REAL-TIME COMPUTATION OF 2-D MOMENTS ON BLOCK REPRESENTED BINARY IMAGES ON THE SCAN LINE ARRAY PROCESSOR

Iraklis M. Spiliotis and Basil G. Mertzios
Department of Electrical and Computer Engineering, Democritus University of Thrace,
67100 Xanthi, HELLAS
Tel: +30 541 79559; Fax: +30 541 26473
e-mail: spiliot@demokritos.cc.duth.gr ; mertzios@demokritos.cc.duth.gr

ABSTRACT

This paper presents an algorithm for the real-time computation of 2-D statistical moments on binary images on the Scan Line Array Processor (SLAP). The binary images are represented as sets of nonoverlapping rectangular areas. This representation scheme is called Image Block Representation. The real-time computation of moments in block represented images is achieved by exploiting the rectangular structure of the blocks. The algorithms for image block representation and for the real-time computation of moments are implemented on the Scan Line Array Processor (SLAP).

I. INTRODUCTION

The most common image representation format is a two-dimensional (2-D) array, each element of which has the brightness value of the corresponding pixel. Other approaches to image representation aim to provide machine perception of images in pieces larger than a pixel and are separated in two categories: Boundary based methods and region based methods. Such approaches include quadtree representations [1], chain code representations [2], contour control point models [3], autoregressive models [4] and the interval coding representation [5]. One common objective of the above methods is the representation of an image in a more suitable form for a specific operation.

Recently an advantageous representation for binary images, which is called *Image Block Representation* (IBR) and constitutes an efficient tool for image processing and analysis techniques has been presented [6]-[8]. Using the block represented binary images, real-time computation of 2-D statistical moments is achieved through analytical formulae [6], [8]. The real-time computation of the moments on the Scan Line Array Processor (SLAP) is presented in this paper.

The SLAP architecture [9]-[12] is a linear array of N processors. Each processor is indexed in ascending order from 0 to (N-1). After each scan line is received, each processor P_k latches in that value with the corresponding column index k. Each processor P_k is connected by a bidirectional communication link to processors P_{k-1} and P_{k+1}, wherever they exist. Processors P_{k-1} and P_{k+1} are referred to as the left and right neighbors of processor P_k, respectively. The bandwidth of each communication link is assumed to be a word, where a word is defined to be $\log N$ bits, and each processor can transfer a constant number of words to its immediate neighbors in a unit time. The processors operate together in an SIMD fashion, and each processor is a general purpose sequential processor with the capacity for conditional command execution and local address generation. Each processor has associated with it a random access memory, which can hold N words. In a unit of time, each processor can access its local memory to read or write a word. Incoming image data is loaded line by line into the processor array, with a distinct processor receiving each image column.

The meaning of the term real-time [10], is that if the required processing is based on neighborhoods of size mxm, then the output is generated at a rate of $O(m)$ operations per line and a latency of $O(m)$ scan lines, which is the best that can be achieved on this model.

II. IMAGE BLOCK REPRESENTATION

II.1. Concepts of the 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 1, in each image. These rectangulars, which are called *blocks* in the terminology of this paper, have their edges parallel to the image axes and contain an integer number of image pixels. At the extreme case, the minimum rectangular area of the image is one pixel.

Consider a set that contains as members all the nonoverlapping blocks of a specific binary image, in such a way that no other block can be extracted from the image (or equivalently each pixel with object level belongs to only one block). It is always feasible to represent a binary image with a set of all the nonoverlapping blocks with object level, i.e. by IBR.

The IBR concept leads to a simple and fast algorithm, which requires just one pass of the image and simple bookkeeping process. In this pass all object level intervals are extracted and compared

with the previous extracted blocks. In the following, the IBR algorithm is given.

Algorithm 1: *Image Block Representation.*
Step 1: Consider each line y of the image f and find the object level intervals in line y.
Step 2: Compare intervals and blocks that have pixels in line y-*1*.
Step 3: If an interval does not match with any block, this is the beginning of a new block.
Step 4: If a block matches with an interval, the end of the block is in the line y.
■

As a result of the application of the above algorithm, we obtain a set of all the rectangular areas with level 1 that form the object. A block represented image is denoted as:

$$f(x,y) = \{b_i : i = 0,1,...,n-1\} \quad (1)$$

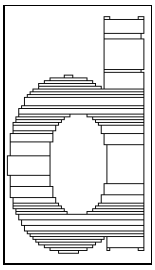

Figure 1. Image of the character d and the blocks.

