



ELSEVIER

Pattern Recognition Letters 17 (1996) 1437–1450

Pattern Recognition
Letters

Fast algorithms for basic processing and analysis operations on block-represented binary images

Iraklis M. Spiliotis^{*}, Basil G. Mertzios[†]

Automatic Control Systems Laboratory, Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi 67100, Greece

Received 14 November 1995; revised 17 September 1996

Abstract

This paper describes a binary image representation scheme, called Image Block Representation and presents new algorithms for a number of basic binary image processing and analysis operations, which are rapidly 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.

Keywords: Image block representation; Image normalization; Image measurements; Logic operations

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 representation approaches 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 (Samet, 1984), run length encoding (Capon, 1959; Pratt, 1991) and interval coding representation (Piper, 1985). The boundary-based representations include chain code

(Freeman, 1974), contour control point models (Paglieroni and Jain, 1988) and autoregressive models (Kashyap and Chellappa, 1981). 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 (Spiliotis and Mertzios, 1993, 1996). 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 that 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 imple-

^{*} Corresponding author. E-mail: spiliot@demokritos.cc.duth.gr.

[†] E-mail: mertzios@demokritos.cc.duth.gr.

mentation of new algorithms for binary image processing and analysis tasks, leads to substantial reduction of the required computational complexity. The image block representation method is superior to run length coding and to quadtree representation.

This paper presents a number of basic image processing and analysis algorithms, which are usually used for preprocessing and feature extraction tasks before any recognition process. These algorithms are characterized by low time complexity and are suitable for very fast processing rates, due to the substitution of image pixels by blocks. Specifically, the operations of image shift, image scale, image rotation, determination of the minimum and of the maximum distance from a point to an object, edge extraction and perimeter measurement, area measurement, connected component labeling and logic operations are presented. In all the above operations the time complexity is significantly lower in comparison to the implementations that are based on the 2-D array, on the run length and on the quadtree representations of binary images.

The algorithms presented in this paper have been used primarily for the development of a vision system, capable of operating in a complex and rapidly varying environment. Such conditions exist in target detection and identification vision systems and also in moving vision systems, which are located on some kind of vehicle. In spite of the fact that a number of powerful parallel architectures are available for image processing and analysis operations, in the most of the real-world applications low-cost serial machines are usually used. From this point of view the presented implementations may be useful for a variety of tasks.

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 non-overlapping

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 non-overlapping blocks that completely cover the image areas with object level and it is denoted as

$$f(x, y) = \{b_i; i = 0, 1, \dots, k - 1\}, \quad (1)$$

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

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

where for simplicity it is assumed that $x_{1,b} \leq x_{2,b}$, and $y_{1,b} \leq y_{2,b}$, 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 an $N_1 \times N_2$ binary image $f(x, y)$, $x = 0, 1, \dots, N_1 - 1$, $y = 0, 1, \dots, N_2 - 1$, the block extraction process requires a pass from each line y of the image. In this pass all object level

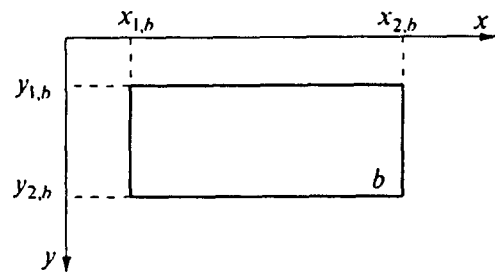


Fig. 1. Each block b is described by the coordinates of its two corner points.

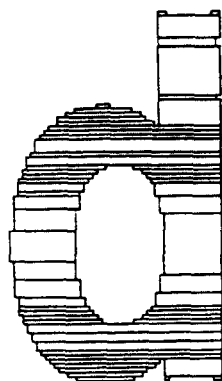


Fig. 2. Image of the character d and the blocks.

intervals are extracted and compared with the extracted blocks that have pixels in the previous line. The extracted blocks are denoted by b_i ; $i = 0, 1, \dots, k$, the indices of the extracted blocks that have pixels in the previous line at each stage are denoted by p_i ; $i = 0, 1, \dots, k_p$, and the indices of the extracted blocks that have pixels in the current line are denoted by c_i ; $i = 0, 1, \dots, k_c$. 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.

Algorithm 1 (Image Block Representation)

```

k := 0; k_p := 0;
for y := 0 to N_2 - 1 {
  k_c := 0;
  for x := 0 to N_1 - 1 {
    Find interval in row y with start x_1 and end x_2;
    if (interval is found) then {
      interval_matched := FALSE;
      for j := 0 to k_p - 1 {
        if (x_1 = x_{1,b_{p_j}} AND x_2 = x_{2,b_{p_j}}) then {
          c_{k_c} := p_j; y_{2,b_{p_j}} := y;
          interval_matched := TRUE;
        }
      }
      if NOT (interval_matched) then {
        x_{1,b_k} := x_1; x_{2,b_k} := x_2;
        y_{1,b_k} := y; y_{2,b_k} := y;
        c_{k_c} := k; k := k + 1;
      }
    }
  }
  k_c := k_c + 1;
}
    
```

```

}
}
for i := 0 to k_c - 1
  p_i := c_i;
  k_p := k_c;
}
    
```

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. In correspondence with the pixel connectivity schemes (Gonzalez and Wintz, 1987), two definitions concerning 4- and diagonal (D-) connectivity among the blocks are given.

Definition 3. Two blocks are defined as *4-connected*, if there exists a pair of 4-connected pixels, one from each block.

