A New Computation of Shape Moments via Quadtree Decomposition ⋆

Chin-Hsiung Wu[1], Shi-Jinn Horng[1,2], Pei-Zong Lee[2], Shung-Shing Lee[3], and Shih-Ying Lin[3]

[1] National Taiwan University of Science and Technology, Taipei, Taiwan, R. O. C.
`horng@mouse.ee.ntust.edu.tw`
[2] Institute of Information Science, Academia Sinica, Taipei, Taiwan, R. O. C.
[3] Fushin Institute of Technology and Commerce, I-Lain, Taiwan, R. O. C.

Abstract. The main contribution of this paper is in designing an optimal and/or optimal speed-up algorithm for computing shape moments. We introduce a new technique for computing shape moments. The new technique is based on the quadtree representation of images. We decompose the image into squares, since the moment computation of squares is easier than that of the whole image. The proposed sequential algorithm reduces the computational complexity significantly. By integrating the advantages of both optical transmission and electronic computation, the proposed parallel algorithm can be run in $O(1)$ time. In the sense of the product of time and the number of processors used, the proposed parallel algorithm is time and cost optimal and achieves optimal speed-up.

1 Introduction

Moments are widely used in image analysis, pattern recognition and low-level computer vision [6]. The computation of moments of a two-dimensional (2-D) image involves a significant amount of multiplications and additions in a direct method. Previously, some fast algorithms for computing moments had been proposed using various computation methods [2, 3, 5, 8, 14, 15]. For an $N \times N$ binary image, Chung [2] presented a constant time algorithm for computing the horizontal/vertical convex shape's moments of order up to 3 on an $N \times N$ reconfigurable mesh. Chung's algorithm is unsuitable for complicated objects. In this paper, we will develop a more efficient algorithm to overcome the disadvantage of Chung's algorithm.

The array with a reconfigurable optical bus system is defined as an array of processors connected to a reconfigurable optical bus system whose configuration can be dynamically changed by setting up the local switches of each processor, and messages can be transmitted concurrently on a bus in a pipelined fashion.

⋆ This work was partially supported by the National Science Council under the contract no. NSC-89-2213-E011-007. Part of this work was carried out when the second author was visiting the Institute of Information Science, Academia Sinica, Taipei, Taiwan, July - December 1999.

More recently, two related models have been proposed, namely the array with reconfigurable optical buses (AROB) [10] and linear array with a reconfigurable pipelined bus system (LARPBS) [9]. The AROB model is essentially a mesh using the basic structure of a classical reconfigurable network (LRN) [1] and optical technology. A 2-D AROB of size $M \times N$, denoted as 2-D $M \times N$ AROB, contains $M \times N$ processors arranged in a 2-D grid. The processor with index $(i_1,\ i_0)$ is denoted by $P_{i_1,\ i_0}$. For more details on the AROB, see [10].

The main contribution of this paper is in designing an optimal speed-up algorithm for computing the 2-D shape moments. The idea of our algorithm is based on the summation of the contribution of each quadtree node where each quadtree node represents a square region. We first represent the image by quadtree decomposition. After that, the image is divided into squares. Then we derive the relationship between the quadtree and the computation of shape moments. Finally, using this representation, an efficient sequential algorithm (SM) and an optimal parallel algorithm (PSM) for shape moment computations are developed. For a constant c, $c \geq 1$, the proposed algorithm PSM can be run in $O(1)$ time using $N \times N^{1+\frac{1}{c}}$ processors when the input image is complicated. If the image is simple (i.e., the image can be represented by a few quadtree nodes), the proposed algorithm PSM can be run in $O(1)$ time using $N \times N$ processors. In the sense of the product of time and the number of processors used, the proposed algorithm PSM is time and cost optimal and achieves optimal speed-up.

