International Journal of Image and Graphics
Vol. 4, No. 2 (2004) 1–21

World Scientific
www.worldscientific.com

FAST IMPLEMENTATION OF MORPHOLOGICAL OPERATIONS USING BINARY IMAGE BLOCK DECOMPOSITION

BASILIOS GATOS* and STAVROS J. PERANTONIS†

Computational Intelligence Laboratory, Institute of Informatics and Telecommunications, National Research Center "Demokritos", 153 10 Athens, Greece
**bgat@iit.demokritos.gr*
†*sper@iit.demokritos.gr*

NIKOS PAPAMARKOS‡ and IOANNIS ANDREADIS§

Department of Electrical and Computer Engineering, Democritus University of Thrace, 67 100 Xanthi, Greece
‡*papamark@ee.duth.gr*
§*iandread@ee.duth.gr*

Received 25 April 2002
Revised 3 July 2003
Accepted 27 April 2003

Morphological transformations are commonly used to perform a variety of image processing tasks. However, morphological operations are time-consuming procedures since they involve ordering and min/max computation of numbers resulting from image interaction with structuring elements. This paper presents a new method that can be used to speed up basic morphological operations for binary images. To achieve this, the binary images are first decomposed in a set of non-overlapping rectangular blocks of foreground pixels that have predefined maximum dimensions. Then off-line dilation and erosion of all rectangular blocks are arbitrary obtained and stored into suitable look-up array tables. By using the look up tables, the results of the morphological operations to the rectangular blocks are directly obtained. Thus, first all image blocks are replaced by their look-up array tables. Then the morphological operations are applied only to the limited number of the remaining pixels. Experimental results reveal that starting from a block represented binary image morphological operations can be executed with different types of structuring elements in significantly less CPU time. Using the block representation, we are able to perform dilation 16 times faster than non-fast implementations and 10 times faster than an alternative fast implementation based on contour processing. Significant acceleration is also recorded when using this approach for repeated application of dilation (for 10 iterations, dilation using the block representation is over 20 times faster than non-fast implementations and over four times faster than using the fast contour based approach).

Keywords: Mathematical morphology; binary image block decomposition; binary processing; erosion; dilation.

1. Introduction

Mathematical morphology is an active and growing area of image processing and analysis. It is based on set theory and topology.[1,2] Mathematical morphology studies the geometric structure inherent within the images. For this reason, it uses a predetermined geometric shape known as the structuring element. Mathematical morphology has provided solutions to many tasks, where image processing can be applied, such as remote sensing,[3] optical character recognition,[4] image restoration,[5] medical imaging[6] etc.

Erosion and dilation are the fundamental operations of mathematical morphology. Let the set A denote the image to be processed and the set B the structuring element. Binary erosion and dilation operations are defined by the following relations[2]:

$$A\Theta B = \bigcap_{b\in B} (A)_{-b}\,, \tag{1}$$

and

$$A \oplus B = \bigcup_{b\in B} (A)_b \tag{2}$$

respectively. A and B are sets in 2D integer space Z^2 and $(A)_x$ is the translation of A by x, which is defined as follows:

$$(A)_x = \{c \in Z^2 | c = a + x \text{ for } a \in A\}\,. \tag{3}$$

Erosion and dilation are the basic morphological operations. Other significant morphological operations are composed through erosions and dilations. The two most important are the opening and closing operations, which are defined as follows[7]

$$AoB = (A\Theta B) \oplus B\,, \tag{4}$$

and

$$A \bullet B = (A \oplus B)\Theta B\,, \tag{5}$$

respectively.

In order to perform all the above morphological operations, each pixel of the binary image should be accessed and transformed according to the morphological equations. Since morphological operations provide solutions to a wide range of applications, it is very useful to have fast algorithms for efficient execution of morphological operations independently of the shape of the structuring elements. Various techniques have been described in the literature which deal with fast implementation of morphological operations. Several of these techniques are only applicable to gray scale images and are not suitable for pure B/W images.[8,9,10,11] However, a few fast implementations of morphological operators applicable to B/W images have been reported. For example, a contour processing based approach described in Ref. 12 achieves an application-dependent speed up when a small structuring

element (3×3) is applied to a binary image. According to this approach, only the pixels belonging to the external contour of binary objects are involved in morphological processing. A significant reduction in the CPU time is obtained when many iterations of the same morphological operations are required. Also, several approaches mentioned in Ref. 13 process regional extrema in order to detect connected sets in gray scale images. When applied to B/W images, those techniques result to the contour processing approach of Ref. 12. Fast implementation of morphological operations to binary images in the case that the same structuring element is applied many times is also reported in Ref. 14. According to this approach, in order to speed up morphological operations with respect to large convex structuring elements, the logarithmic decomposition of structuring elements is employed. Additionally, bitmap data representation is used and found efficient in terms of memory requirements and in terms of algorithm efficiency because the CPU operates on 32 pixels in parallel.

In this paper, we propose a new fast technique for performing morphological operations in binary images. The proposed technique can be applied with any type of structuring element independently of the shape and size, and is also efficient for repeated applications of the structuring elements. According to it, the binary documents are first decomposed to a set of non-overlapping rectangular blocks of foreground pixels.[15,16,17] Also, suitable look up array tables that contain the results of applying erosion or dilation to all related rectangular blocks, are constructed off-line. By using these look up tables and superposition, the application of the structuring element to all image blocks are directly obtained and all blocks are replaced by their look-up array tables. Then, the morphological operations are applied only to the limited number of the remaining pixels. The final image obtained is exactly the same as the image produced by the classical morphological procedures. It must be noticed that the look up array tables must be initially obtained and then can be applied to any binary morphological operation.

The proposed technique was extensively tested with a variety of images and structuring elements. Experimental results reveal that starting from a block represented binary image we can execute morphological operations using different types of structuring elements with significant reduction in the CPU time.

2. Binary Image Block Decomposition

The representation of binary images using rectangular blocks as primitives has been applied with great success to several image processing tasks, such as image compression,[18,19] fast implementation of the Hough transform[15,16,17] and skeletonization.[20] In our approach, we consider binary image decomposition into non-overlapping rectangular blocks. Since we want to process these blocks off-line, it is preferable to limit their maximum length and width (constrained block decomposition). We use a modified algorithm based on Ref. 16 in order to limit the maximum size of the extracted blocks. This algorithm involves one raster scanning

of the image. According to it, the image is scanned in a top-down direction until the first foreground pixel (x_0, y_0) is found. Then, a procedure that searches and constructs the "best fitting block" at (x_0, y_0), which is the block with the largest area that has the (x_0, y_0) pixel as its upper left corner is applied. All pixels of the "best fitting block" are transformed to background pixels and the procedure is repeated until the whole image is scanned.