Definition 4. Two blocks are defined as *D-connected*, if there exists a pair of D-connected pixels one from each block, while there does not exist a pair of 4-connected pixels one from each block.

Figs. 3 and 4 show the 4-connectivity and the D-connectivity of a point and of a block, respectively. The connectivity among the image blocks may be determined during the image block representation process, or directly on a given block represented image, using the following criteria.

Lemma 1. Two blocks are 4-connected if their projections on one of the x or y axes are overlapped and their projections on the other axis are neighbors.

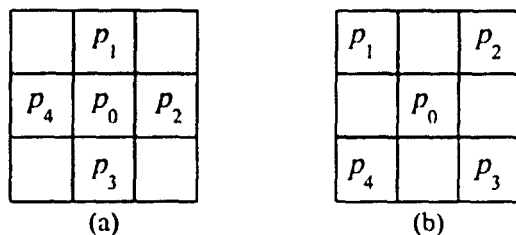


Fig. 3. Connectivity of the pixel p_0 . (a) The 4-connected pixels. (b) The D-connected pixels.

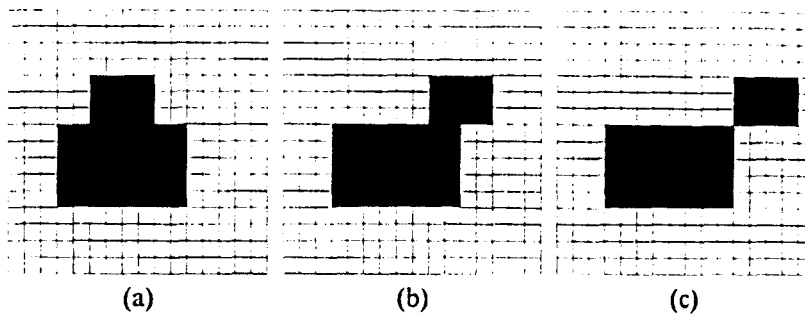


Fig. 4. Connectivity among the blocks. (a), (b) 4-connected blocks. (c) D-connected blocks.

Proof. Suppose that the projections of the two blocks on the x -axis are overlapped and that their projections on the y -axis are neighbors. Then there exists a pair of pixels, one from each block, which are located at the same image column and at a neighboring image row, i.e. they are 4-connected and according to Definition 3 the two blocks are 4-connected. The proof for the case of two blocks with overlapped projections on the y -axis and neighbor projections on the x -axis is analogous. \square

Lemma 2. *Two blocks are D-connected if their projections on both axes are neighbors.*

Proof. Then there exists a pair of pixels one from each block, which are located on neighboring image rows and columns, i.e. are D-connected. A pair of 4-connected pixels one from each block does not exist, since a pair of pixels one from each block, which are located at the same image row or column,

does not exist. Then according to Definition 4 the two blocks are D-connected. \square

The information concerning block connectivity (either 4- and D-connectivity) requires a suitable data structure for storage. Therefore, each block b_i is represented as the ordering:

$$b_i = (x_{1,b_i}, x_{2,b_i}, y_{1,b_i}, y_{2,b_i}, nc_i, c_i), \quad (3)$$

where x_{1,b_i}, x_{2,b_i} are the horizontal coordinates of the i th block, y_{1,b_i}, y_{2,b_i} are the vertical coordinates of the block, nc_i is the number of the connected blocks and c_i is a list with the indices of these connected blocks.

3. Comparison with other region based representations

In this section some comparisons with other region-based binary image representation schemes and

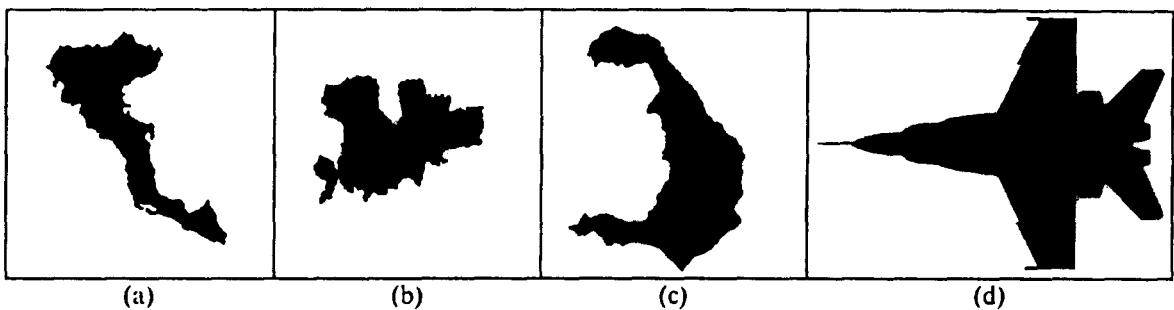


Fig. 5. A set of test images. (a) Image of the island Corfu of 512×512 pixels. (b) Image of the island Mikonos of 512×512 pixels. (c) Image of the island Santorini of 512×512 pixels. (d) Aircraft image of 512×697 pixels.