2 Basic Data Manipulation Operations

Given N integers a_i with $0 \leq a_i < N$, $0 \leq i < N$, let *sum* stand for

$$\sum_{i=0}^{N-1} a_i. \tag{1}$$

For computing Eq. (1), Pavel and Akl [11] proposed an $O(1)$ time algorithm on a 2-D $N \times \log N$ AROB. In the following, we will use another approach to design a more flexible algorithm for this problem on a 1-D AROB using $N^{1+\frac{1}{c}}$ processors, where c is a constant and $c \geq 1$. Since $a_i < N$ and $0 \leq i < N$, each digit has a value ranging from 0 to $\omega - 1$ for the radix-ω system and a ω-ary representation $\cdots m_3 m_2 m_1 m_0$ is equal to $m_0\omega^0 + m_1\omega^1 + m_2\omega^2 + m_3\omega^3 \cdots$. The maximum of *sum* is at most $N(N-1)$. With this approach, a_i and *sum* are equivalent to

$$a_i = \sum_{k=0}^{T-1} m_{i,\ k}\ \omega^k, \tag{2}$$

$$sum = \sum_{l=0}^{U-1} S_l\ \omega^l, \tag{3}$$

where $T = \lfloor \log_\omega N \rfloor + 1$, $0 \leq i < N$, $U = \lfloor \log_\omega N(N-1) \rfloor + 1$, and $0 \leq m_{i,\ k}$, $S_l < \omega$.

As $sum = \sum_{i=0}^{N-1} \sum_{k=0}^{T-1} m_{i,\ k}\ \omega^k = \sum_{k=0}^{T-1} \sum_{i=0}^{N-1} m_{i,\ k}\ \omega^k$, let d_k be the sum of N coefficients $m_{i,\ k}$, $0 \le i < N$, which is defined as

$$d_k = \sum_{i=0}^{N-1} m_{i,\ k}, \tag{4}$$

where $0 \le k < T$. Then sum can be also formulated as

$$sum = \sum_{k=0}^{T-1} d_k\ \omega^k, \tag{5}$$

where $0 \le d_k < \omega N$.
Let $C_0 = 0$ and $d_u = 0$, $T \le u < U$. The relationship between Eqs. (3) and (5) is described by Eqs. (6)-(8).

$$e_t = C_t + d_t,\ 0 \le t < U, \tag{6}$$

$$C_{t+1} = e_t\ \ \mathbf{div}\ \ \omega,\ 0 \le t < U, \tag{7}$$

$$S_t = e_t\ \ \mathbf{mod}\ \omega,\ 0 \le t < U, \tag{8}$$

where e_t is the sum at the t^{th} digit position and C_t is the carry to the t^{th} digit position. Hence, S_t of Eq. (8) corresponds to the coefficient of sum of Eq. (3) under the radix-ω system. Since the carry to the t^{th} digit position of e_t is not greater than N, we have $C_t \le N$, $0 \le t < U$. Since $sum \le N(N-1)$, the number of digits representing sum under radix-ω is not greater than U, where $U = \lfloor \log_\omega N(N-1) \rfloor + 1$. Therefore, instead of computing Eq. (1), we first compute the coefficient $m_{i,\ k}$ for each a_i. Then each S_t can be computed by Eqs. (4), (6)-(8). Finally, sum can be computed by Eq. (3). For more details, see [13].

Lemma 1. *The N integers each of size $O(\log N)$-bit, can be added in $O(1)$ time on a 1-D $N^{1+1/c}$ AROB for a constant c and $c \ge 1$.*

Consequently, given an $N \times N$ integer matrix each of size $O(\log N)$-bit, the sum of these N^2 integers can be computed by the following three steps. First, apply Lemma 1 to compute the partial sum of each row in parallel. Then, route the partial sums located on the first column to the first row. Finally, apply Lemma 1 to accumulate these N partial sums.

