Comparing Point and Block Representation in Computer Vision and Image Processing Tasks

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

[1] Computational Intelligence Laboratory,
Institute of Informatics and Telecommunications,
National Research Center "Demokritos",
153 10 Athens, Greece
{bgat, sper}@iit.demokritos.gr}
http://www.iit.demokritos.gr/cil

[2] Department of Electrical and Computer Engineering,
Democritus University of Thrace,
67 100 Xanthi, Greece
papamark@ee.duth.gr
http://ipml.ee.duth.gr/~papamark/

Abstract. The description of a digital image in terms of simple geometrical shapes, such as polygonal shapes, is a well established methodology that often proves useful for several image processing tasks, mainly to speed up image processing operations. The representation of binary images using rectangular blocks as primitives has been applied with great success to several computer vision and image processing tasks, such as fast implementation of morphological operations, fast implementation of the Hough transform, fast run length smoothing algorithm (RLSA) evaluation and fast projection profiles evaluation. Such decomposition can also contribute towards effective and fast shape analysis, image compression, image segmentation and feature identification. In this paper, we present an overview of the implementations of the most important image processing and vision tasks using binary image block representation. We focus on the acceleration achieved in comparison with the classical pixel based approaches.

1 Introduction

Digital image representation in terms of simple geometrical shapes has proved useful for effective and fast implementation of several computer vision and image processing tasks [1]. The main motivation for suggesting image representations other than the classical pixel based representation is that processing all the pixels of the image is a difficult and time consuming task. Existing approaches for image representation include image description in terms of:

Morphological operations [2]: A structuring element is used as a geometrical primitive to evaluate the shape of an object. The morphological operation translates a structuring element and sequentially finds the points where the translated structuring

element is included in the objects under consideration, but without overlapping the previously included structuring elements. The result of the operation is thus a subset of erosion. The representation results in a set of loci of the translated structuring elements that are included in the object but do not overlap. It is an information preserving, shift and scale invariant and non-redundant representation. Many image processing and analysis tasks can be easily performed by using image representation based on morphological operations.

Quadtrees [3]: A class of hierarchical data structures whose common property is that they are based on the principle of recursive decomposition of space. The most studied quadtree approach to region representation, termed a region quadtree, is based on the successive subdivision of the image array into four equal-sized quadrants. The region quadtree representation is especially useful for performing set operations such as the union and intersection of several images, for basic transformations, for area and moments calculation, for connected component labeling etc.

Skeletons [4]: Image representation by shape skeletons has been a constant topic of research for decades. The skeleton is the set of mid-lines that preserve much of the topological shape information of the image. It can be calculated entirely by the basic operations of mathematical morphology. Generally, the skeleton must be well centered, well connected, must accurately reflect the shape and allow the precise reconstruction of the original image. Skeleton representation can be used for efficient image compression, feature extraction, image transformations etc.

Chain codes [5]: The line structures that result from image contour tracing are characterized by the property that each data node in sequence coincides with one of the eight grid nodes that surround the previous data node. If we label these eight neighboring grid nodes from 0 to 7 in counterclockwise sense starting from the positive x axis, we can represent the line structures simply by sequences of octal digits (chain code). Chain code encoding is convenient for processing highly irregular line drawings and is especially useful for moment calculation, area calculation, image smoothing, object rotation, object correlation etc.

Rectangular blocks [6]: 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, fast implementation of morphological operations, fast implementation of the Hough transform.

In this paper, we present an overview of the implementations of the most important image processing and vision tasks based on image representation by a set of rectangular blocks. We start with the rectangular block decomposition algorithm according to which an image can be transformed to a set of non-overlapping rectangles. Then, we present an abstract description of all approaches that perform computer vision and image processing tasks starting from a block represented binary image. These include image segmentation, image compression, morphological operations, Hough transform, run length smoothing algorithm (RLSA), projection profiles and two-dimensional moments. Finally, we present a comparative chart that demonstrates the reported accelerations for several Computer Vision and Image Processing tasks when using binary image block representation.