Consider a binary image $I(x, y)$, $x = 1, 2, \ldots, x_{\max}$, $y = 1, 2, \ldots, y_{\max}$, defined as follows:

$$I(x,y) = \begin{cases} 1\,, & \text{for foreground pixel,} \\ 0\,, & \text{for background pixel.} \end{cases} \tag{6}$$

A function $B(x_1, y_1, x_2, y_2)$ is defined to specify if the pixels with coordinates (x_1, y_1) and (x_2, y_2) are the opposite vertices of a rectangular block consisting of foreground pixels:

$$B(x_1,y_1,x_2,y_2) = \begin{cases} 1\,, & \text{if } I(x,y) = 1 \;\forall\; x\,, \quad y : x \in [x_1, x_2] \wedge y \in [y_1, y_2] \\ 0\,, & \text{otherwise,} \end{cases} \tag{7}$$

where $x_1, x_2 \in [1, 2, \ldots, x_{\max}]$ and $y_1, y_2 \in [1, 2, \ldots, y_{\max}]$.

Defining that x_m and y_m are the maximum width and length of the rectangular blocks, respectively, the algorithm for the constrained block decomposition is as follows:

Step 1: Set iter = 1.
Step 2: Perform a raster scanning of the image to find a foreground pixel (x_0, y_0). That is $I(x_0, x_0) = 1$.
Step 3: Find the opposite vertex $(x_{\text{op}}, y_{\text{op}})$ of the "best fitting block" at (x_0, y_0), as follows:
  (a) Find $x_1 \geq x_0 : B(x_0, y_0, x_1, y_0) = 1 \wedge x_1 - x_0$ is maximized $\wedge\; x_1 - x_0 \leq x_m$.
  (b) Find $y_1 \geq y_0 : B(x_0, y_0, x_1, y_1) = 1 \wedge y_1 - y_0$ is maximized $\wedge\; y_1 - y_0 \leq y_m$.
  (c) Find $y_2 \geq y_0 : B(x_0, y_0, x_0, y_2) = 1 \wedge y_2 - y_0$ is maximized $\wedge\; y_2 - y_0 \leq y_m$.
  (d) Find $x_2 \geq x_0 : B(x_0, y_0, x_2, y_2) = 1 \wedge x_2 - x_0$ is maximized $x_2 - x_0 \leq x_m$.
  (e) Find $q \in \{1, 2\} : (x_{\text{q}} - x_0)(y_{\text{q}} - y_0) = \max$. Set $x_{\text{op}} = x_{\text{q}}$ and $y_{\text{op}} = y_{\text{q}}$.
Step 4: Set $XF[iter] = x_0$, $XL[iter] = x_{\text{op}}$, $YF[iter] = y_0$, $YL[iter] = y_{\text{op}}$.
Step 5: Set $I(x_0, x_0) = 0 \;\forall\; x \in [XL[iter] \cdots XF[iter]] \wedge y \in [YF[iter] \cdots YL[iter]]$.
Step 6: Set $iter = iter + 1$
Step 7: Until there remains no unscanned foreground pixel, continue with the raster scanning of Step 2.

After following the above steps, the image is represented with a number of rectangular blocks, $b_{\max}$, whose opposite vertices have coordinates $(XF[b], YF[b])$

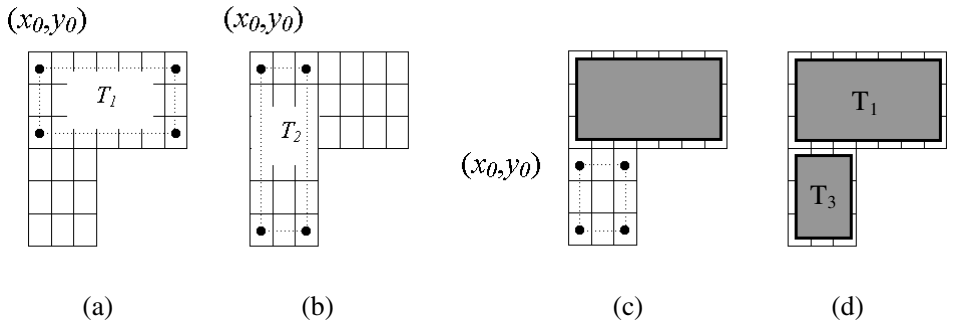


Fig. 1. Block decomposition of a simple binary image.

Fig. 2. Block decomposition of a binary image that contains text.

(upper left vertex) and $(XL[b], YL[b])$ (lower right vertex), where $b \in [1, \ldots, b_{\max}]$. The decomposition of an image into rectangular blocks is demonstrated in Fig. 1. By using a top-down raster scanning a foreground pixel (x_0, y_0) is obtained (Fig. 1(a)). Then two candidate blocks T_1 (Fig. 1(a)) and T_2 (Fig. 1(b)) are obtained, of which T_1 has the larger area. Block T_1 is considered as the first block of the image. To proceed with the extraction of the next rectangular block, all pixels of T_1 are transformed to background pixels. Then using the raster scanning process we arrive at pixel (x_0, y_0) of Fig. 1(c). Similarly block T_3 is considered as the second block of the image. In this way the original image is decomposed into the two rectangular blocks T_1 and T_3 (Fig. 1(d)). In the general case, this procedure is repeated until the whole image is decomposed into blocks. An example of binary image block decomposition of an image that contains text is demonstrated in Fig. 2.

3. Morphological Operations

This section describes how basic morphological operations are performed in a block decomposed binary image.

3.1. *Dilation*

Based on the following property of dilation[21]:

$$(X \cup Y) \oplus B = (X \oplus B) \cup (Y \oplus B)\,, \tag{8}$$

we can state that dilating the original image with structuring element B is equivalent to applying dilation to every rectangular block and then replacing all blocks by their corresponding dilated images. In order to speed up the dilation process, a look up the array table is constructed off-line, containing the dilation results of all rectangles of predefined maximum dimensions. Thus, the dilation results for these blocks are directly obtained during the dilation process. Below follows a detailed step-by-step description of the entire algorithm.

Step 1: Binary image constrained block decomposition. We extract a set of $b_{\max}$ non-overlapping rectangular blocks with maximum dimension x_m by y_m.

Step 2: Construction of a look up array table

$$L(\nu) = \begin{bmatrix} \lambda_{11} \cdots \lambda_{K1} \\ \lambda_{1\Lambda} \cdots \lambda_{K\Lambda} \end{bmatrix}\,, \quad v = 1 \cdots x_m \cdot y_m\,,$$

assigning for every block C of dimension $i \times j$ the $K \times \Lambda$ array $L(j \cdot x_m + i)$, in the following way:

$$L = C \oplus B\,. \tag{9}$$

Step 3: Every block of the image is replaced by its corresponding L table and all values are added in order to construct a new image.

