Applying Morphological Transformations on a Block Represented Binary Image

B. Gatos[1], N. Papamarkos[2], I. Andreadis[2] and S. J. Perantonis[3]

[1] Department of Digital Technologies, Lambrakis Press Archives,
8, Heyden Str, 104 34 Athens, Greece
bgat@dolnet.gr

[2] Department of Electrical and Computer Engineering, Democritus University of Thrace,
67 100 Xanthi, Greece
papamark@voreas.ee.duth

[3] Institute of Informatics and Telecommunications, National Research Center "Demokritos",
153 10 Athens, Greece
sper@iit.demokritos.gr

Abstract. Morphological transformations are commonly used to carry out a variety of image processing tasks. In this paper, morphological transformations are applied to a binary image represented by a set of non-overlapping rectangular blocks. Off-line dilated and eroded images of all related rectangular blocks are used to accelerate certain morphological operations. Suitable look up array tables are constructed off-line for all rectangular blocks. At a first step, all image blocks are replaced by their look-up array tables. At a second step, morphological operations are applied only to a limited number of remaining pixels. The experimental results showed that starting from a block represented image we can construct an opened or closed image in significantly less CPU time (24% and 57% CPU time gain for opening and closing operations, respectively).

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 image. 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 in remote sensing, optical character recognition, radar image sequence recognition, medical imaging 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 are defined by:

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

and

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

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

$$(A)_x = \left\{ c \in Z^2 \mid c = a + x \text{ for } a \in A \right\} \tag{3}$$

Two other important morphological operations are opening and closing defined as:

$$A \circ B = (A \ominus B) \oplus B \tag{4}$$

and

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

respectively.

In order to perform the morphological operations, each pixel of the binary image should be accessed and transformed according to equations (1) and (2). In this paper we propose a fast method for performing morphological operations on a binary image represented by a set of non-overlapping rectangular blocks. We calculate off-line dilated or eroded images of all related rectangular blocks and form suitable look up array tables. At a first step, all image blocks are replaced by their look-up array tables. At a second step, morphological operations are applied only to a limited number of remaining pixels.

The representation of binary images using rectangular blocks as primitives has been applied with great success for several image processing tasks, such as image compression [3,4], Hough transform fast implementation [5-7] and skeletonization [8]. 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 revised algorithm based on [6] in order to limit the maximum size of the extracted blocks. Figure 1 illustrates block decomposition of a binary image.

2 Morphological operations

In this section, basic morphological operations are performed starting from a block represented binary image.

(a) (b)

Fig. 1. Block decomposition of a binary image

2.1 Dilation

Based on the following property of dilation [9]:

$$(X \cup Y) \oplus B = (X \oplus B) \cup (Y \oplus B) \quad (6)$$

we can state that dilating the original image using structuring element B is equivalent to applying dilation to every rectangular block and replacing all blocks by their corresponding dilated images. Considering that all rectangular blocks have maximum width and length xm and ym, we can proceed off-line to the dilation of all possible blocks and store the results to look up array tables. There follows a complete description of the algorithm.

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

Step 2: Construction of a look up array table $L(v) = \begin{bmatrix} \lambda_{11} & \dots & \lambda_{K1} \\ & & \\ \lambda_{1\Lambda} & \dots & \lambda_{K\Lambda} \end{bmatrix}$, $v = 1 \dots xm * ym$, assigning for every block ixj the KxΛ array $L(j * xm+i)$, in the following way:

$$L = C \oplus B \quad (7)$$

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 final dilated image is extracted considering as foreground points all points that have values greater than 0.

Figure 2 illustrates the application of dilation with structuring element B to an image that consists of 2 rectangular blocks A_1, A_2.

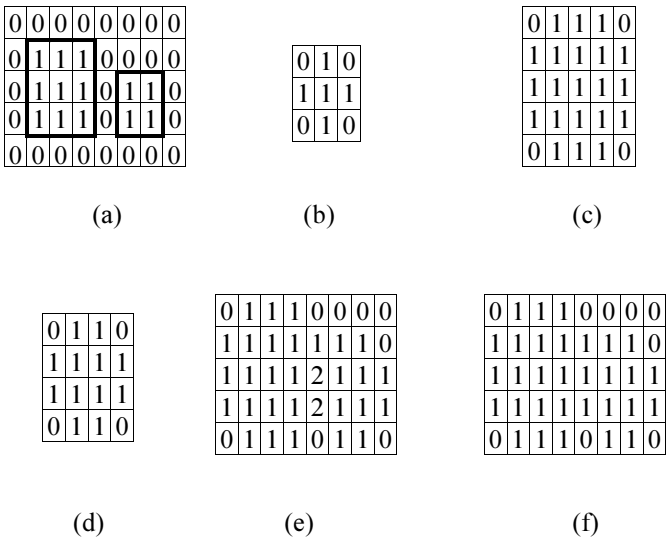

Fig. 2. (a) Original image consisting of two rectangular blocks A_1, A_2. (b) Structuring element B. (c),(d) Dilation of two blocks. (e) Final image after replacing the blocks by their lookup tables. (f) Final dilated image