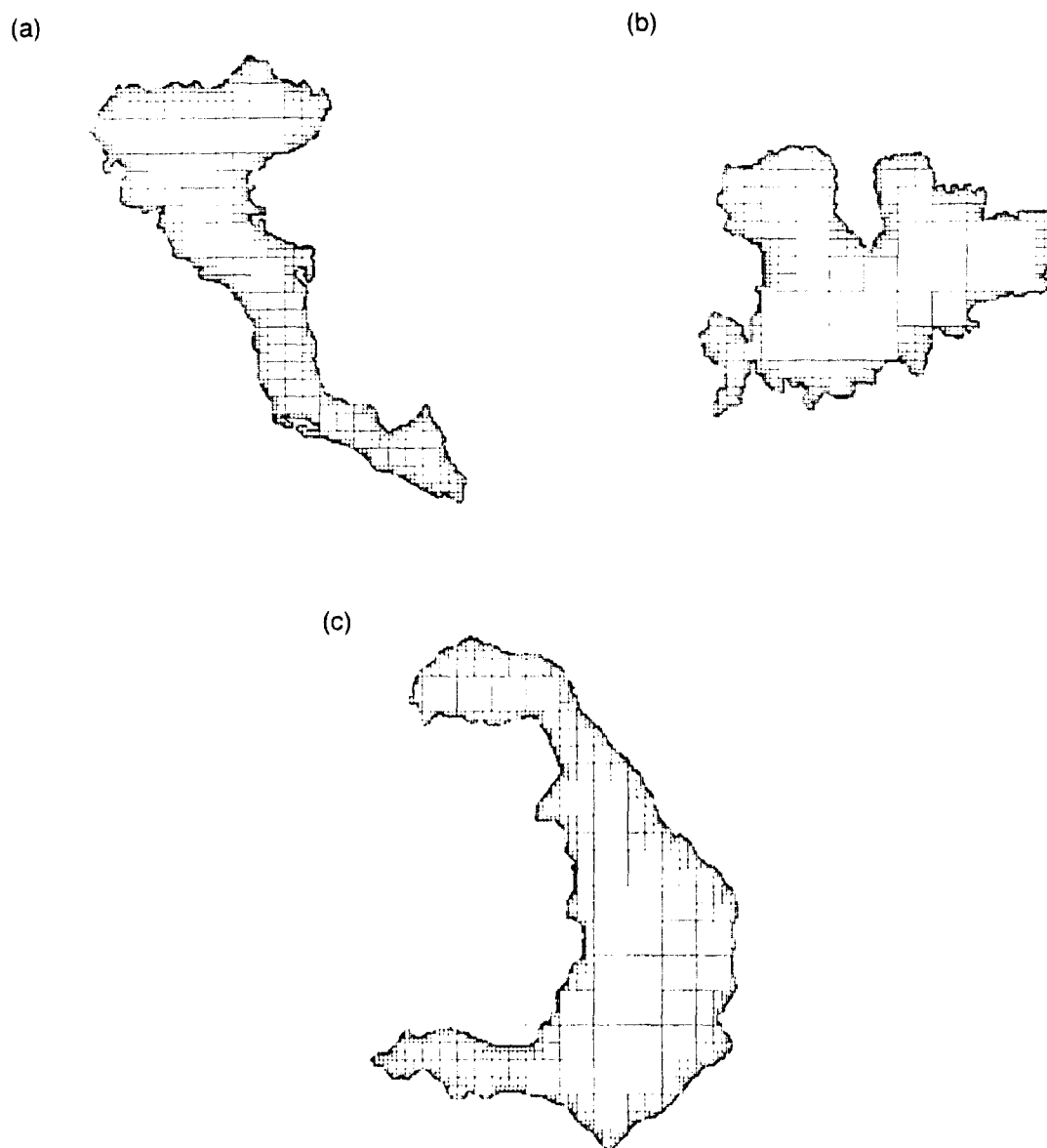


Fig. 6. The quadtree representation of the images (a) of the island Corfu, (b) of the island Mikonos, and (c) of the island Santorini.

specifically with the run length coding and the quadtree representation are presented.

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 the 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.

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 Fig. 5

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
Aircraft	118831	494	397	0.879	656	0.849		

In Fig. 5 four 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. The quadtree representation for the aircraft image is not given, since it is not a square image. 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, 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. To clarify the above point, in Figs. 6(a), (b) and (c) the representations with quadtrees of the test images of the islands Corfu, Mikonos and Santorini are demonstrated. 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 (Baird et al., 1990; McKenna et al., 1985; Orłowski, 1990) 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 minimum number of blocks. It is pointed out that, given a specific binary image, different sets of different blocks can be formed. Actually, the non-unique block representation does not have any implications on the implementation of any operation on a block represented image.

In the following sections some basic processing and analysis operations, which exploit the fact of machine perception of image areas in block-represented images are described.

4. Image normalization

The operations of image shift, image scale and image rotation are presented in this section. These

operations are usually used for image and object normalization, before any feature extraction process.

4.1. Image shift

Let $f(x, y)$ be the original binary image and $f'(x, y)$ the resulted binary image after the application of the operation $\text{SHIFT}(m, n)$. The operation $\text{SHIFT}(m, n)$ results to a 2-D shift of m horizontal units and n vertical units, of each pixel (x, y) of the original image and is defined by the relation

$$f'(x, y) = \text{SHIFT}(m, n)[f(x, y)] \\ = f(x - m, y - n). \quad (4)$$

Using block-represented binary images, the shift operation is performed rapidly by shifting each block of the image. Specifically, if the image $f(x, y)$ is represented by k blocks, as is described in (1), then the application of the operation $\text{SHIFT}(m, n)$ on $f(x, y)$ results in the image $f'(x, y)$, which is represented also by k blocks, i.e.,

$$f'(x, y) = \{b'_0, b'_1, \dots, b'_{k-1}\}, \quad (5)$$

where the i th block is described as

$$b'_i = (x_{1,b'_i}, x_{2,b'_i}, y_{1,b'_i}, y_{2,b'_i}) \\ = (m + x_{1,b}, m + x_{2,b}, n + y_{1,b}, n + y_{2,b}), \\ \forall i \in [0, k - 1]. \quad (6)$$