Lemma 2. *The N^2 integers each of size $O(\log N)$-bit, can be added in $O(1)$ time on a 2-D $N \times N^{1+\frac{1}{c}}$ AROB for a constant c and $c \ge 1$.*

3 The Quadtree Decomposition

The quadtree is constructed by recursively decomposing the image into four equal-sized quadrants in top-down fashion. Given an $N \times N$ image ($N = 2^d$ for some d), the quadtree representation of it is a tree of degree four which can be defined as follows. The root node of the tree represents the whole image. If the whole image has only one color, we label that root node with that color and stop; otherwise, we add four children to the root node, representing the four quadrants of the image. Recursively we apply this process for each of the four nodes, respectively. If a block has a constant color, then its corresponding node is a leaf node; otherwise, its node has four children.

Recently, Lee and Horng *et al.* [7] addressed a constant time quadtree building algorithm for a given image based on a specified space-filling order.

Lemma 3. *[7] The quadtree of an $N \times N$ image can be constructed in constant time on an $N \times N$ AROB.*

Let the data structure of a quadtree node consist of four fields r, c, I and sz, respectively. The row and column coordinates of the top-left corner of a quadtree node are represented by r and c, the image color of it is represented by I and sz represents the index of the block size of a quadtree node; if the block size is 4^s then sz is s. For a binary image, the third field I can be omitted. In this paper, only the leaves of the quadtree which represent black blocks are useful for computing shape moments; the non-terminal nodes are omitted.

4 Computing Shape Moments

For a 2-D digital image $A = a(x, y)$, $1 \leq x, y \leq N$, the moment of order (p, q) is defined as:

$$m_{pq} = \sum_{x=1}^{N} \sum_{y=1}^{N} x^p y^q a(x, y), \tag{9}$$

where $a(x, y)$ is an integer representing the intensity function (gray level or binary value) at pixel (x, y).

Delta algorithm [15] and Chung's algorithm [2] were based on the summation of the contribution of each row. Ours is based on the summation of the contribution of each quadtree node where each quadtree node represents a square region. For an object represented by a quadtree with α leaves, exactly α non-overlapped squares, Q_1, Q_2,$\cdots$, Q_α, are defined. From the definition of moments, computing the double summations in Eq. (9) of a square is easier than that of an arbitrary shape. Thus, compared to a direct method, the computational complexity can be reduced significantly.

Since the double summations in Eq. (9) are linear operations, the moments of the whole object can be derived from the summations of the moments of these squares. The (p, q)th order moments of theses squares can be computed as follows. From the data structure of quadtree nodes, we can easily find the location of the four corners of the corresponding square. For a square Q_i, assume the coordinates of its top-left corner are (r, c) and its size is 4^s. Let $u = 2^s$, denote the length of each side of the square. Then the coordinates of the other three corners of Q_i are $(r+u-1, c)$, $(r, c+u-1)$ and $(r+u-1, c+u-1)$, respectively.

For a binary digital image, the moment computation of a quadtree node Q_i reduces to the separable computation

$$m_{pq,i} = \sum_{x=r}^{r+u-1} x^p \sum_{y=c}^{c+u-1} y^q = \sum_{x=r}^{r+u-1} x^p h_{q,i} = h_{q,i} \sum_{k=0}^{u-1} (r+k)^p = g_{p,i} \cdot h_{q,i}, \quad (10)$$

where $g_{p,i}$ and $h_{q,i}$ are the p-order and q-order moments for dimension x and dimension y, respectively and they are defined as:

$$\begin{aligned} g_{p,i} &= \sum_{x=r}^{r+u-1} x^p = \sum_{k=0}^{u-1} (r+k)^p, \\ h_{q,i} &= \sum_{y=c}^{c+u-1} y^q = \sum_{k=0}^{u-1} (c+k)^q. \end{aligned} \quad (11)$$

Similarly, the corresponding moments of other quadtree nodes can be obtained from Eqs. (10)-(11) by replacing r, c and u with their corresponding values since they are also represented as squares. Thus, the 2-D shape moments of order (p, q) can be obtained by summing up the corresponding moments of all α square regions:

$$m_{pq} = \sum_{i=1}^{\alpha} m_{pq,i}. \quad (12)$$

Let us conclude this section by stating a sequential algorithm for computing shape moments from the above derivations.