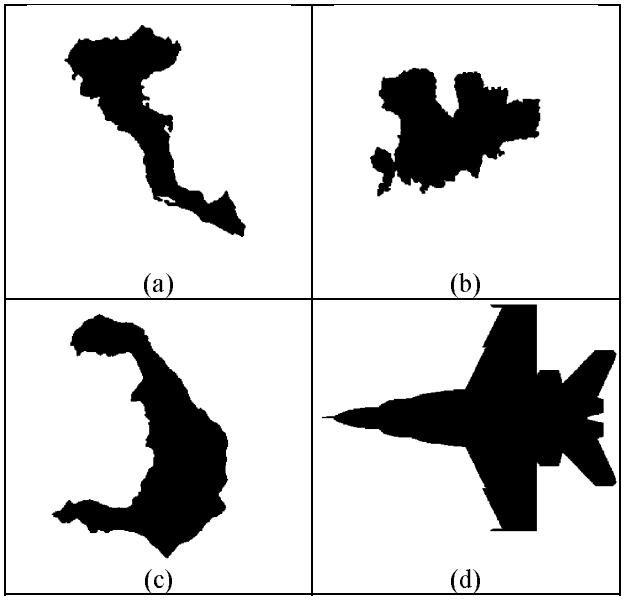

Figure 2. A set of test images. (a). Image of the island Corfu of 512x512 pixels. (b) Image of the island Mikonos of 512x512 pixels. (c) Image of the island Santorini of 512x512 pixels. (d) Aircraft image of 512x697 pixels.

where n is the number of the blocks. Each block is described by four integers, the coordinates of the upper left and down right corner in vertical and horizontal axes. Fig. 1, illustrates the blocks that represent an image of the character d.

In Fig. 2 four test images are illustrated, while in Table 1 the number of the pixels with object level, the number of the rows with object pixels, the number of the blocks extracted from these images (using the Algorithm 1) and the required storage space for both the 2-D rerpesented and the block represented images and are shown. It can be seen that the number of the blocks generated by the Algorithm 1, is significantly less than the number of the rows with black pixels. In the worst case of the island Mikonos image of Fig. 5 (b), where the number of the rows with object pixels is 249 and the number of the blocks is 232, it should be noted that the number of the gulfs and peninsulas of the island is significantly large and therefore the number of the blocks is respectively large.

Table 1. The number of the pixels with object level, the number of the rows with object pixels, the number of the blocks, the required storage for the 2-D images and the required storage for the block represented images for the set of the test images of Figure 2.

Image	object pixels	object rows	blocks	2-D image bytes	bytes for blocks
Corfu	41605	411	250	32768	2000
Mikonos	47368	249	232	32768	1856
Santorini	63203	474	257	32768	2056
Aircraft	118831	494	397	44608	3176

II.2. SLAP Implementation of the Image Block Representation Algorithm

Consider the binary image $\mathbf{F}= f(j,k)$, $j,k=0,1,...,N-1$ to load to the SLAP. Each image row $\mathbf{r}_j$ is loaded simultaneously to the N processors of the SLAP. Each processor P_k receives the value of the pixel $f(j,k)$. Then the following cases are discriminated:

- If $f(j,k)=1$, the processor P_k broadcasts the value k to its left immediate neighbor P_{k-1}, if this latter exists. At the next time unit, the processor P_k broadcasts the value k+1, if it has been sent from the processor P_{k+1} to the processor P_{k-1}. At the next time unit, the processor P_k broadcasts the value k+2, if it has been sent from the processor P_{k+2}, to the processor P_{k-1}, e.t.c.
- If $f(j,k)=0$, the processor P_k broadcasts the value 1 to its right immediate neighbor P_{k+1}, if this latter exists.