Step 4: The final dilated image is extracted, from the image obtained in Step 3, by considering as foreground points all points that have values greater than 0.

It should be noted that the look up array tables are initially calculated and then can be used for any binary image. This is obvious for all the morphological operations procedures described in this paper. Figure 3 illustrates a simple example of the application of dilation with structuring element B to an image that consists of two rectangular blocks A_1 and A_2.

3.2. *Erosion*

Based on the property that erosion is an increasing transformation, that is[15]:

$$X \subseteq Y \Rightarrow X \Theta B \subseteq Y \Theta B\,, \tag{10}$$

and since every rectangular block is a part of the original image, we can state that pixels belonging to the eroded rectangular blocks also belong to the eroded original image. To find other possible points that belong to the eroded image but not to the eroded blocks, we use the following property of erosion:

$$X \Theta B \subseteq X\,. \tag{11}$$

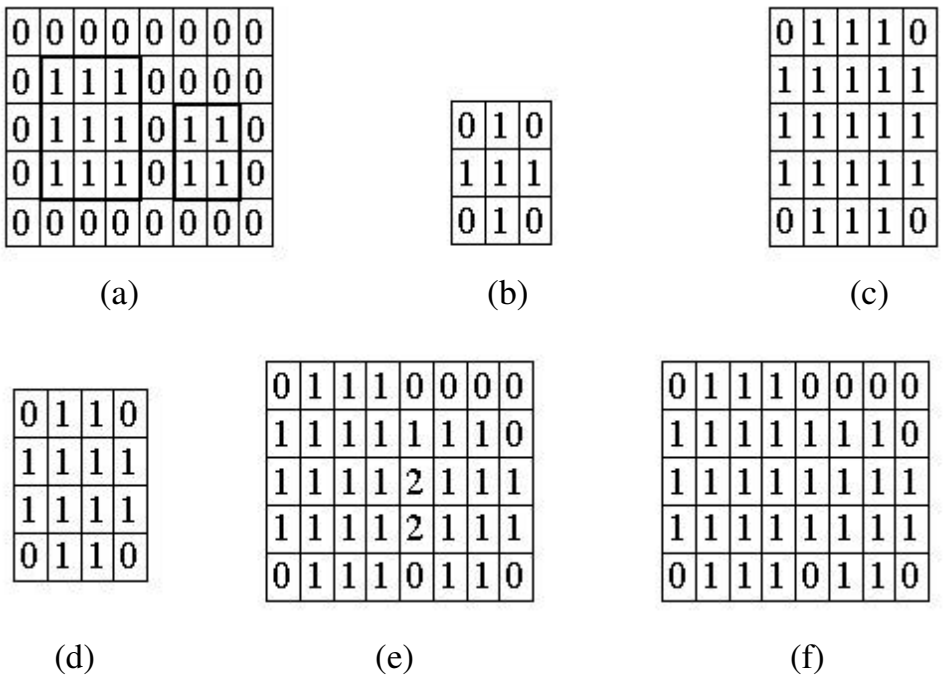


Fig. 3. (a) Original image consisting of two rectangular blocks A_1 and A_2. (b) Structuring element B. (c), (d) Dilation of two blocks. (e) Final image after replacing the blocks by their lookup tables. (f) Final dilated image.

Based on Eqs. (10) and (11), it can be stated that in order to find the erosion of a binary image we can replace all blocks by their eroded images and just check if we have to include points that belong to the original image but not to the eroded block images. In order to track those points we add the original image C to the eroded block image $C\Theta B$. As a result we obtain for every block the value 2 for every original pixel that has not been removed and the value 1 for every pixel that has been removed through erosion. First, all blocks are replaced by their corresponding look up arrays, and then, the erosion is obtained by applying the structuring element only to pixels with value 1. The entire algorithm consists of the following steps:

Step 1: Binary image constrained block decomposition. A set of $b_{\max}$ non-overlapping rectangular blocks with maximum dimension x_m by y_m is extracted.

Step 2: Construction of a look up array table $L(v)$, $v = 1 \cdots x_m \cdot y_m$, assigning for every block $i \times j$ the $K \times \Lambda$ array $L(j \cdot x_m + i)$, in the following way:

$$L = (C\Theta B) + C\,, \tag{12}$$

where C is the corresponding block of the table L.

Step 3: Every block of the image is replaced by its corresponding L table and all values are added in order to construct the final image.

Step 4: The erosion is obtained by applying the structuring element only to points with value 1 considering as foreground points all points with values greater than 0. The points that turn to foreground have value 2. The final eroded image is extracted considering as foreground points all points having value 2.

For example, if the look up table is constructed to support blocks of maximum dimension 30×30 and we want to apply the structuring element B of Fig. 3(b), then block C of dimension 5×3 corresponds to the look up table $L(153)$ (since $5 \cdot 30 + 3 = 153$) which is constructed in the following way:

$L = (C \Theta B) + C =$

0	0	0	0	0
0	1	1	1	0
0	0	0	0	0

$+$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

$=$

1	1	1	1	1
1	2	2	2	1
1	1	1	1	1

Figure 4 illustrates the application of erosion with structuring element B to an image that consists of two rectangular blocks A_1 and A_2.

3.3. *Closing*

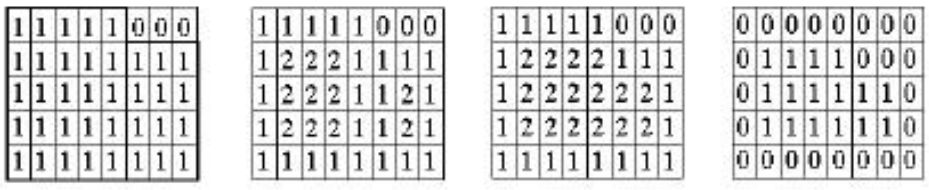
The closing morphological operation for binary images is defined as[7]:

$$X \bullet B = (X \oplus B) \Theta B \,. \tag{13}$$

Taking into account the following property[12]:

$$X \subseteq X \bullet B \subseteq X \oplus B \,, \tag{14}$$

we extract a closed block represented binary image by first performing a dilation (Sec. 3.1) and then applying an erosion to pixels that belong to $X \oplus B$ and not to X. To achieve that, we construct a look-up table $L = (C \oplus B) + \alpha C$ for every block C, where B is the structuring element of size $n \times m$ and a is the surrounding area of the structuring element $B(a = n \cdot m)$. As a result we have for every block a



(a) (b) (c) (d)

Fig. 4. (a) Original image consisting of two rectangular blocks A_1, A_2. (b) Resulting image after applying Step 3 of the erosion algorithm described in Sec. 3.2. (c) Final image after applying Step 4 of the erosion algorithm described in Sec. 3.2. (d) Final eroded image.

value of 1 only for pixels that must be checked for erosion at the next step. Then, all blocks are replaced by their corresponding look up arrays. All points that must not be candidates for erosion have values less than a. The parameter a is used to increase the value of the original block pixels in order to ensure that they will not be candidates in the erosion process. The complete description of the closing algorithm is as follows:

Step 1: Binary image constrained block decomposition. We extract a set of $b_{\max}$ non-overlapping rectangular blocks with maximum dimension $x_m \times y_m$.