2.2 Erosion

Based on the property that erosion is an increasing transformation [9], that is:

$$X \subseteq Y \Rightarrow X \ominus B \subseteq Y \ominus B \tag{8}$$

and since rectangular block is 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 obvious erosion property:

$$X \ominus B \subseteq X \tag{9}$$

Based on equations (8) and (9), we can state that in order to produce an eroded block represented 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 just add to the eroded block image $C \ominus B$ the original block image C. As a result we have for every block a 2 value for every original pixel that has not been removed and an 1 value for every pixel that has been removed due to erosion. First, we replace all blocks by their corresponding look up arrays, and then, we proceed to erosion applying the structuring

element only to pixels with value 1. The complete description of the 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 *xm*x*ym*.

Step 2: Construction of a look up array table $L(v)$, $v = 1 \ldots xm * ym$, assigning for every block *i*x*j* the *K*x*Λ* array $L(j * xm+i)$, in the following way:

$$L = (C \ominus B) + C \qquad (10)$$

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: 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. The points that turn to foreground are given value 2. The final eroded image is extracted considering as foreground points all points with 2 value.

For example, for maximum block 30x30, structuring element B (Fig. 2b), block C 5x3 corresponds to look up table $L(153)$ (since 5*30+3 = 153) which is constructed in the following way:

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

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 3 illustrates the application of erosion with structuring element B to an image that consists of 2 rectangular blocks A_1, A_2.

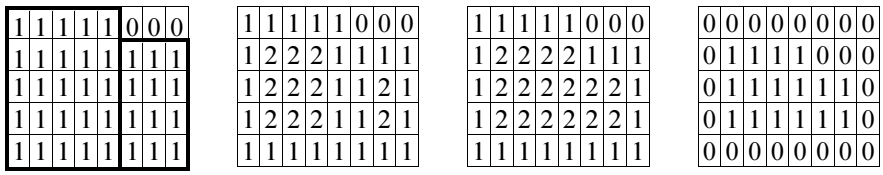

Fig. 3. (a) Original image consisting of two rectangular blocks A_1, A_2. (b) Resulting image after applying step 3 of section 2.2. (c) Final image after applying step 4 of section 2.2. (d) Final eroded image.

2.3. Closing

Taking into account the following property [9]:

$$X \subseteq X \bullet B \subseteq X \oplus B \tag{11}$$

we extract a closed block represented binary image by first proceeding to a dilation (section 2.1) and then checking to apply erosion to pixels that belong to $X \oplus B$ and not to X. To achieve that, we construct for every block C a look-up table $L = (C \oplus B) + \alpha C$, where B is the structuring element nxm and α is the surrounding area of the structuring element B ($a = n*m$). As a result we have for every block an 1 value only for pixels that must be checked for erosion at the next step. Then, we replace all blocks from their corresponding look up arrays. All points that must not be candidates for erosion have value less than a. We use parameter a to increase the value of the original block pixels in order to ensure that they will not be candidates for 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 xmxym.

Step 2: Construction of a look up array table $L(v)$, $v = 1 \ldots xm * ym$, assigning for every block ixj the KxΛ array $L(j * xm+i)$, in the following way:

$$L = (C \oplus B) + \alpha C \tag{12}$$

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

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 applying the structuring element only to points with values less than α considering as foreground points all points with values greater than 0. The points that turn to 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 30x30, structuring element B (Fig. 2b), $a = 9$, block C 5x3 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 4 illustrates the application of closing.

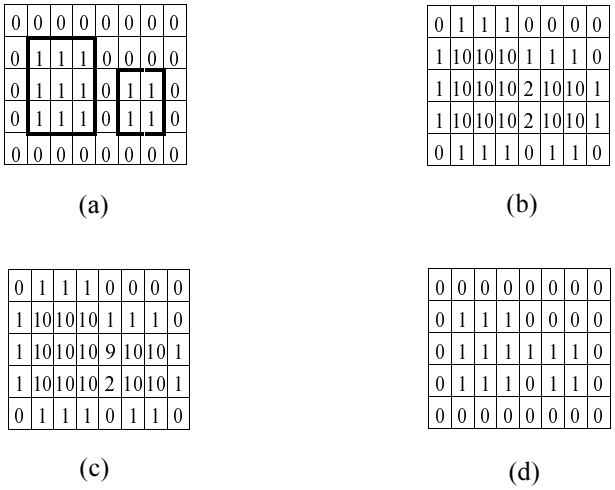

Fig. 4. (a) Original image consisting of rectangular blocks A_1, A_2. (b) Resulting image after applying step 3 of section 2.3. (c) Final image after applying step 4 of section 2.3. (d) Final closed image.