Using the above procedure, each processor P_{k1} for which $f(j,k_1)=1$, $k_1>0$ and receives the value 1 from its left processor, or if $f(j,k_1)=1$, $k_1=0$, corresponds to a left limit of an object level interval in row $\mathbf{r}_j$. Also the processor P_{k1} receives

the broadcasted value k_2 from the processor P_{k2} at the right end of the object level interval. Clearly, this maximum requirement of this process is $O(N)$ time units per scan line, in the case where an object interval occupies an entire image row.

Finally, each processor P_{k1} that corresponds to the left limit of an object level interval of this image row I_j, or to a left limit of a block from the previous row, checks the triplet (k_1,k_2,j) with the triplet (x_1,x_2,y_1) that corresponds to a block of the previous row, if this exists. If such a block does not exist, the triplet (k_1,k_2,j) is stored in the local memory of the processor P_{k1} and this is the beginning of a new block. If the triplet (x_1,x_2,y_1) exists, while the triplet (k_1,k_2,j) does not exist, the previous row block is stored as (x_1,x_2,y_1,y_2), where $y_2 = j-1$. If both the triplets exist and $k_1 \neq x_1$ or $k_2 \neq x_2$, then the previous row block is also stored as (x_1,x_2,y_1,y_2), where $y_2 = j-1$.

The whole IBR process requires $O(N^2)$ unit times, where NxN are the image dimensions.

III. COMPUTATION OF THE 2-D MOMENTS

III.1. Computation of the moments on block represented binary images

Consider a binary digital image $f(x,y)$, with N_1 pixels in horizontal axis and N_2 pixels in vertical axis. The 2-D geometrical moments of order (p,q) of the image are defined by the relation:

$$m_{pq} = \sum_{x=0}^{N_1-1} \sum_{y=0}^{N_2-1} x^p y^q f(x,y), \quad p,q = 0,1,2,... \qquad (2)$$

Since the background level is 0, only the pixels with level 1 are taken into account for the computation of the moments. Therefore, if the image $f(x,y)$ is represented by n blocks, as it is described in (1), all the image pixels with level 1 belong to the k image blocks and therefore (2) may be rewritten as:

$$m_{pq} = \sum_{i=0}^{n-1} m_{pq}^{b_i} = \sum_{i=0}^{n-1} \sum_{x=x_{1,b_i}}^{x_{2,b_i}} \sum_{y=y_{1,b_i}}^{y_{2,b_i}} x^p y^q \qquad (3)$$

where x_{1,b_i}, x_{2,b_i} and y_{1,b_i}, y_{2,b_i} are the coordinates of the block b_i with respect to the horizontal axis and to the vertical axis, respectively. According to (3), the 2-D geometrical moments of the whole image are computed as the summation of the 2-D geometrical moments of all the individual blocks of the binary image.

In (3), if the rectangular form appeared within the blocks is taken into account, then the geometrical moments of one block b, are given by

$$m_{pq}^b = \sum_{x=x_{1b}}^{x_{2b}} \sum_{y=y_{1b}}^{y_{2b}} x^p y^q = \left(\sum_{x=x_{1b}}^{x_{2b}} x^p \right) \left(\sum_{y=y_{1b}}^{y_{2b}} y^q \right) \qquad (4)$$

Using the rectangular form appeared within the block, the computational effort, which is characterized by the complexity $O(N^2)$ for the calculation of moments using (2), is reduced to $O(N)$ for the calculation of moments using (4). For the computation of (4), it is adequate to calculate the following summations of the powers of x and y :

$$S_{x_{1b},x_{2b}}^p = \sum_{x=x_{1b}}^{x_{2b}} x^p ,$$
$$S_{y_{1b},y_{2b}}^q = \sum_{y=y_{1b}}^{y_{2b}} y^q , \qquad x,y,p,q \in Z \qquad (5)$$

Moreover, taking into account the known formulae:

$$S_{1,n}^1 = \frac{n(n+1)}{2} , \quad S_{1,n}^2 = \frac{n(n+1)(2n+1)}{6} ,$$
$$S_{1,n}^3 = \frac{n^2(n+1)^2}{4} ,$$
$$S_{1,n}^4 = \frac{n(n+1)(2n+1)(3n^2+3n+1)}{30} \qquad (6)$$
$$\binom{m+1}{1} S_{1,n}^1 + ... + \binom{m+1}{m} S_{1,n}^m = (n+1)^{m+1} - (n+1)$$