Step 2: Construction of a look up array table $L(v)$, $v = 1 \cdots x_m \cdot y_m$, assigning for every block of size $i \times j$ the $K \times \Lambda$ array $L(j \cdot x_m + i)$, in the following way:

$$L = (C \oplus B) + \alpha C\,, \tag{15}$$

where C is the corresponding block of the table L, B is the structuring element $Bn \times m$ and $\alpha = n \times m$.

Step 3: Every block of the image is replaced by its corresponding L table and all values are added in order to construct the final image.

Step 4: We proceed with erosion by applying the structuring element only to points with values less than a considering as foreground points all points with values greater than 0. The points that turn into foreground are given value α. The final closed image is extracted considering as foreground points all points that are greater or equal to α.

For example, for maximum block 30×30, structuring element B (Fig. 3(b)), $a = 9$, block C 5×3 corresponds to look up table $L(153)$ which is constructed in the following way:

$$L = (C \oplus B) + \alpha C =$$

0	1	1	1	1	1	0
1	10	10	10	10	10	1
1	10	10	10	10	10	1
1	10	10	10	10	10	1
0	1	1	1	1	1	0

Figure 5 illustrates the application of closing.

3.4. *Opening*

The opening operation of a binary image X with a structuring element B, denoted $X \circ B$, is defined as[7]:

$$XoB = (X \Theta B) \oplus B\,. \tag{16}$$

As it can be observed, the opening operation is obtained by calculating the erosion of X by B, followed by a dilation of the result by B.

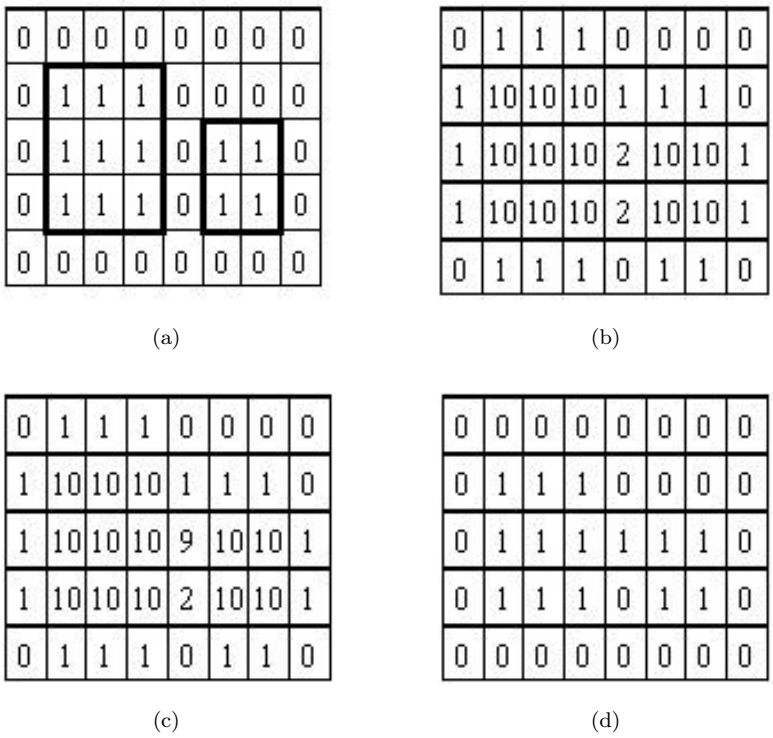


Fig. 5. (a) Original image consisting of rectangular blocks A_1 and A_2. (b) Resulting image after applying Step 3 of the closing algorithms described in Sec. 3.3. (c) Final image after applying Step 4 of the closing algorithm described in Sec. 3.3. (d) Final closed image.

The following morphological property[12]:

$$X \Theta B \subseteq X o B \subseteq X \tag{17}$$

leads us to the conclusion that all points of the eroded image belong to the opened image and a further check must be made only for points that belong to the original image and not to the eroded image. First we follow Steps 1–3 of Sec. 3.2 in order to extract the eroded image. To trace the points that must be checked for belonging to the opened image, we give value 3 to all pixels that belong to the original image but not to the eroded image. At a next step we check only those pixels for dilation. The algorithm is as follows:

Step 1: We follow Steps 1–3 of Sec. 3.2.

Step 2: We proceed with erosion applying the structuring element only to points with 1 value considering as foreground points all points with values greater than 0. Points that turn into foreground are given value 2 while points that turn to background are given value 3.

Step 3: We proceed with dilation by applying the structuring element only to points with value 3 considering as foreground points all points with value 3. Points

that turn to foreground are given value 1. The final closed image is extracted considering as foreground pixels only pixels equal to 1 or 2.

Figure 6 illustrates the application of closing with structuring element B (Fig. 3(b)) to an image that consists of two rectangular blocks A_1 and A_2.

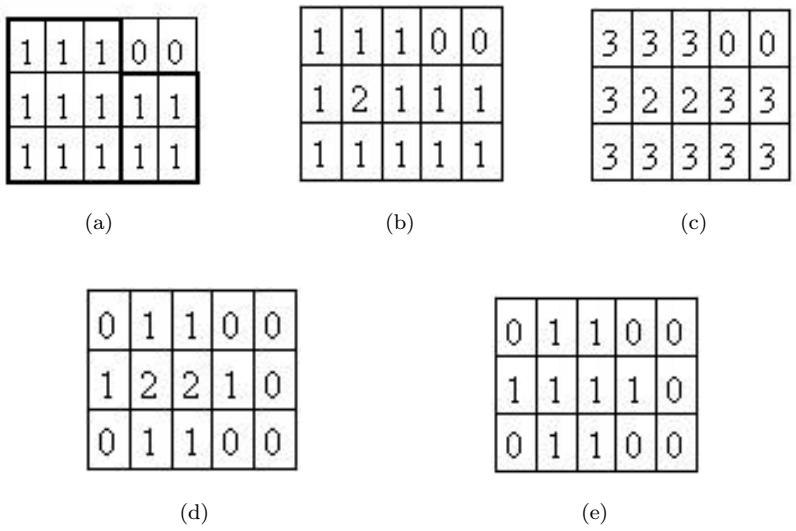


Fig. 6. (a) Original image consisting of two rectangular blocks A_1 and A_2. (b) Resulting image after applying Step 1 of the opening algorithm described in Sec. 3.4. (c) Resulting image after applying Step 2 of the opening algorithm described in Sec. 3.4. (d) Image after the dilation of Step 3 of Sec. 3.4. (e) Final opened image.