Algorithm SM;

1: For each quadtree node Q_i, compute the 2-D shape moments $m_{pq,i}$, $1 \le i \le \alpha$, according to Eqs. (10)-(11).

2: Compute the 2-D shape moments m_{pq} by summing up $m_{pq,i}$, $1 \le i \le \alpha$, according to Eq. (12).

Theorem 1. *Given an $N \times N$ binary image A, the 2-D shape moments up to order 3 can be computed in $O(\alpha)$ time on a uniprocessor, where α is the number of quadtree nodes.*

Proof : The correctness of this algorithm directly follows from Eqs. (9)-(12). The time complexity is analyzed as follows. Step 1 and 2 each take $O(\alpha)$ time, where α is the number of quadtree nodes. Hence, the time complexity is $O(\alpha)$.

If we consider an $N \times N$ binary image whose entire image has only 1-valued, the comparison of the computational complexity in computing all the moments of order up to $p + q \le 3$ is shown in Table 1. From Table 1, we see that the proposed method reduces the computational computation significantly.

In addition to the computing operations shown in Table 1, contour following, which needs a few comparison operations per pixel, is required for all the non-direct methods to identify the shape of all objects and it takes $O(N^2)$ time. Our algorithm also needs a preprocessing time to create the quadtree nodes for the given image and this can be done in $O(N^2)$ time by the optimal quadtree construction algorithm proposed by Shaffer and Samet [12].

Table 1: Comparison of computational complexity for shape moment methods.

Method	Direct [6]	Delta [15]	Green's [8]	Integral [3]	This paper
Multiplication	$20N^2$	$25N$	0	$8N$	8
Addition	$10N^2$	N^2+6N	$128N$	$22N$	22

5 Parallel Moment Computation Algorithm

From Eqs. (9)-(12), the algorithm for computing 2-D shape moments m_{pq} includes the following three steps. First build the quadtree for the given image. Then for each quadtree node, compute its corresponding 2-D shape moments by multiplying the two dimensional moments derived from Eqs. (10)-(11). Finally the 2-D shape moments can be obtained by summing up the corresponding moments which were computed by Step 2.

Initially, assume that the given image A is stored in the local variable $a(i,\ j)$ of processor $P_{i,\ j}$, $1 \leq i,\ j \leq N$. Finally, the results are stored in the local variable $m_{pq}(1,1)$ of processor $P_{1,1}$. Following the definitions of moments, quadtree, and the relationship between them, the detailed moments algorithm (PSM) is listed as follows.

Algorithm PSM;

```
1: Apply Lemma 3 to build the quadtree for the given image. After that, the
   results $Q_i$, $1 \leq i \leq \alpha$, are stored in local variable $Q(x,y)$ in processor $P_{x,\ y}$,
   where $i = xN + y$.
2: //For each quadtree node computes its 2-D shape moments. //
  2.1: For each quadtree node $Q_i$, $1 \leq i \leq \alpha$, computes its 1-D shape mo-
       ments $g_p(x,y)$ and $h_q(x,y)$ of dimension $x$ and dimension $y$ respectively
       according to Eq. (11).
  2.2: For each quadtree node $Q_i$, $1 \leq i \leq \alpha$, compute its 2-D shape moments
       by computing Eq. (10) (i.e., $m_{pq,i}(x,y) = g_p(x,y) \times h_q(x,y)$).
3: Compute the 2-D shape moments $m_{pq}$ by summing up $m_{pq,i}$, $1 \leq i \leq \alpha$, using
   Lemmas 1 or 2 according to the value of $\alpha$. After that, the 2-D moments
   $m_{pq}$ are stored in the local variable $m_{pq}(1,1)$ of processor $P_{1,\ 1}$.
```

Theorem 2. *Given an $N \times N$ binary image A, the 2-D shape moments up to order 3 can be computed in $O(1)$ time either on an $N \times N$ AROB if A is simple (i.e., α is bounded by $O(N)$), or on an $N \times N^{1+\frac{1}{c}}$ AROB for a constant c and $c \geq 1$ if A is complicated.*

Proof : The time complexity is analyzed as follows. Step 1 takes $O(1)$ time using $N \times N$ processors by Lemma 3. Step 2 takes $O(1)$ time. Step 3 takes $O(1)$ time using $N \times N$ or $N \times N^{1+\frac{1}{c}}$ processors by Lemmas 1 and 2. Hence, the time complexity is $O(1)$.