4.2. Image scale

Consider a block $b = (x_{1,b}, x_{2,b}, y_{1,b}, y_{2,b})$ in an image f . The application of the scale operation on the image f by a factor a transforms the block b to the block $b' = (x_{1,b'}, x_{2,b'}, y_{1,b'}, y_{2,b'})$, where

$$x_{1,b'} = ax_{1,b}, \quad y_{1,b'} = ay_{1,b}, \\ x_{2,b'} = ax_{2,b} + a - 1, \quad y_{2,b'} = ay_{2,b} + a - 1. \quad (7)$$

The use of (7) guarantees that the width w' and the height h' of the block b' , where $w' = x_{2,b'} - x_{1,b'} + 1$ and $h' = y_{2,b'} - y_{1,b'} + 1$, are scaled properly, i.e. $w' = aw$ and $h' = ah$, where w is the width and h the height of the block b . If a is not an integer then the calculated values from (7) are truncated to the next lower integer.

In image enlargement ($a > 1$), the scaled image f' is described by the same number of blocks as f , where the coordinates of each scaled block are computed using (7).

In image subsampling ($a < 1$), there is always loss of information. Suppose, for example, that $a = 1/3$. This means that one pixel of the image f' corresponds to nine pixels of the image f . In the proposed method on block-represented images there is also loss of information, as in the scale methods that are based on pixels. Suppose that the image f is comprised by k blocks. Eq. (7) should be used to compute the coordinates of the k scaled blocks. However, due to the information loss, the k scaled blocks are overlapped and some of them should be changed or eliminated. For this purpose, after the computation of the coordinates of a block, a procedure is applied to check if the new block is partially or totally covered by a previous scaled block. In the case of partial overlapping, the coordinates of the new scaled block are changed in such a way that the new block starts or ends at the free pixel that is next to the previous scaled block. In the case of total overlapping, the overlapped (the smaller) block is eliminated. Therefore, if the new image f' should be smaller than the original image f , then it is possible that the number of the blocks of f' is smaller than the number of the blocks of f , due to the inability to operate in subpixel accuracy.

4.3. Image rotation

Consider a point P with coordinates (x_p, y_p) on the image plane. Under image rotation by an angle θ , the position (x'_p, y'_p) of the point P in the new coordinate system, is given by

$$x'_p = x_p \cos \theta - y_p \sin \theta, \\ y'_p = x_p \sin \theta + y_p \cos \theta. \quad (8)$$

Now consider the block b with coordinates x_1, x_2 with respect to the horizontal axis and coordinates y_1, y_2 with respect to the vertical axis, whose corner points are denoted by P_1, P_2, P_3 and P_4 . Under image rotation by an angle θ , the new positions P'_1, P'_2, P'_3, P'_4 of the corner points may be found using (8). Each other point of the block retains its

Table 2

The required time for the execution of the image rotation, edge extraction and area measurement operations for a set of test images. The last column on each operation is the time reduction factor using the image block representation-based algorithm, in comparison with the pixel-based algorithm

Image	Image rotation			Edge extraction			Area measurement		
	Pixel method (sec)	Block method (sec)	Reduction factor	Pixel method (sec)	Block method (sec)	Reduction factor	Pixel method (sec)	Block method (sec)	Reduction factor
Island Corfu	0.989	0.009	109	0.276	0.038	7.2	0.330	0.001	330
Island Mikonos	1.099	0.012	91	0.282	0.037	7.6	0.330	0.001	330
Island Santorini	1.208	0.011	109	0.356	0.059	6.0	0.390	0.001	330
Aircraft	2.032	0.021	96	0.594	0.112	5.3	0.549	0.001	549

relative position according to the corner points and therefore the block is transformed to a rectangle with corner points P'_1, P'_2, P'_3, P'_4 . Using region filling techniques, all the points of the resultant rectangle are set to the value 1, since they are object points. The rotated image is in 2-D form and therefore a second extraction is required in order to revert in block represented form. Table 2 shows the required time for the execution of the image rotation operation, considering block-represented images and 2-D array-represented images respectively, for the set of the test images shown in Fig. 5. The pixel-based method, which has been used for the time comparisons of Table 2, consists in the computation of the coordinates in the rotated image of each object pixel of the original image.

5. Image measurements

5.1. Minimum and maximum distance from a point to an object

The minimum and the maximum distance between a point and an object can be computed using block-represented images. First the minimum distance d_{\min} , and the maximum distance d_{\max} , between a point $P(x_p, y_p)$ and the block $b_i = (x_{1,b_i}, x_{2,b_i}, y_{1,b_i}, y_{2,b_i})$ are determined.

If $x_p \in [x_{1,b_i}, x_{2,b_i}]$, then these distances are given by

$$\begin{aligned} d_{\min} &= \min\{|y_p - y_{1,b_i}|, |y_p - y_{2,b_i}|\}, \\ d_{\max} &= \max\{|y_p - y_{1,b_i}|, |y_p - y_{2,b_i}|\}, \end{aligned} \quad (9)$$

and if $y_p \in [y_{1,b_i}, y_{2,b_i}]$, then these distances are given by

$$\begin{aligned} d_{\min} &= \min\{|x_p - x_{1,b_i}|, |x_p - x_{2,b_i}|\}, \\ d_{\max} &= \max\{|x_p - x_{1,b_i}|, |x_p - x_{2,b_i}|\}. \end{aligned} \quad (10)$$

If the conditions $x_p \in [x_{1,b_i}, x_{2,b_i}]$ and $y_p \in [y_{1,b_i}, y_{2,b_i}]$ are both valid for the specific block b_i , then the point P belongs to that block. Therefore it is a point of the object and the minimum distance is 0.