4. Experimental Results

Experiments were conducted with a variety of binary images. All methods were implemented in C++ language at a Pentium II/800 MHz with Windows 2000 O/S. We constructed the final images after applying the four basic morphological operations of dilation, erosion, opening and closing, with a variety of structuring elements. The resulting images were found to be identical to the corresponding images that result from point represented images. Tables 1–4 show the execution times when applying the four basic morphological operations to the 8 CCITT benchmark b/w images (ftp://ftp.funet.fi/pub/graphics/misc/test-images/) with a variety of structuring elements, starting from point or block representations or the contour processing method described in Ref. 12. Since for the contour processing method only 3×3 structuring elements are directly applicable (12), application of 5×5 elements is implemented through twofold application of 3×3 elements. Figures 7–10 show the average execution times for all images. Using the block representation, we are able to perform dilation 16 times faster than using the points representation and

Table 1. Processing times for dilation, erosion, opening and closing using a 3×3 square structuring element starting from point or block representations or the contour processing method described in Ref. 12.

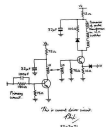
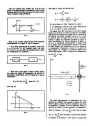
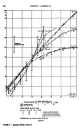
Structuring element: **Square 3 x 3** 		**Dilation**	**Erosion**	**Opening**	**Closing**
	Points	1.74 sec	0.24 sec	1.98 sec	2.04 sec
	Contours	0.71 sec	0.84 sec	1.67 sec	1.56 sec
	Blocks	0.04 sec	0.26 sec	0.46 sec	0.24 sec
	Points	1.74 sec	0.24 sec	1.98 sec	2.04 sec
	Contours	0.64 sec	0.84 sec	1.66 sec	1.54 sec
	Blocks	0.06 sec	0.26 sec	0.44 sec	0.22 sec
	Points	1.66 sec	0..28 sec	2.00 sec	2.04 sec
	Contours	0.71 sec	0.84 sec	1.87 sec	1.68 sec
	Blocks	0.10 sec	0.32 sec	0.54 sec	0.28 sec
	Points	1.18 sec	0.30 sec	1.60 sec	1.60 sec
	Contours	0.83 sec	0.93 sec	2.04 sec	1.97 sec
	Blocks	0.16 sec	0.40 sec	0.62 sec	0.34 sec
	Points	1.26 sec	0.26 sec	1.56 sec	1.60 sec
	Contours	0.71 sec	0.84 sec	1.85 sec	1.70 sec
	Blocks	0.08 sec	0.30 sec	0.50 sec	0.26 sec
	Points	1.30 sec	0.22 sec	1.56 sec	1.58 sec
	Contours	0.64 sec	0.84 sec	1.74 sec	1.61 sec
	Blocks	0.04 sec	0.26 sec	0.46 sec	0.24 sec
	Points	1.24 sec	0.26 sec	1.58 sec	1.6 sec
	Contours	0.83 sec	0.93 sec	1.95 sec	1.99 sec
	Blocks	0.14 sec	0.34 sec	0.58 sec	0.30 sec
	Points	0.84 sec	0.66 sec	1.58 sec	1.56 sec
	Contours	0.90 sec	0.67 sec	1.76 sec	1.63 sec
	Blocks	0.24 sec	0.56 sec	0.78 sec	0.42 sec
Average	Points	1.37 sec	0.31 sec	1.73 sec	1.76 sec
	Contours	0.75 sec	0.84 sec	1.81 sec	1.71 sec
	Blocks	0.11 sec	0.34 sec	0.55 sec	0.29 sec

Table 2. Processing times for dilation, erosion, opening and closing using a 3×3 cross structuring element starting from point or block representations or the contour processing method described in Ref. 12.

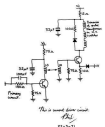
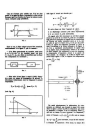
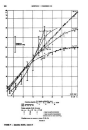
Structuring element: **Cross 3 x 3** 		**Dilation**	**Erosion**	**Opening**	**Closing**
	Points	1.30 sec	0.22 sec	1.54 sec	1.58 sec
	Contours	0.95 sec	0.89 sec	2.46 sec	2.23 sec
	Blocks	0.04 sec	0.24 sec	0.44 sec	0.22 sec
	Points	1.30 sec	0.24 sec	1.56 sec	1.58 sec
	Contours	1.04 sec	0.89 sec	2.34 sec	2.29 sec
	Blocks	0.04 sec	0.26 sec	0.44 sec	0.22 sec
	Points	1.24 sec	0.26 sec	1.56 sec	1.58 sec
	Contours	1.04 sec	0.89 sec	2.66 sec	2.54 sec
	Blocks	0.08 sec	0.32 sec	0.52 sec	0.26 sec
	Points	1.18 sec	0.30 sec	1.58 sec	1.62 sec
	Contours	1.13 sec	0.93 sec	2.99 sec	2.87 sec
	Blocks	0.16 sec	0.40 sec	0.64 sec	0.34 sec
	Points	1.26 sec	0.26 sec	1.56 sec	1.58 sec
	Contours	1.13 sec	0.89 sec	2.68 sec	2.50 sec
	Blocks	0.08 sec	0.30 sec	0.52 sec	0.26 sec
	Points	1.30 sec	0.24 sec	1.54 sec	1.58 sec
	Contours	0.96 sec	0.89 sec	2.59 sec	2.37 sec
	Blocks	0.06 sec	0.26 sec	0.46 sec	0.24 sec
	Points	1.24 sec	0.26 sec	1.60 sec	1.62 sec
	Contours	1.13 sec	0.93 sec	2.92 sec	2.93 sec
	Blocks	0.12 sec	0.34 sec	0.58 sec	0.32 sec
	Points	0.84 sec	0.66 sec	1.56 sec	1.56 sec
	Contours	1.31 sec	0.71 sec	2.58 sec	2.42 sec
	Blocks	0.26 sec	0.56 sec	0.76 sec	0.42 sec
Average	Points	1.21 sec	0.30 sec	1.56 sec	1.59 sec
	Contours	1.09 sec	0.88 sec	2.65 sec	2.52 sec
	Blocks	0.10 sec	0.33 sec	0.54 sec	0.28 sec

Table 3. Processing times for dilation, erosion, opening and closing using a 5×5 square structuring element starting from point or block representations or the contour processing method described in Ref. 12.