2 Rectangular Block Decomposition

The rectangular block decomposition algorithm involves one raster scanning of the image. 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:

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

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:

$$B(x_1,y_1,x_2,y_2) = \begin{cases} 1, & \text{if } I(x,y)=1 \; \forall x,y : x \in [x_1,x_2] \wedge y \in [y_1,y_2] \\ 0, & \text{otherwise,} \end{cases} \quad (2)$$

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

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_{op},y_{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.
(b) Find $y_1 \geq y_0 : B(x_0,y_0,x_1,y_1) = 1 \; \wedge \; y_1 - y_0$ is maximized.
(c) Find $y_2 \geq y_0 : B(x_0,y_0,x_0,y_2) = 1 \; \wedge \; y_2 - y_0$ is maximized.
(d) Find $x_2 \geq x_0 : B(x_0,y_0,x_2,y_2) = 1 \; \wedge \; x_2 - x_0$ is maximized.
(e) Find $q \in \{1,2\}$: $(x_q - x_0)\,(y_q - y_0) = \max$. Set $x_{op} = x_q$ and $y_{op} = y_q$.

Step 4: Set $XF[iter] = x_0$, $XL[iter] = x_{op}$, $YF[iter] = y_0$, $YL[iter] = y_{op}$.

Step 5: Set $I(x_0,x_0)=0 \; \forall \; x \in [XL[iter] \ldots XF[iter]] \wedge y \in [YF[iter] \ldots YL[iter]]$.

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 $(XF[b], YF[b])$ (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.

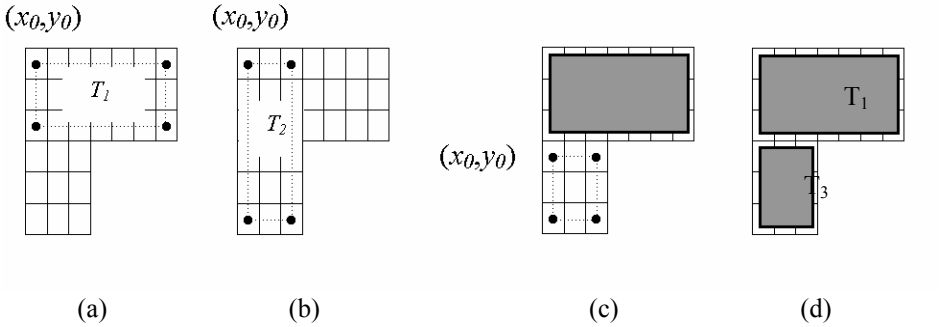

Fig. 1. Decomposition of a simple binary

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

3 Block Representation Applications

In this section, we present an abstract description of all approaches that perform computer vision and image processing tasks starting from a block represented binary image. These include image segmentation, image compression, morphological operations, Hough transform, run length smoothing algorithm (RLSA), projection profiles and two-dimensional moments.

3.1 Image Segmentation

Using an extension of the algorithm in section 2, it is possible to group together all blocks that belong to the same connected component [7]. The proposed algorithm is as follows:

The "best fitting block" at the first foreground pixel is found using a top-down scan. Then, a search is conducted for all "best fitting blocks" at all foreground pixels which are adjacent to the perimeter of already detected blocks and lie outside these blocks. This process is repeated iteratively, transforming the pixels of all detected blocks to background pixels, until no foreground pixel remains immediately outside the perimeter of all blocks of the current connected image component. Once the process has been completed, the coordinates of the extracted blocks for the first connected component are stored in matrices *XF[i]*, *XL[i]*, *YF[i]*, *YL[i]*, for $i = 1 \ldots Blocks$. The above process is repeated until all image objects have been segmented. A complete description of the segmentation algorithm based on block decomposition can be found in [7].

3.2 Image Compression

A lossless binary image coding technique has been proposed based on image representation using rectangular blocks [8,9]. These rectangular blocks are either non-overlapping [8] or overlapping [9]. The two opposite vertices of each rectangle are compressed using a simple encoding technique.

According to [9], the opposite vertices of the rectangles are represented in a matrix *R* having the same size as the image. The upper-left vertex of any rectangle (larger than the size of one pixel) is given the symbol "1", the lower-right vertex of the rectangle is given the symbol "2", whereas for isolated pixels the symbol "3" is used. An example is shown in Fig. 3. The coordinate encoding procedure of all nonzero values of the *R* matrix is based on row by row matrix scan and described in [9]. Image compression based on block representation outperforms the modified READ, REC, CCITT run-length coding and Rectangular coding by 14.54%, 21.68%, 42.49% and 25.04% respectively.