Otherwise, if neither of the two conditions is valid then the minimum and the maximum distance are given by

$$\begin{aligned} d_{\min} &= \left(\min\{|x_p - x_{1,b_i}|, |x_p - x_{2,b_i}|\}^2 \right. \\ &\quad \left. + \min\{|y_p - y_{1,b_i}|, |y_p - y_{2,b_i}|\}^2 \right)^{1/2}, \\ d_{\max} &= \left(\max\{|x_p - x_{1,b_i}|, |x_p - x_{2,b_i}|\}^2 \right. \\ &\quad \left. + \max\{|y_p - y_{1,b_i}|, |y_p - y_{2,b_i}|\}^2 \right)^{1/2}. \end{aligned} \quad (11)$$

The computations involved in the above analytical formulae are executed rapidly. To find the minimum distance and the maximum distance, d_{\min} and d_{\max} , between the object and the point P , it is sufficient to calculate the minimum of the distances d_{\min} , and the maximum of the distances d_{\max} , among the point P and all the k blocks b_i , $i = 0, 1, \dots, k-1$, that constitute the object. Therefore,

$$\begin{aligned} d_{\min} &= \min\{d_{\min_0}, d_{\min_1}, \dots, d_{\min_{k-1}}\}, \\ d_{\max} &= \max\{d_{\max_0}, d_{\max_1}, \dots, d_{\max_{k-1}}\}. \end{aligned} \quad (12)$$

5.2. Edge extraction and perimeter measurement

Each point of the image with object level and with at least one 4-connected neighbor with background level, belongs to the edge of the object and is called an *edge point*. Using this definition, the edge extraction operation on binary images, which are represented by a 2-D array, is executed in one pass from each image pixel. Consider the block b , with coordinates $(x_{1,b}, x_{2,b}, y_{1,b}, y_{2,b})$, which is 4-connected with m blocks $b_j = (x_{1,b_j}, x_{2,b_j}, y_{1,b_j}, y_{2,b_j})$, $j = 1, 2, \dots, m$. The contribution of the block b to the edge points of the whole object are all the boundary points of the block b , except of those boundary points that are 4-connected neighbors with the points of the m connected blocks b_j . Care should be taken if the block b has unity width, where a point is possible to be connected from the one side of the block but not connected from the other side of the block. In this latter case, the specific point is retained as an edge point. The required time for the execution of the edge extraction operation for the set of the test images of Fig. 5, using the pixel checking process and the block representation-based algorithm, is given in Table 2. The pixel-based method, which has been used for the time comparisons of Table 2, is based on the checking of the four neighbors of each pixel with object level of the original image. The measurement of the perimeter is simply the count of the edge pixels. It is noted that the perimeter is not corrected for the diagonal elements of the object contour.

5.3. Area measurement

Since the image pixels with level 1 belong to the image blocks, the area A of the object is expressed as the summation of the areas A_i of the blocks b_i , as follows:

$$\begin{aligned} A &= \sum_{i=0}^{k-1} A_i \\ &= \sum_{i=0}^{k-1} (x_{2,b_i} - x_{1,b_i} + 1)(y_{2,b_i} - y_{1,b_i} + 1). \end{aligned} \quad (13)$$

The fast computation of the area A is achieved with the above simple and analytical formula. The

required time for the execution of the area measurement operation, for the set of the test images of Fig. 5 using the pixel counting and the summation of the areas of the blocks, is given in Table 2.

The area of an object can be expressed in terms of statistical moments as the moment of order $(0, 0)$. Using similar analysis, real-time (i.e. video rate) computation of the statistical moments is also achieved as the summation of the moments of the blocks (Spiliotis and Mertzios, 1993, 1996). Specifically, if the rectangular form appearing within the blocks is taken into account, then the geometrical moments of one block b , with coordinates $x_{1,b}, x_{2,b}, y_{1,b}, y_{2,b}$, are given by

$$\begin{aligned} m_{pq} &= \sum_{x=x_{1,b}}^{x_{2,b}} \sum_{y=y_{1,b}}^{y_{2,b}} x^p y^q = x_{1,b}^p \sum_{y=y_{1,b}}^{y_{2,b}} y^q \\ &\quad + (x_{1,b} + 1)^p \sum_{y=y_{1,b}}^{y_{2,b}} y^q + \dots + x_{2,b}^p \sum_{y=y_{1,b}}^{y_{2,b}} y^q \\ &= \left(\sum_{x=x_{1,b}}^{x_{2,b}} x^p \right) \left(\sum_{y=y_{1,b}}^{y_{2,b}} y^q \right). \end{aligned} \quad (14)$$

Moreover, the summations of powers of x and y in (14) are computed using analytical formulae based on the known formulae for the summation of the terms $1^m + 2^m + \dots + n^m$. It has also been shown that other sets of moments are computed in real-time on block-represented images.

5.4. Connected component labeling

The region connectivity and object detection is performed using a connected component labeling procedure (Rosenfeld and Kak, 1982), where the scheme of the connectivity among the blocks (see Section 2) is used. The procedure results in a number of components, each one of which holds the blocks of a distinct image object. The time complexity of component labeling is $O(N^2)$ in the case of an $N \times N$ image represented by a 2-D array and $O(k)$ in the case of an image represented by k blocks. Obviously, k is less than N^2 and this procedure is very efficient in block-represented images.