2.4. Opening

The following morphological property [9]:

$$X \ominus B \subseteq X \circ B \subseteq X \tag{13}$$

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. We first follow steps 1-3 of section 2.2 in order to extract the eroded image. To trace the points that must be checked if they belong to the opened image, we give value 3 for all pixels that belong to the original image and 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 section 2.2.

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 to foreground are given value 2 while points that turn to background are given value 3.

Step 3: We proceed with dilation applying the structuring element only to points with 3 value considering as foreground points all points with 2 value. 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 5 illustrates the application of closing with structuring element B (Fig. 2b) to an image that consists of 2 rectangular blocks A_1, A_2.

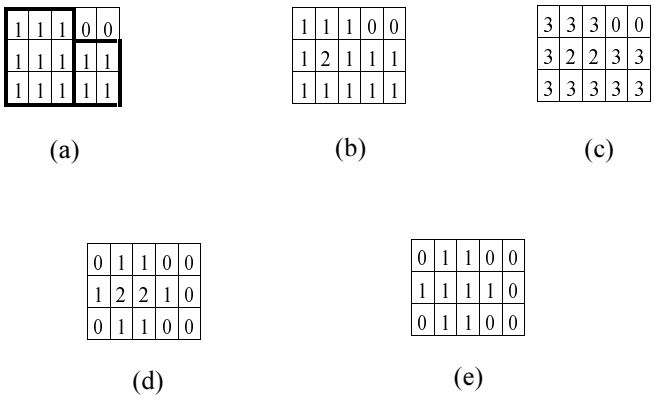

Fig. 5. (a) Original image consisting of two rectangular blocks A_1, A_2. (b) Resulting image after applying step 1 of section 2.4. (c) Resulting image after applying step 2 of section 2.4. (d) Image after the dilation of step 3 of section 2.4. (e) Final closed image.

Table 1. Processing times

	Point representation	Block representation	CPU gain
Erosion	0,37 sec	0,48 sec	- 29,7 %
Dilation	0,66 sec	0,20 sec	69,7 %
Opening	1,08 sec	0,82 sec	24,1 %
Closing	1,29 sec	0,55 sec	57,4 %

3 Experimental results

Experiments were conducted with a variety of binary images represented by a set of non-overlapping rectangular blocks. We constructed the final images after applying the four basic morphological operations, that is 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. Table 1 shows the average times consumed for all operations starting from point or block representations as well as the CPU time gained when using block representation. Despite the fact that erosion takes more CPU time when using block representation, we notice a significant acceleration of dilation which boosts the operation of opening and closing.

4 Conclusions

This paper proposes a new technique for fast implementation of morphological transformations in binary images represented by a set of non-overlapping rectangular blocks. Using predetermined look-up tables, certain morphological operations are fast implemented. The experimental results obtained are very promising.

5 References

[1] Matheron, G.: Random Sets and Integral Geometry. Wiley, New York (1975)

[2] Serra, J.:Image Analysis and Mathematical Morphology: Vol.I. Academic Press, London (1982)

[3] Quddus, A., Fahmy, M.M.: Binary text image compression using overlapping rectangular partitioning. Pattern Recognition Letters, 20 (1999) 81-88

[4] Mohamed, S.A., Fahmy, M.M.: Binary image compression using efficient partitioning into rectangular regions. IEEE Trans. Commun., 43 (1995) 1888-1892

[5] Gatos, B., Perantonis, S.J., Papamarkos, N.: Accelerated Hough tranform using rectangular image decomposition. Electronic Letters, 32 (1996) 730-732

[6] Perantonis, S.J., Gatos, B., Papamarkos, N.: Block decomposition and segmentation for fast Hough transform evaluation. Pattern Recognition, 32 (1999) 811-824

[7] Perantonis, S.J., Gatos, B., Papamarkos, N.: Image segmentation and linear feature identification using rectangular block decomposition. Third IEEE Int. Conf. On Electronics, Circuits and Systems, ICECS 96 (1996) 183-186.

[8] Fan, K.C., Chen, D., Wen, M.: Skeletonization of binary images with nonuniform width via block decomposition and contour vector matching. Pattern Recognition, 31 (1998) 823-828

[9] Sonka, M., Hlavac, V., Boyle, R.: Image Processing Analysis and Machine Vision, Chapman & Hall, London (1993)