(p) = \sum_{i=1}^{8} p_i$ and $A(p)$ is defined as the number of transitions of 01 patterns in the ordered set $p_1, p_2, \ldots, p_8$.

p_8	p_1	p_2
p_7	p	p_3
p_6	p_5	p_4

Figure 6. The neighborhood of the pixel p.

A. North subiteration

At first the neighboring blocks of the considered block are determined, sorted and placed in two lists; one for the upper neighbors and one for the lower neighbors. Different procedures are applied for those blocks that have unity width than those with greater width.

A.1. Blocks with unity width

For the blocks with unity width, the blocks that remain after the subiteration are described as:

$$R = N \cup \bar{S} \cup A \tag{6}$$

where N are the areas that have a north neighbor, S are those areas that have a south neighbor and A are those areas that have transitions from 01 patterns different from 1. If b is the considered block and $a_i, i = 1, \ldots, k$ are the north neighbor blocks of b, then

$$N = (\max(b_{x1}, a_{i,x1}), \min(b_{x2}, a_{i,x2})), i = 1, \ldots, k. \tag{7}$$

S is computed in a similar manner, and $\bar{S}$ is computed using logic operations in blocks, as described in [11].

The algorithm counts the transitions for the two extreme pixels of the considered block. For the middle pixels of the block, the algorithm recognizes the areas A as those that have at least one north and at least one south neighbor pixel.

A.2. Blocks with width greater than 1

For the blocks with width greater than 1, the algorithm determines the remaining areas as those that have a north neighbor block. The two extreme pixels, the upper left and the upper right of the block, are remaining pixels only if the number of transitions from 01 patterns is different than 1.

B. East subiteration

At first, the neighboring blocks of the considered block are determined, sorted and placed in two lists; one for the upper neighbors and one for the lower neighbors.

B.1. Blocks with unity width

In the case of a block with unity width, its right pixel p is a remaining pixel if the following two conditions are valid:

$B(p) = 1$ OR $B(p) = 7$

$A(p) \neq 1$

B.2. Blocks with width greater than 1

In the case of a block with width greater than 1, the algorithm deletes all the middle points of the left side of the block if the length of the block is greater than 1. For the two extreme pixels, the upper left and the lower left of the block, the criteria of case B1 are also applied.

The procedures for the south and west subiterations are quite similar with those of north and east subiterations, respectively. At the end of each subiteration the new formed blocks are the input to the next subiteration.

5. Examples

The algorithm is fast implemented, since it operates on image areas instead on single pixels. Table 3 demonstrates the required computational times for the execution of the skeletonization for the images of Fig. 3, using the well known Zhang and Suen [14] algorithm and the proposed algorithm that operates in blocks.