6. Logic operations

Logic operations are performed in block-represented binary images. Since each image is represented by a set of non-overlapping 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 similar concise work concerning logic operations on 3-D volumes has been described (Spiliotis and Mertzios, 1995).

6.1. 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. 7 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.

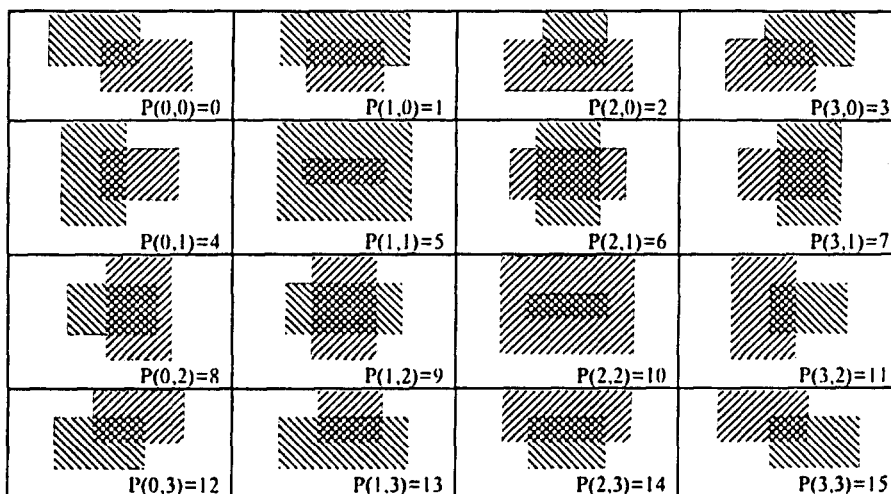


Fig. 7. The relative positions of the blocks b_0 (▨ shading) and b_1 (▩ shading) according to a rectangular grid.

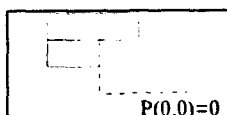

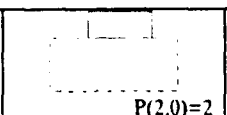
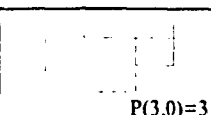
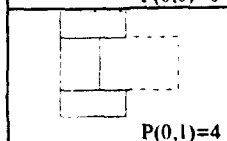
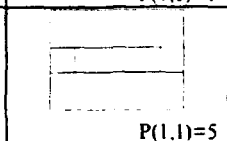
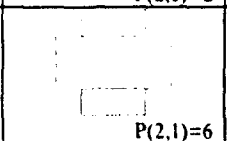
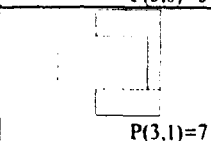
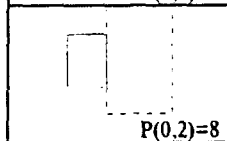
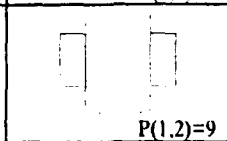
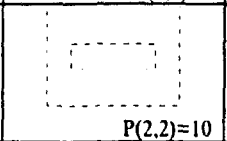
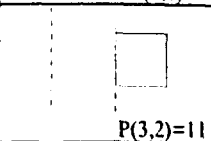
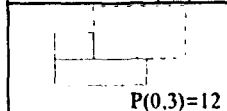
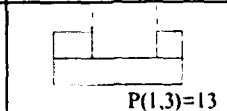
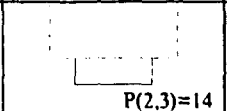
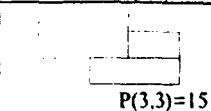
 P(0,0)=0	 P(1,0)=1	 P(2,0)=2	 P(3,0)=3
 P(0,1)=4	 P(1,1)=5	 P(2,1)=6	 P(3,1)=7
 P(0,2)=8	 P(1,2)=9	 P(2,2)=10	 P(3,2)=11
 P(0,3)=12	 P(1,3)=13	 P(2,3)=14	 P(3,3)=15

Fig. 8. The blocks that result from the operation $SETDIF(b_0, b_1)$ for each specific relative position of the blocks b_0, b_1 .

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 result blocks are formulated for each value of the function P , as shown in Fig. 8. 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.

Operation SETDIF between two images. Consider the implementation of the $SETDIF(f_1, f_2)$ operation between two block-represented images f_1 and f_2 .

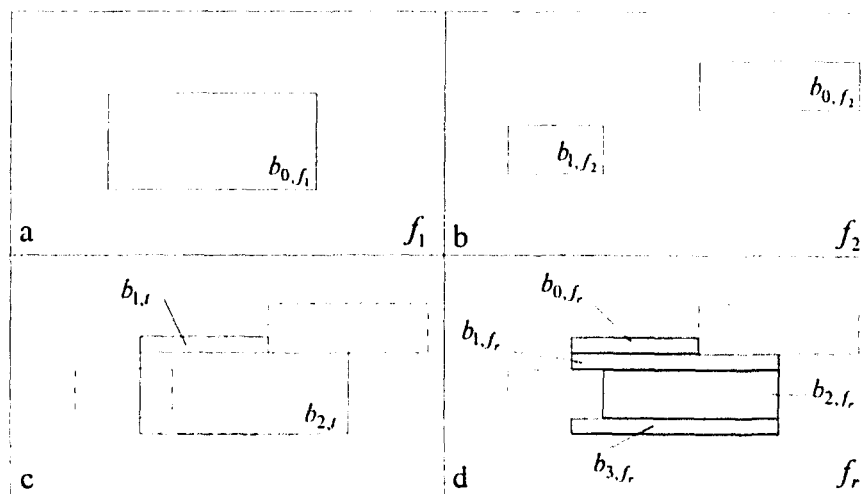


Fig. 9. Implementation of the operation $SETDIF(f_1, f_2)$. (a) The image f_1 . (b) The image f_2 . (c) The execution of the $SETDIF(b_{0,f_1}, b_{0,f_2})$. (d) The execution of the $SETDIF(b_{2,f_1}, b_{1,f_2})$ and the final result.