For computing high order shape moments, Steps 2 and 3 will take $\max\{p,q\}$ times. If both p and q are constant, then the expression for $g_{p,i}$ (or $h_{q,i}$) defined in Eq. (11) will have a constant number of terms with a constant number of powers. Therefore, the results of Theorem 2 can be extended.

6 Concluding Remarks

In this paper, we introduce a new technique based on the quadtree decomposition for computing shape moments. The quadtree decomposition divides the image into squares, where the number of squares is dependant on the image complexity. In the most application, the $N \times N$ image can be decomposed into $O(N)$ squares by quadtree decomposition. As a result, the shape moments can be parallelized and computed in $O(1)$ time on an $N \times N$ AROB.

References

1. Ben-Asher, Y., Peleg, D., Ramaswami, R., Schuster, A.: The Power of Reconfiguration. Journal of Parallel and Distributed Computing **13** (1991) 139–153
2. Chung, K.-L.: Computing Horizontal/vertical Convex Shape's Moments on Reconfigurable Meshes. Pattern Recognition **29** (1996) 1713-1717
3. Dai, M., Batlou, P., Najim, M.: An Efficient Algorithm for Computation of Shape Moments from Run-length Codes or Chain Codes. Pattern Recognition **25** (1992) 1119-1128
4. Guo, Z., Melhem, R. G., Hall, R. W., Chiarulli, D. M., Levitan, S. P.: Pipelined Communications in Optically Interconnected Arrays. Journal of Parallel and Distributed Computing **12** (1991) 269–282
5. Hatamian, M.: A Real Time Two-dimensional Moment Generation Algorithm and Its Single Chip Implementation. IEEE Trans. ASPP **34** (1986) 546-553
6. Hu, M.-K.: Visual Pattern Recognition by Moment Invariants. IRE Trans. Inform. Theory **IT-8** (1962) 179-187
7. Lee, S.-S., Horng, S.-J., Tsai, H.-R., Tsai, S.-S.: Building a Quadtree and Its Applications on a Reconfigurable Mesh. Pattern Recognition **29** (1996) 1571-1579
8. Li, B.-C., Shen, J.: Fast Computation of Moment Invariants. Pattern Recognition **24** (1991) 8071-813
9. Pan, Y., Li, K.: Linear Array with a Reconfigurable Pipelined Bus System— Concepts and Applications. Information Sciences – An Int. Journal **106** (1998) 237-258
10. Pavel, S., Akl, S. G.: On the Power of Arrays with Reconfigurable Optical Bus. Proc. Int. Conf. Parallel and Distributed Processing Techniques and Applications (1996) 1443-1454
11. Pavel, S., Akl, S. G.: Matrix Operations Using Arrays with Reconfigurable Optical Buses. Parallel Algorithms and Applications **8** (1996) 223-242
12. Shaffer, C. A., Samet, H.: Optimal Quadtree Construction Algorithms. Computer Vision, Graphics, Image processing **37** (1987) 402-419
13. Wu, C.-H., Horng, S.-J., Tsai, H.-R.: Template Matching on Arrays with Reconfigurable Optical Buses. Proc. Int. Symp. Operations Research and its Applications (1998), 127-141
14. Yang, L., Albregtsen, F.: Fast and Exact Computation of Cartesian Geometric Moments Using Discrete Green's Theorem. Pattern Recognition **29** (1996) 1061-1073
15. Zakaria, M. F., Zsombor-Murray, P. J. A., Kessel, J. M. H. H.: Fast Algorithm for the Computation of Moment Invariants. Pattern Recognition **20** (1987) 639-643