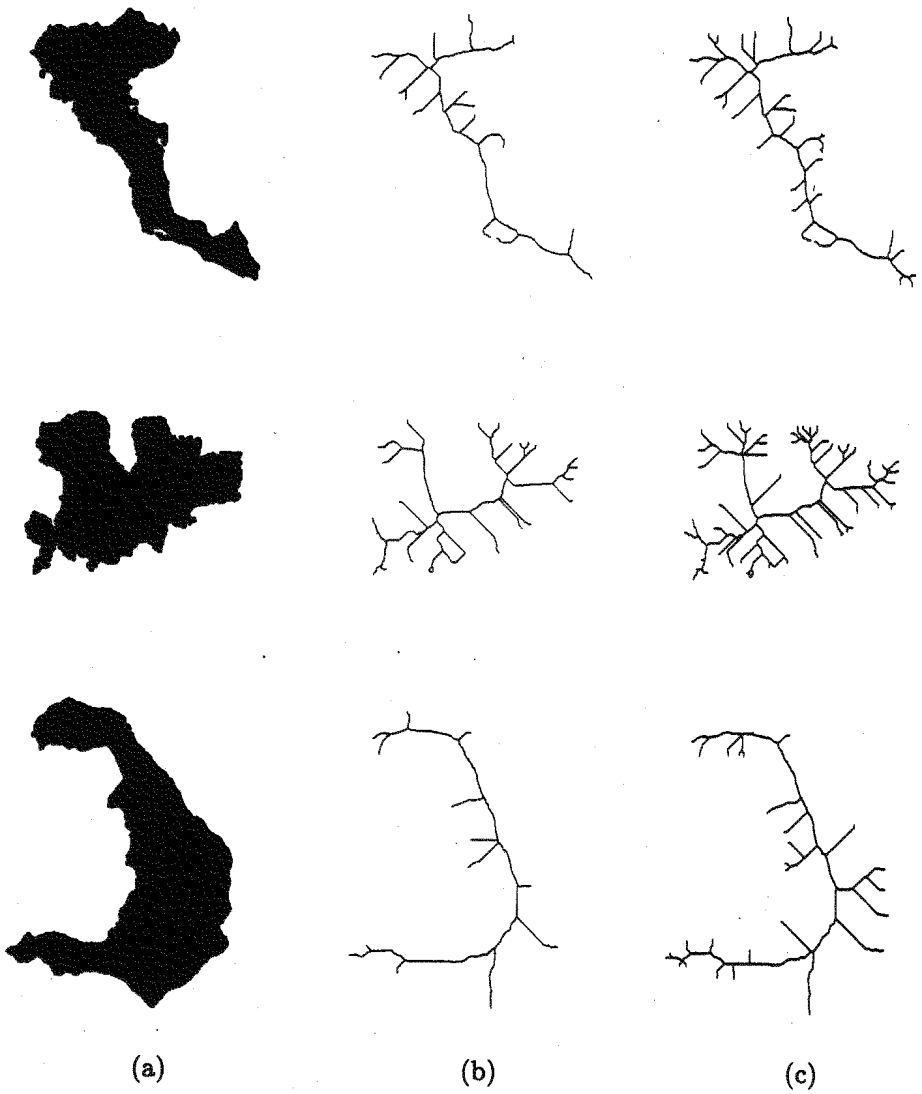

Figure 7. (a). Images of the islands Corfu, Mikonos and Santorini. (b) Skeletonization using the Zhang and Suen algorithm. (c). Skeletonization using the proposed algorithm.

In Fig. 8 some character images and their skeletons are shown. The skeletons which are produced using the proposed algorithm are almost identical with the skeletons produced using the Zhang and Suen algorithm, while the average time reduction factor is almost 6. A direct analysis concerning computational complexity of the propoced algorithm in comparison with the Zhang and Suen algorithm is not possible, since the time and space complexity is image dependent.

Table 3. The required computational times in seconds, for the execution of the skeletonization operation using the Zhang and Suen algorithm and the proposed algorithm. The third column is the reduction factor.

Image	Zhang - Suen	Blocks	Reduction factor
Corfu	12.1	2.9	4.1
Mikonos	14.6	2.8	5.2
Santorini	12.6	3.1	4.1

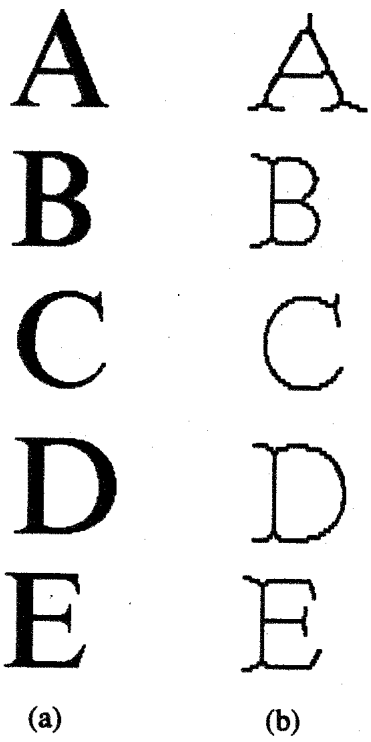

Figure 8. (a) Character images. (b) Their skeletons produced with the proposed algorithm.

6. Conclusions

In the recent years fast image processing and analysis algorithms that operate on block represented binary images have been presented in the literature: specifically the real-time computation of the statistical moments (both software [9] and hardware algorithms [12]), the fast implementation of image shift, image scale, image rotation, determination of the minimum and of the maximum distance from a point to an object, perimeter measurement, area measurement, logic operations, connectivity checking, object detection and edge extraction [11]. The proposed skeletonization algorithm may be considered as a parallel skeletonization algorithm that operates in a serial machine. This conclusion arises from the fact that the algorithm decides which are the pixels that should form a new block, or equivalently deletes simultaneously a number of boundary pixels for which the conditions of removing are valid.

References

[1] H. Samet, "The quadtree and related hierarchical data structures", *Computing Survey*, vol. 16, no. 2, pp. 187-260, 1984.

[2] J. Capon, "A propabilistic model for run-length coding of pictures", *IRE Trans. Information Theory*, vol. IT-5, no. 4, pp. 157-163, 1959.