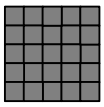
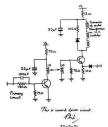
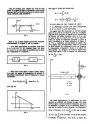
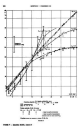
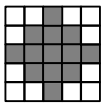
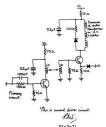
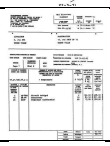
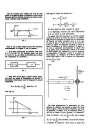
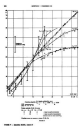
Structuring element: **Square 5 x 5** 		**Dilation**	**Erosion**	**Opening**	**Closing**
	Points	3.60 sec	0.30 sec	4.66 sec	4.56 sec
	Contours	1.15 sec	1.83 sec	3.36 sec	3.13 sec
	Blocks	0.06 sec	0.32 sec	0.62 sec	0.38 sec
	Points	3.60 sec	0.34 sec	4.60 sec	4.58 sec
	Contours	1.53 sec	1.83 sec	3.29 sec	3.06 sec
	Blocks	0.06 sec	0.36 sec	0.60 sec	0.32 sec
	Points	3.36 sec	0.36 sec	4.62 sec	4.52 sec
	Contours	1.37 sec	1.94 sec	3.75 sec	3.35 sec
	Blocks	0.10 sec	0.40 sec	0.78 sec	0.46 sec
	Points	3.02 sec	0.38 sec	4.76 sec	4.46 sec
	Contours	1.51 sec	2.04 sec	4.02 sec	3.90 sec
	Blocks	0.26 sec	0.46 sec	1.10 sec	0.76 sec
	Points	3.38 sec	0.34 sec	4.64 sec	4.52 sec
	Contours	1.37 sec	1.84 sec	3.66 sec	3.43 sec
	Blocks	0.12 sec	0.40 sec	0.78 sec	0.50 sec
	Points	3.52 sec	0.32 sec	4.64 sec	4.54 sec
	Contours	1.19 sec	1.95 sec	3.44 sec	3.18 sec
	Blocks	0.08 sec	0.36 sec	0.66 sec	0.38 sec
	Points	3.32 sec	0.34 sec	4.70 sec	4.48 sec
	Contours	1.61 sec	1.93 sec	3.93 sec	4.03 sec
	Blocks	0.20 sec	0.40 sec	0.90 sec	0.68 sec
	Points	2.20 sec	1.78 sec	4.48 sec	4.44 sec
	Contours	2.02 sec	1.45 sec	3.49 sec	3.25 sec
	Blocks	0.28 sec	1.02 sec	1.30 sec	0.58 sec
Average	Points	3.25 sec	0.52 sec	4.64 sec	4.51 sec
	Contours	1.47 sec	1.85 sec	3.62 sec	3.42 sec
	Blocks	0.14 sec	0.46 sec	0.84 sec	0.51 sec

Table 4. Processing times for dilation, erosion, opening and closing using a 5×5 diamond structuring element starting from point or block representations or the contour processing method described in Ref. 12.

Structuring element: Diamond 5 x 5 		**Dilation**	**Erosion**	**Opening**	**Closing**
	Points	2.92 sec	0.30 sec	3.32 sec	3.30 sec
	Contours	2.01 sec	1.98 sec	4.98 sec	4.25 sec
	Blocks	0.08 sec	0.32 sec	0.58 sec	0.32 sec
	Points	2.92 sec	0.34 sec	3.28 sec	3.32 sec
	Contours	2.05 sec	1.98 sec	4.64 sec	4.28 sec
	Blocks	0.08 sec	0.36 sec	0.56 sec	0.30 sec
	Points	2.76 sec	0.38 sec	3.30 sec	3.28 sec
	Contours	2.10 sec	1.88 sec	5.12 sec	5.24 sec
	Blocks	0.16 sec	0.42 sec	0.70 sec	0.38 sec
	Points	2.54 sec	0.38 sec	3.38 sec	3.30 sec
	Contours	2.12 sec	2.01 sec	5.89 sec	5.85 sec
	Blocks	0.30 sec	0.52 sec	0.96 sec	0.62 sec
	Points	2.76 sec	0.36 sec	3.32 sec	3.32 sec
	Contours	2.35 sec	1.88 sec	5.30 sec	5.12 sec
	Blocks	0.16 sec	0.42 sec	0.70 sec	0.42 sec
	Points	2.86 sec	0.32 sec	3.32 sec	3.30 sec
	Contours	2.03 sec	1.89 sec	5.04 sec	4.75 sec
	Blocks	0.10 sec	0.36 sec	0.62 sec	0.34 sec
	Points	2.66 sec	0.34 sec	3.38 sec	3.30 sec
	Contours	2.30 sec	1.98 sec	5.89 sec	6.03 sec
	Blocks	0.24 sec	0.42 sec	0.82 sec	0.54 sec
	Points	1.82 sec	1.34 sec	3.24 sec	3.24 sec
	Contours	2.56 sec	1.55 sec	5.03 sec	4.98 sec
	Blocks	0.42 sec	0.86 sec	1.12 sec	0.54 sec
Average	Points	2.65 sec	0.47 sec	3.32 sec	3.29 sec
	Contours	2.19 sec	1.89 sec	5.24 sec	5.06 sec
	Blocks	0.19 sec	0.46 sec	0.76 sec	0.43 sec

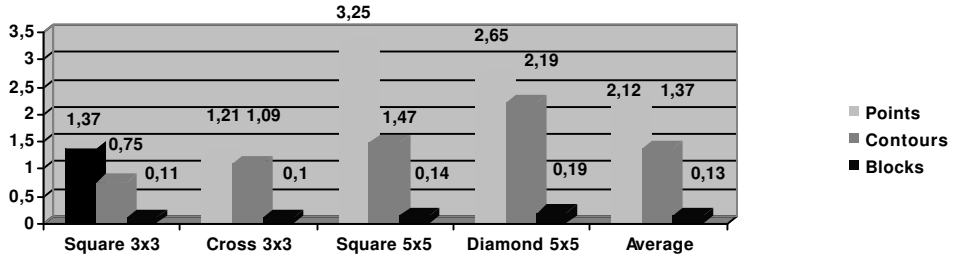


Fig. 7. Average times for dilation.