Due to the nature of the SETDIF operation, the following point has to be taken into account. A number of sub-blocks is derived when the execution of the operation SETDIF is applied between a block of the first image and the first block of the second image. The execution of the SETDIF operation between these sub-blocks and the next block of the second image results in new sub-blocks, etc. Therefore, it is appropriate to formulate temporary blocks $b_{m,i}$: $m = 0, 1, \dots, k_i - 1$, in order to store these sub-blocks until the execution of the SETDIF operation with all the blocks of the second image is terminated. After the execution of the operation SETDIF between a temporary block $b_{m,i}$ and a block of the second image, the block $b_{m,i}$ is eliminated and the resulted sub-blocks are placed in the temporary blocks. The elimination of a temporary block is simply achieved by setting a coordinate to a non-acceptable value, i.e. by setting $x_{1,b_{m,i}}$ to the value -1 . Finally, the temporary blocks are placed in the resulting image $f_r = \{b_{i,f_r}; i = 0, 1, \dots, k_r - 1\}$. This procedure is repeated for each block of the first image.

In the sequel, the algorithm for the implementation of the SETDIF operation between block-represented images and an example illustrating the algorithm are given.

Algorithm 2 (Implementation of the SETDIF(f_1, f_2) operation)

```

 $k_r := 0;$ 
for  $i := 0$  to  $k_1 - 1$  {
   $b_{0,i} := b_{i,f_1};$ 
   $k_i := 1;$ 
  for  $j := 0$  to  $k_2 - 1$  {
     $\hat{k}_i := k_i;$ 
    for  $m := 0$  to  $\hat{k}_i - 1$  {
      if  $x_{1,b_{m,i}} = -1$  then continue;
      Execute the operation
      SETDIF( $b_{m,i}, b_{j,f_2}$ ). Place the  $n$  resulting blocks in
      the set of temporary blocks starting from the position
       $k_i;$ 
       $k_i := k_i + n;$ 
       $x_{1,b_{m,i}} := -1;$ 
    }
  }
}

```

```

}
}
for  $m := 0$  to  $\hat{k}_i - 1$  {
  if  $x_{1,b_{m,i}} = -1$  then continue;
   $b_{k_r,f_r} := b_{m,i};$ 
   $k_r := k_r + 1;$ 
}
}

```

Example 1. Consider the block-represented binary images $f_1 = \{b_{0,f_1}\}$ and $f_2 = \{b_{0,f_2}, b_{1,f_2}\}$ of Fig. 9. For the execution of the function SETDIF(f_1, f_2), the block b_{0,f_1} is copied to the temporary blocks, $b_{0,i} = b_{0,f_1}$. The execution of the function SETDIF($b_{0,i}, b_{0,f_2}$) results in the temporary blocks $b_{1,i}, b_{2,i}$ and in the elimination of the block $b_{0,i}$. The blocks $b_{1,i}$ and b_{1,f_2} do not have common pixels, therefore the SETDIF($b_{1,i}, b_{1,f_2}$) is not executed. The execution of the SETDIF($b_{2,i}, b_{1,f_2}$) results in the temporary blocks $b_{3,i}, b_{4,i}, b_{5,i}$ and in the elimination of the block $b_{2,i}$. Finally, the temporary blocks are placed into the resulting blocks and the resulting image f_r is comprised of the blocks $\{b_{0,f_r}, b_{1,f_r}, b_{2,f_r}, b_{3,f_r}\}$, where $b_{0,f_r} = b_{1,i}, b_{1,f_r} = b_{3,i}, b_{2,f_r} = b_{4,i}$ and $b_{3,f_r} = b_{5,i}$.

6.2. 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 in block-represented images. 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 and two images f_1 and f_2 , is executed according to the following formula:

$$\begin{aligned}
 b_0 \text{ OR } b_1 &= \text{SETDIF}(b_0, b_1) \cup b_1, \\
 f_1 \text{ OR } f_2 &= \text{SETDIF}(f_1, f_2) \cup f_2,
 \end{aligned} \tag{15}$$

where the symbol of set union, \cup , means that the block b_1 and the image f_2 are contained in the results.

6.3. The AND operation

The AND operation between the block represented images f_1 and f_2 is implemented according to the formula

$$f_1 \text{ AND } f_2 = \bigcup_{i \in [0, k_1 - 1], j \in [0, k_2 - 1]} (b_{i, f_1} \text{ AND } b_{j, f_2}), \quad (16)$$

where b_{m, f_n} denotes the m th block of the n th image. The operation AND between the two blocks results in a new block, defined by its coordinates as follows:

$$b_{i, f_1} \text{ AND } b_{j, f_2} = \left(\begin{array}{l} \max(x_{1, b_{i, f_1}}, x_{1, b_{j, f_2}}), \\ \min(x_{2, b_{i, f_1}}, x_{2, b_{j, f_2}}), \\ \max(y_{1, b_{i, f_1}}, y_{1, b_{j, f_2}}), \\ \min(y_{2, b_{i, f_1}}, y_{2, b_{j, f_2}}) \end{array} \right). \quad (17)$$

However, the AND operation between block-represented binary images may result in adjacent blocks with exactly the same width. To this end, each one of the resulting blocks should be checked with all the others, in order to determine pairs of blocks that can be merged.