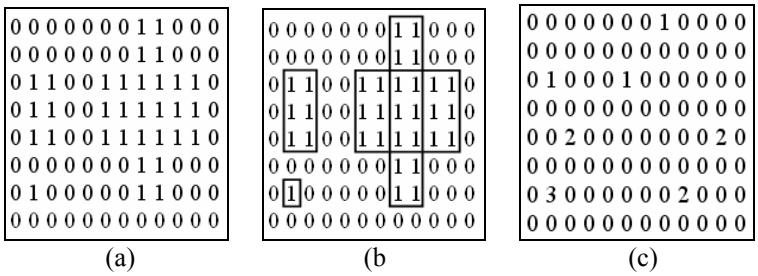

Fig. 3. (a) Original image. (b) Overlapping block partitioning. (c) Image compression using *R* matrix.

3.3 Morphological Operations

Mathematical morphology [10] is an active and growing area of image processing and analysis that has provided solutions to many tasks, such as remote sensing, optical character recognition, image restoration, medical imaging etc. Erosion and dilation are the fundamental operations of mathematical morphology. Other significant morphological operations are based on erosion and dilation. The two most important are the opening and closing operations. 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 [11]. Fast techniques for performing morphological operations in binary images have been proposed based on binary image block representation [12],[13].

The binary image is first decomposed into a set of non-overlapping rectangular blocks of foreground pixels. 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.

3.4 Hough Transform

The Hough transform [14] has emerged in recent decades as a powerful method for many image processing and pattern recognition applications. A major drawback of its implementation in large images is its low speed. The development of fast Hough transform algorithms has attracted much attention in the literature [15]. There is proposed that the description of binary images using rectangular blocks can be useful for speeding up the implementation of the Hough transform algorithm [6].

The block representation of binary images achieves fast evaluation of the Hough transform field by analytically calculating the contribution to cells in the Hough accumulator array of a whole rectangular block rather than of each individual pixel. Consider a block R whose opposite vertices are located at at (k_1, l_1) and (k_2, l_2), where $k_2 \geq k_1$ and $l_2 \geq l_1$. Clearly, (k_2-k_1+1) (l_2-l_1+1) pixels lie in the interior or on the perimeter of R (see Fig. 4). Consider a cell in the accumulator array $A(r_1, \theta)$ that corresponds to all straight lines determined by the parameters θ and $r_1-1/2<r< r_1+1/2$, with r_1 integer. Clearly, the number of points P in R contributing to this cell can be approximated by the area occupied by the intersection of the rectangle R and of the strip in the xy plane restricted between the straight lines $(r_1-1/2, \theta)$ and $(r_1+1/2, \theta)$ (small discrepancies are

due to the discrete image grid). The analytical calculation of P can found in [6]. Using block representation for the evaluation of Hough transform field, significant acceleration is observed, especially in applications where prevalent linear features cannot be captured correctly using edge detection, so that the whole image should preferably be used.

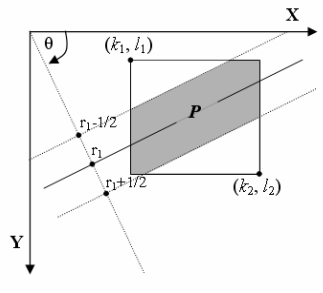

Fig. 4. Evaluation of rectangular block contribution to Hough transform space.

3.5 Run Length Smoothing Algorithm (RLSA)

RLSA is based on examining the white runs existing in the horizontal and vertical directions. For each direction, white runs whose lengths are smaller than a threshold smoothing value are eliminated. RLSA is usually used for image segmentation and object recognition [16].

Block representation can be useful for speeding up the implementation of RLSA [17]. In the case that the image is represented by a set of non-overlapping rectangular blocks, we do not need to scan the entire image and pass from all possible white and black runs. We just examine the white runs that start from the right side of all possible image blocks in the case of horizontal smoothing (see Fig. 5), or from the bottom side of the blocks for vertical smoothing. From all white runs, we select those with short length and turn them to black runs. In this way, a significant acceleration of RLSA evaluation is achieved.