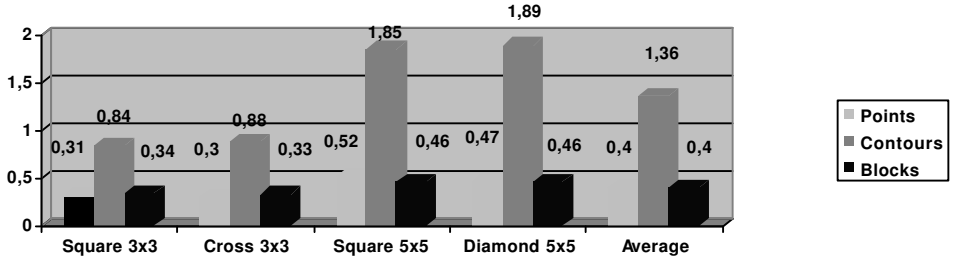


Fig. 8. Average times for erosion.

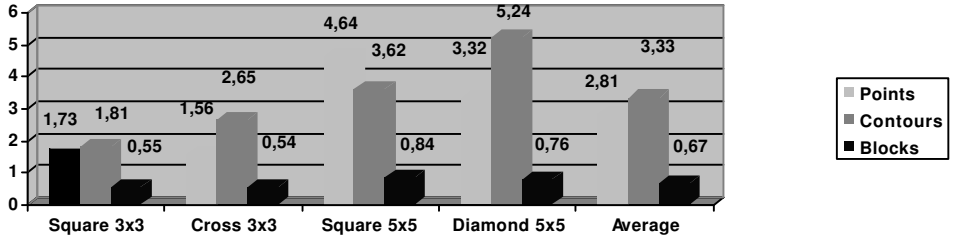


Fig. 9. Average times for opening.

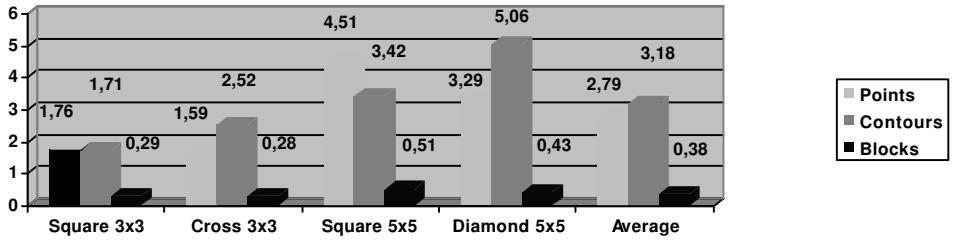


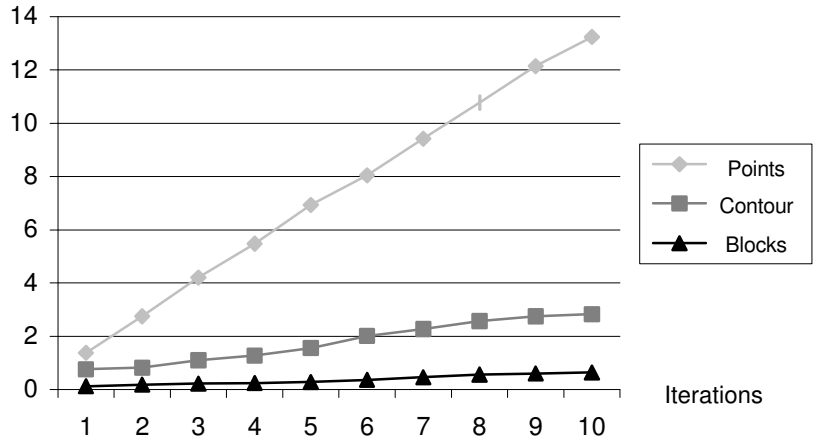
Fig. 10. Average times for closing.

10 times faster than using the contour approach. Although the erosion operation is not accelerated when using block representation, the significant reduction in the execution time for dilation boosts the operation of opening and closing as well (opening is four times faster than using points and five times faster than using blocks, while closing is seven times faster than using points and eight times faster than using blocks) (Figs. 5, 6).

The basic contour processing based approach for fast morphological operations described in Ref. 12 is reported to obtain a significant reduction in the CPU time for many iterations. Therefore we recorded times for applying dilation with a 3×3 structuring element for many iterations. Table 5 shows the processing times required for many iterations when applying dilation with a 3×3 square structuring element with all three approaches (Points, Contours and Blocks). The recorded results show that for the block representation approach there is always a significant decrease in processing time compared to the other two approaches irrespectively of the number of iterations. For example, a 10-fold application of dilation using the block representation is over 20 times faster than using the points representation and over four times faster than using the contour based approach.

A further indication of the usefulness of our approach for accelerating morphological transformations on binary images comes from a comparison with the state of the art SDC morphology Matlab toolbox.[22] In the general case of gray scale

Table 5. Processing times (sec) for many iterations when applying dilation with a 3×3 square structuring element.



Iterations	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Points	1.37	2.74	4.20	5.48	6.94	8.04	9.41	10.78	12.15	13.24
Contour	0.75	0.82	1.09	1.28	1.55	2.01	2.28	2.56	2.74	2.83
Blocks	0.11	0.18	0.21	0.24	0.27	0.36	0.46	0.55	0.60	0.64

images, the algorithms implemented in this Matlab toolbox give very significant acceleration with respect to the standard Matlab routines for morphological operations included in the image processing toolbox. However, our experiments showed that in the case of binary images the morphology toolbox does not give significant acceleration over the standard Matlab image processing toolbox and in some cases it is even slower. We have computed average times regarding the execution of one iteration for each of the four basic morphological operations to the 8 CCITT benchmark b/w images. For the Matlab morphological toolbox routines, average times are 4.01 sec for dilation, 4.16 sec for erosion, 7.00 sec for opening and 6.97 sec for closing. For the standard Matlab routines, the corresponding figures are 5.49 sec for dilation, 12.14 sec for erosion, 3.23 sec for opening and 3.29 sec for closing. Hence for these binary images, dilation and erosion are faster, while opening and closing are somewhat slower. Moreover, as is evident from Figs. 7–10, average times obtained using the mathematical morphology Matlab toolbox are much larger than the corresponding times obtained using our block representation implementation. It is evident from these tables that the block representation routines give at least a ten-fold acceleration over the corresponding Matlab morphological toolbox routines. These results show that there is a clear need for fast implementation of morphological algorithms specifically for binary images and that our approach addresses this need with considerable success.

5. Conclusions

This paper proposes new techniques for fast implementation of basic morphological operations. The binary images are first decomposed in non-overlapping rectangular blocks of foreground pixels. Then, using look up tables and superposition the morphological operations are directly obtained. The resultant images are exactly the same with the images produced by the classical morphological procedures. The presented experimental results confirm the advantages of the proposed approach.