[3] W.K. Pratt, *Digital Image Processing*, John Wiley & Sons, 2nd Edition, 1991.

[4] J. Piper, "Efficient implementation of skeletonisation using interval coding", *Pattern Recognition Letters*, vol. 3, pp. 389-397, 1985.

[5] H. Freeman, "Computer processing of line drawings", *ACM Computing Surveys*, vol. 6, pp. 57-97, 1974.

[6] D.W. Paglieroni and A.K. Jain, "Control point transforms for shape representation and measurement", *Computer Vision, Graphics and Image Processing*, vol. 42, pp. 87-111, 1988.

[7] R.L. Kashyap and R. Chellappa, "Stochastic models for closed boundary analysis: Representation and reconstruction", *IEEE Trans. Information Theory*, vol. IT-27, no. 5, pp. 627-637, 1981.

[8] I.M. Spiliotis and B.G. Mertzios, "Real-time computation of statistical moments on binary images using block representation", *Proceedings of the 4th International Workshop on Time-Varying Image Processing and Moving Object Recognition*, Florence, Italy, June 10-11, pp. 27-34, 1993.

[9] I.M. Spiliotis and B.G. Mertzios, "Real-time computation of two-dimensional moments on binary images using image block representation", *IEEE Transactions on Image Processing*. Accepted for publication.

[10] I.M. Spiliotis and B.G. Mertzios, "Logic operations on 3-dimensional volumes using block representation", *Proceedings of the International Workshop on Stereoscopic and Three Dimensional Imaging (IWS3DI'95)*, pp. 317-322, 6-8 September 1995, Fera, Santorini, Greece.

[11] I.M. Spiliotis and B.G. Mertzios, "Fast algorithms for basic processing and analysis operations on block represented binary images", *Pattern Recognition Letters*, vol.17, pp. 1437-1450, 1996.

[12] I.M. Spiliotis and B.G. Mertzios, "Real-Time Computation of 2-D Moments on Block Represented Binary Images on the Scan Line Array Processor", *Proc. of VIII European Signal Processing Conference (EUSIPCO-96)*, September 10-13, 1996, Trieste, Italy.

[13] H. Blum, "A transformation for extracting new descriptors on shape", in *Symposium on Models for Perception of Speech and Visual Form*, Weiant Whaten-Dunn, Ed., MIT Press, Cambridge, Mass., 1967.

[14] T.Y. Zhang and C.Y. Suen, "A fast parallel algorithm for thinning digital patterns". *Commun. ACM*, vol. 27, No. 3, March 1984, pp. 236-239.

[15] J.F. Jenq and S. Sanhi, "Serial and parallel algorithms for the medial axis transform", *IEEE Trans. on Pattern Anal., Machine Intel..*, Vol. 14, No. 12, Dec. 1992.

[16] A.Y. Wu, S.K. Bhaskar and A. Rosenfeld, "Parallel computation of geometric properties from the medial axis transform", *Computer Vision, Graphics and Image Proc.* , vol. 41, pp. v323-332, 1988.

[17] Y. Xia, "Skeletonization via the realization of the fire's front propagation and extinction in digital binary shapes", *IEEE Trans. on Pattern Anal., Machine Intel.*, Vol. 11, No. 10, pp. 1076-1086, 1989.

[18] O. Baruch, "Line thinning by line following", *Pattern Recognition Letters*, vol. 8, pp. 271-276, 1988.

[19] R. M. K. Sinha, "A width independent algorithm for character skeleton estimation", *Computer Vision, Graphics and Image Processing 40*, 388-397, 1987.

[20] L.Lam, S. - W. Lee and C.Y. Suen, "Thinning methodologies - a comprehensive survey". *IEEE Trans. PAMI*, vol. 14, No. 9, pp. 869-885, September 1992.

[21] M. McKenna, J. ORourje and S. Suri, "Finding the largest empty rectangle in an orthogonal polygon", *Proc. of the 23rd Annual Allerton Conf. on Communication, Control and Computing,* Urbana-Campaign, Illinois, 486-495, 1985.

[22] M. Orlowski, "A new algorithm for the largest empty rectangle problem", *Algorithmica,* 5, 65-73, 1990.

[23] H.S. Baird, S.E Jones. and S.J. Fortune, "Image segmentation by shape-directed covers", *Proc. 10th Intern. Conf. on Pattern Recognition,* Atlantic City, NJ, USA, 820-825, 1990.