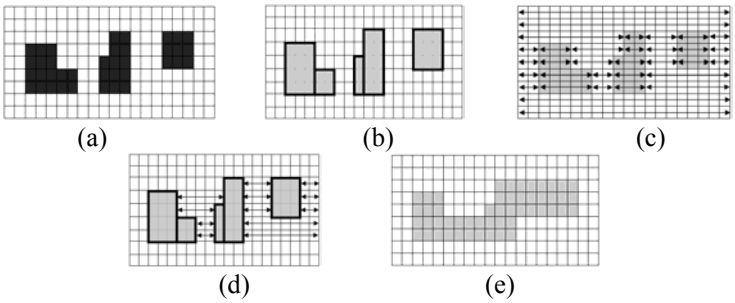

Fig. 5. Horizontal RLSA. (a) Original image. (b) Block decomposition. (c) Runs examined for point representation. (d) Runs examined for block representation. (e) Final smoothed image

3.6 Projection profiles

Projection profiles are based on image profiling in various directions. By calculating the local minima of horizontal and vertical projections we can define several segments into which the image can be divided. Projection profiles are also used for document skew detection [18].

The implementation of horizontal and vertical projection profiles can be accelerated when using binary image block representation [17]. In the case of a block represented image, the lengths of all black runs of all blocks are known in advance. Taking advantage of this information, the projection profiles are evaluated by adding all black runs of all image blocks at the horizontal and vertical directions. Starting from a block represented binary image, a significant acceleration of projection profiles evaluation is achieved.

3.7 Two dimensional moments

2-D statistical moments are widely used for several computer vision and image processing tasks [19]. Various types of moments have been used to recognize image patterns in a number of applications (geometrical moments, central moments, normalized moments, moment invariants, Zernike moments, Legendre moments and complex moments). Since moments implementation involves high computational time, several techniques for fast moment calculation have been reported [20]. Binary image block representation has also been reported suitable for fast 2-D moments implementation [21].

The 2-D geometrical moments of order (p,q) of the image *I* are given by the following formula:

$$m_{pq} = \sum_{x}\sum_{y} x^p y^q \qquad \forall x, y : I(x,y) = 1 \tag{3}$$

If the image *I* is represented with b_{max} rectangular blocks $B(x_1,y_1,x_2,y_2)$, then the 2-D geometrical moments of order (p,q) of the image are given by the following formula:

$$m_{pq} = \sum_{i=1}^{b_{\max}} bm_{pq}^{x_1 y_1 x_2 y_2} \quad \text{where} \quad bm_{pq}^{x_1 y_1 x_2 y_2} = \sum_{x=x1}^{x2}\sum_{y=y_1}^{y_2} x^p y^q \tag{4}$$

Analytical formulas are provided for *bm* fast calculation which corresponds to the geometric moments of any rectangular block $B(x_1,y_1,x_2,y_2)$ [21]. The 2-D geometrical moments of the whole image are computed as the summation of the 2-D geometrical moments of all individual blocks of the binary image. The same approach is used for the acceleration of the central moments, the normalized central moments and the moment invariants. The reported reduction of the computation time for the geometrical moments up to the order (4,4), is by a factor between 10 and 50 for images with a high entropy value and by a factor of ~ 200 for images with large areas of object level.

4 Accelerations for all tasks

The reported acceleration for the tasks of section 3 when using binary image block representation are the following:

Morphological operations: A reduction in computational time by a factor up to 20 (20 for dilation with 10 iterations, 16 for dilation, 7 for closing, 4 for opening). No acceleration is reported for erosion operation.

Hough transform: A reduction in computational time by a factor up to 18 (up to 18 when using RLSA pre-processing, otherwise up to 4).

RLSA: A reduction in computational time by a factor up to 6.

Projection profiles: A reduction in computational time by a factor up to 23.

2-D moments: A reduction in computational time by a factor between 10 to 50.

A comparative chart that demonstrates the reported accelerations for several computer vision and image processing tasks when using binary image block representation can be found in Fig.6.

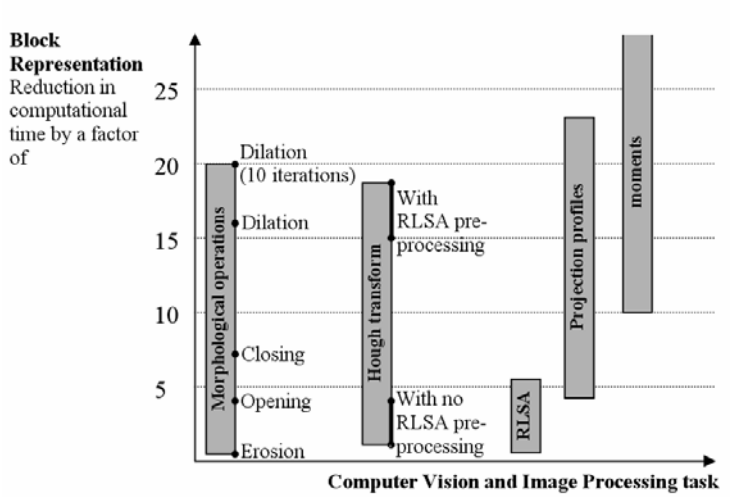

Fig. 6. Reported accelerations for several computer vision and image processing tasks when using binary image block representation.