6.4. The XOR operation

The SETDIF operation is used for the execution of the XOR operation in block-represented images. The XOR operation produces the non-common points of the two blocks, or equivalently of the two images.

The application of the XOR operation between two blocks b_0 and b_1 and two images f_1 and f_2 , is executed according to the following formula:

$$\begin{aligned} b_0 \text{ XOR } b_1 &= \text{SETDIF}(b_0, b_1) \cup \text{SETDIF}(b_1, b_0), \\ f_1 \text{ XOR } f_2 &= \text{SETDIF}(f_1, f_2) \cup \text{SETDIF}(f_2, f_1). \end{aligned} \quad (18)$$

6.5. 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 back-

Table 3

The required time for the execution of the logic operations OR, AND and XOR between 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
AND	2.198 sec	0.272 sec	8.1
XOR	2.308 sec	0.549 sec	4.2
NOT	1.408 sec	0.276 sec	5.1

ground. 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 f , i.e.,

$$\text{NOT}(f) = \text{SETDIF}(f_\Theta, f). \quad (19)$$

Table 3 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 the islands Corfu and Mikonos of Fig. 5. The pixel-based method, which has been used for the time comparisons of Table 3, consists of the execution of the logic operation between each pixel of the first image and the corresponding pixel of the second image. The relevant algorithms can be directly applied in cases where logic operations should be executed recursively, such as in morphological skeletons, in opening and closing mathematical morphology operations and in open-close filters (Serra, 1982; Maragos and Schafer, 1990).

7. Conclusions

The image block representation may be seen as a physical model for the representation of binary images. Each block is represented by four integers, the coordinates of the upper left and lower right corner in vertical and horizontal axes. The image block representation is an information lossless process and,

therefore, from an information theory perspective, it is equivalent to the 2-D array image representation. Moreover, the image block representation is advantageous for image modelling. Usually, a block-represented binary image requires considerably less storage space (Table 1) and therefore it is characterized by less entropy. The main advantage of the algorithms presented is that the complexity is independent of the image size, in contrast to implementations that are based on the 2-D array image representation, in the sense that the number of blocks is unchanged under image magnification. This holds also in both the interval coding and the quadtree representations.

References

- Baird, H.S., S.E. Jones and S.J. Fortune (1990). Image segmentation by shape-directed covers. *Proc. 10th Internat. Conf. on Pattern Recognition*, Atlantic City, NJ, 820–825.
- Capon, J. (1959). A probabilistic model for run-length coding of pictures. *IRE Trans. Inform. Theory* 5, 157–163.
- Freeman, H. (1974). Computer processing of line drawings. *ACM Comput. Surveys* 6, 57–97.
- Gonzalez, R.C. and P. Wintz (1987). *Digital Image Processing*, 2nd edition. Addison-Wesley, Reading, MA.
- Kashyap, R.L. and R. Chellappa (1981). Stochastic models for closed boundary analysis: Representation and reconstruction. *IEEE Trans. Inform. Theory* 27, 627–637.
- Maragos, P. and R.W. Schafer (1990). Morphological systems for multidimensional signal processing. *Proc. IEEE* 78, 690–710.
- McKenna, M., J. O'Rourje and S. Suri (1985). Finding the largest empty rectangle in an orthogonal polygon. *Proc. 23rd Annual Allerton Conf. on Communication, Control and Computing*, Urbana-Campaign, IL, 486–495.
- Orlowski, M. (1990). A new algorithm for the largest empty rectangle problem. *Algorithmica* 5, 65–73.
- Paglieroni, D.W. and A.K. Jain (1988). Control point transforms for shape representation and measurement. *Computer Vision, Graphics and Image Processing* 42, 87–111.
- Pavlidis, T. (1982). *Algorithms for Graphics and Image Processing*. Computer Science Press, Rockville, MD.
- Piper, J. (1985). Efficient implementation of skeletonisation using interval coding. *Pattern Recognition Lett.* 3 (6), 389–397.
- Pratt, W.K. (1991). *Digital Image Processing*, 2nd edition. Wiley, New York.
- Rosenfeld, A. and A.C. Kak (1982). *Digital Picture Processing*, Vol. 2, 2nd edition. Academic Press, New York.
- Samet, H. (1984). The quadtree and related hierarchical data structures. *Computing Survey* 16, 187–260.
- Schalkoff, R.J. (1989). *Digital Image Processing and Computer Vision*. Wiley, New York.
- Serra, J. (1982). *Image Analysis and Mathematical Morphology*. Academic Press, New York.
- Spiliotis, I.M. and B.G. Mertzios (1993). Real-time computation of statistical moments on binary images using block representation. *Proc. 4th Internat. Workshop on Time-Varying Image Processing and Moving Object Recognition*, Florence, Italy, 27–34.
- Spiliotis, I.M. and B.G. Mertzios (1995). Logic operations on 3-dimensional volumes using block representation. *Proc. Internat. Workshop on Stereoscopic and Three-Dimensional Imaging (IWS3DI'95)*, 6–8 September 1995, Santorini, Greece, 317–322.
- Spiliotis, I.M. and B.G. Mertzios (1996). Real-time computation of two-dimensional moments on binary images using image block representation. *IEEE Trans. Image Process.*, accepted for publication.