where $m,n \in Z$ and $\binom{i}{j} = \frac{i!}{j!(i-j)!}$ are the combinations of i objects, taken j each time, it is concluded that the summation S_{x_1,x_2}^p can be directly calculated as $S_{x_{1b},x_{2b}}^p = S_{1,x_{2b}}^p - S_{1,x_{1b}-1}^p$. The summation $S_{y_{1b},y_{2b}}^q$ is computed in a similar manner. Obviously, the computation of the moments of a block using the above analytical formulae (6), is independent of the size of the block. It has been proved [8], that the computational complexity of the proposed technique is in the order of $O(L^2)$, where $(L-1,L-1)$ is the order of the moments to be computed.

III.2. Computation of the 2-D moments on the SLAP

The computation of the moments of a block represented binary images on the SLAP, is executed in two discrete steps:

Step 1:

The first step does not involve any communication among the processors and therefore is executed very fast. Let the n blocks that constitute the image f, where usually n is less than the number

of the processors N, i.e $n<N$. A SLAP architecture with N processors, as the one described above is sufficient for the computation of the 2-D statistical moments of the image. Each block b_k , k=0,1,...,n-1 is loaded as input to the processor P_k , in four time units. The corresponding values of the moments until the order (L-1, L-1) are computed according to the analytical formulae (3), (6) and stored in the local memory of the processor. If $n>N$, then this process is repeated for the rest of the blocks and each processor adds the moment values of each block and holds this summation.

The above procedure requires $4L$ power calculations, $2L^2 - L$ multiplications and $L^2 - L$ additions [8]. The corresponding time units depends on the type of the general purpose processor, but without loss it can be assumed that $O(2L^2)$ time units are required. If $n>N$, then the relevant time units are multiplied by the factor modulo(n/N). In the most of real world images it holds that $n < N$.

Step 2:

In the second step each processor broadcasts in its left immediate processor the value of the moment with order (0,0), while the first processor from the left adds the values to the corresponding moment. This process requires N time units. The same process is repeated for all the moments and finally the first left processor holds the moment values of the block represented binary image.

Finally the transmition of the values of one moment to the first processor from the left requires $O(N)$ time units and the transmission of the values of all the moments requires $O(L^2N)$ time units. In typical pattern recognition applications the moments usually are calculated up to the order (4,4), since the higher order moments are very sensitive to noise. Therefore for L=5, $L^2 < N$ and the computational compexity is less than $O(N^2)$.

The whole computational complexity for the moments computation on the SLAP, is the summation of the complexity of the two steps. Therefore, the whole process requires $\mathrm{MOD}(n/N)O(2L^2)+O(L^2N)$ time units and under the reasonable assumptions that $n < N$ and that $L^2 < N$, this value is less than $O(N^2)$ time units.

IV. CONCLUSIONS

In this paper an efficient algorithm for the computation of the 2-D geometrical moments of a block represented binary image, on the Scan Line Array Processor in real-time has been presented. It can be shown that other set of moments (central, normalized central, moment invariants) may be also computed on block represented images and therefore their real-time computation on the SLAP is a direct extension of the work presented here.

REFERENCES

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

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

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

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

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

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

[7] I.M. Spiliotis, D.A. Mitzias and B.G. Mertzios, "A skeleton-based hierarchical system for learning and recognition", *Proceedings of MTNS 93, International Symposium on the Mathematical Theory of Networks and Systems*, Regensburg, Germany, August 2-6, pp. 873-878, 1993.

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

[9] A.I. Fisher and P.T. Highnam, "Real-time image processing on scan line array processors", *IEEE Computer Society Workshop on Computer Architectures for Pattern Analysis and Image Database Management*, pp. 484-489, 1985.

[10] D. Helman and J. Jaja, "Efficient image processing algorithms on the scan line array processor", *IEEE Trans. PAMI*, vol. 17, no. 1, pp. 47-56, 1995.

[11] S. Knight, D. Chin, H. Taylor and J. Peters, "The Sarnoff Engine: a massively parallel computer for high definition system simulation", *Proc. of Application Specific Array Processors*, pp. 342-356, 1992.

[12] A.L. Fisher, "Scan Line Array Processors for image computation", *Intern. Conf. on Computer Architectures*, pp. 348-345, 1986.