5 Conclusions

In this paper, we present an overview of the implementations of the most important image processing and vision tasks based on image representation by a set of rectangular blocks. The reported accelerations in comparison with the classical pixel based approaches show that starting from a block represented image we can have an average reduction in computational time by a factor of ~10 which can be even higher for evaluating project profiles or two-dimensional moments.

References

1. Sonka, M., Hlavac, V., Boyle, R.: Image Processing, Analysis and Machine Vision. Chapman & Hall, London (1993)
2. Wang, D., Haese-Coat, V., Ronsin, J.: Shape decomposition and representation using a recursive morphological operation. Pattern Recognition 28 (1995) 1783-1792
3. Samet, H.: The quadtree and related hierarchical data structures. Comput. Surv. 16 (1984) 187-260
4. Kresch, R., Malah, D.: Skeleton-Based Morphological Coding of Binary Images. IEEE Trans. Image Process. Vol. 7, No. 10 (1998) 1387-1399
5. Freeman, H.: Computer processing of line drawings. ACM Comput. Surv. 6 (1974) 57-97
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 1996) 183-186
8. Mohamed, S. A., Fahmy, M. M.: Binary image compression using efficient partitioning into rectangular regions. IEEE Trans. Commun. 43 (1995) 1888-1892
9. Quddus, A., Fahmy, M. M.: Binary text image compression using overlapping rectangular partitioning. Pattern Recognition Letters 20 (1999) 81-88
10. Serra, J.: Image Analysis and Mathematical Morphology , Vol. I. Academic Press, London (1982)
11. Ko, S. J., Morales A., Lee, K. H.: Fast recursive algorithms for morphological operators based on the basis matrix representation. IEEE Trans. IP 5, No. 6 (1996) 1073-1077
12. Gatos, B., Papamarkos N., Andreadis, I., Perantonis, S. J.: Fast Implementation of Morphological Operations Using Image Block Decomposition. To appear at the International Journal of Image and Graphics (2003)
13. Gatos, B., Papamarkos N., Andreadis, I., Perantonis, S. J.: Applying Morphological Transformations on a Block Represented binary Image. Fourth International Workshop on Document Analysis Systems (DAS 2000) 487-495
14. Duda, R. D., Hart, P. E.: Use of the Hough transform to detect lines and curves in pictures. Commun. ACM 15 (1972) 11-15
15. Guil, N., Villalba, J., Zapata, E. L.: A fast Hough transform for segment detection. IEEE Trans. Image Process. 4 (1995) 1541-1548
16. Kasturi, R., Bow, S., El-Masri, W., Shah, J., Gattiker, J., Mokate, U.: A System for Interpretation of Line Drawings. IEEE Trans. Patt. Anal. Mach. Intell. 12 (1990) 978-991
17. Gatos, B., Papamarkos, N.: Applying Fast Segmentation Techniques at a Binary Image Represented by a Set of Non-Overlapping Blocks. Sixth International Conference on Document Analysis and Recognition (ICDAR 2001) 1147-1151
18. Ciardiello, G., Scafur, G., Degrandi, M. T., Spanda, M. R., Roccoteli, M. P.: An experimental system for office document handling and text recognition. 9th Int. Conf. On Patt. Recogn. (1998) 739-743
19. Teague, M. R., Image analysis via the general theory of moments. J. Opt. Soc. Amer. 70 (1980) 920-930
20. Jiang, X. Y., Bunke, H.: Simple and fast computation of moments. Pattern Recognition 24 (1991) 801-806
21. Spiliotis, I. M., Mertzios, B. G.: Real-Time Computation of Two-Dimensional Moments on Binary Images Using Image Block Representation. IEEE Trans. Image Process. Vol. 7, No. 11 (1998) 1609-1615