References

1. G. Matheron, *Random Sets and Integral Geometry* (Wiley, New York, 1975).
2. J. Serra, *Image Analysis and Mathematical Morphology*, Vol. I. (Academic Press, London, 1982).
3. J. Chanussot and P. Lambert, An application of mathematical morphology to road network extraction on SAR images, *Computational Imaging and Vision*, Kluwer Academic Publishers, Dordrecht, **12** 309–406 (1998).
4. S. Mori, H. Nishida, abd H. Yamada, "Optical character recognition," Wiley-Interscience (1999).
5. F. Decenciere, "Restauration automatique de films anciens," *PhD Thesis* (Ecole des Mines de Paris, 1997).
6. I. Pratikakis, "Watershed-driven image segmentation," *PhD thesis* (Vrije Universiteit Brussel, 1998).
7. R. C. Gonzalez and R. E. Woods, *Digital Image Processing* (Addison-Wesley Pub. Company, 1993).

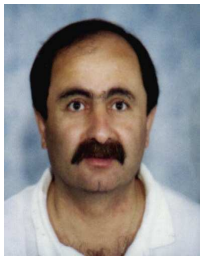
8. R. Wai-Kit and C. K. Li, "A fast algorithm to morphological operations with flat structuring element," *IEEE Trans. on Circ. and Systems-II: Analog and Signal Processing* **45** 387–391 (1998).
9. M. V. Droogenbroeck and H. Talbot, "Fast computation of morphological operations with arbitrary structuring elements," *Pattern Recognition Letters* **17** 1451–1460 (1996).
10. S. J. Ko, A. Morales, and K. H. Lee, "Fast recursive algorithms for morphological operators based on the basis matrix representation," *IEEE Trans. IP* **5**(6), 1073–1077 (1996).
11. S. J. Ko, A. Morales, and K. H. Lee, "Block basis matrix implementation of the morphological open-closing, close-opening," *IEEE Signal Processing Letters* **2**(1), 7–9 (1995).
12. L. J. Vliet and B. J. H. Verwer, "A contour processing method for fast binary neighbouring operations," *Pattern Recognition Letters* **7** 27–36 (1998).
13. A. Meijster and M. H. F. Wilkinson, "A comparison of algorithms for connected set openings and closings," *IEEE PAMI* **24**(4), 484–494 (2002).
14. R. D. Boomgaard and R. V. Balen, "Methods for fast morphological image transforms using bitmapped binary images," *CVGIP: Graphical Models and Image Processing* **54** 252–258 (1992).
15. B. Gatos, S. J. Perantonis, and N. Papamarkos, "Accelerated Hough tranform using rectangular image decomposition," *Electronic Letters* **32** 730–732 (1996).
16. S. J. Perantonis, B. Gatos, and N. Papamarkos, "Block decomposition and segmentation for fast Hough transform evaluation," *Pattern Recognition* **32** 811–824 (1999).
17. S. J. Perantonis, B. Gatos, and N. Papamarkos, "Image segmentation and linear feature identification using rectangular block decomposition," in *Proc. Third IEEE Int. Conf. on Electronics, Circuits and Systems* (ICECS 1996), pp. 183–186.
18. A. Quddus and M. M. Fahmy, "Binary text image compression using overlapping rectangular partitioning," *Pattern Recognition Letters* **20** 81–88 (1999).
19. S. A. Mohamed and M. M. Fahmy, "Binary image compression using efficient partitioning into rectangular regions," *IEEE Trans. Commun.* **43** 1888–1892 (1995).
20. K. C. Fan, D. Chen, and M. Wen, "Skeletonization of binary images with nonuniform width via block decomposition and contour vector matching," *Pattern Recognition* **31** 823–828 (1998).
21. M. Sonka, V. Hlavac, and R. Boyle, *Image Processing Analysis and Machine Vision* (Chapman & Hall, London, 1993).
22. http://www.mmorph.com/

Dr. Basilios Gatos was born in 1967, in Athens, Greece. He received his Electrical Engineering Diploma in 1992, and his PhD degree in 1998, both from the Electrical and Computer Engineering Department of the Democritus University of Thrace, Xanthi, Greece. His PhD thesis is on optical character recognition techniques. He is currently working as a Researcher at the Institute of Informatics and Telecommunications of the National Center for Scientific Research "Demokritos", Athens, Greece. His main research interests are in Image Processing and Document Image Analysis, OCR and Pattern Recognition. He has 29 publications in journals and international conference proceedings and has participated in several research programs funded by the European community. He is a member of the Technical Chamber of Greece.

Nikos Papamarkos was born in Alexandroupoli, Greece, in 1956. He received his Diploma Degree in Electrical and Mechanical Engineering from the University of Thessaloniki, Thessaloniki, Greece, in 1979 and the PhD Degree in Electrical Engineering in 1986 from the Democritus University of Thrace, Greece. From 1987 to 1990 Dr. Papamarkos was a lecturer, from 1990 to 1996 an Assistant Professor, 1996–2003 Associate Professor in the Democritus University of Thrace where he is currently Professor since 2003. During 1987 and 1992 he has also served as a Visiting Research Associate at the Georgia Institute of Technology, USA. His current research interests are in digital signal processing, filter design, optimization algorithm, image processing, pattern recognition, neural networks and computer vision. Professor Nikos Papamarkos is a member of IEEE, a member of IAPR and a member of the Greek Technical Chamber.

Dr. Stavros J. Perantonis was born in Athens, Greece in 1960. He is the holder of a BS degree in Physics from the Department of Physics, University of Athens, an MSc degree in Computer Science from the Department of Computer Science, University of Liverpool and a PhD in Computational Physics from the Department of Physics, University of Oxford. He was an associate researcher at the Department of Applied Mathematics, University of Liverpool in the period of 1987–1991. Since 1992 he has been with the Institute of Informatics and Telecommunications, National Centre for Scientific Research "Demokritos", where he currently holds the position of Senior Researcher and Head of the Computational Intelligence Laboratory. His current technical and research activities are in the areas of computational intelligence, neural networks, pattern recognition and multimedia processing. Dr. Perantonis is the author or coauthor of over 90 papers in international journals and

conference proceedings. He has managed or participated in many EU or nationally funded projects, involving R&D of the above mentioned technologies for industrial, medical, financial, environmental or web-based applications.



Ioannis Andreadis received the Diploma Degree from the Department of Electrical, DUTH, Greece, in 1983 and the MSc and PhD from the University of Manchester Institute of Science & Technology (UMIST), UK, in 1985 and 1989, respectively. His research interests are mainly in machine vision and VLSI based computing architectures for machine vision. He joined the Department of Electrical & Computer Engineering, DUTH in 1993. He is a member of the Editorial Board of the Pattern Recognition Journal, Technical Chamber of Greece and IEEE.