An Optimal Parallel Algorithm for Computing Moments on Arrays with Reconfigurable Optical Buses *

Chin-Hsiung Wu† Shi-Jinn Horng† Horng-Ren Tsai‡ Jinn-Fu Lin† Tsrong-Lay Lin§
†Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, R. O. C.
E-mail:horng@mouse.ee.ntust.edu.tw
‡Department of Information Management, Chinese College of Commerce, Taichung, Taiwan, R. O. C.
§SKuang Wu Institue of Technology and Commerce, Taipei, Taiwan, R. O. C.

Abstract

Computing the moments of a two-dimensional (2-D) image involves a significant amount of multiplications and additions in a direct method. In this paper, we use the suffix sums to compute the 2-D moments instead of using a direct method. This method can reduce the number of multiplications tremendously. By integrating the advantages of both optical transmission and electronic computation, the 2-D moments can be computed in constant time on a 2-D arrays with reconfigurable optical buses (AROB). This result achieves optimal speed-up.

1 Introduction

Moments are by far the most popular descriptors for image regions and boundary segments. Many important geometric features of an image can be determined from its moments. Hu [6] proposed a set of moment invariants based on the 10 low order moments. These moment invariants are simple functions of moments and independent of scaling, translation and rotation. To compute the 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 methods [2, 3, 5, 7, 12, 13, 14, 15, 16, 17, 18]. Since the moments of a gray level image are more widely used in many applications, such as texture analysis, we focus on the moment computation of gray level images in this paper.

Given an $N \times N$ image, some parallel algorithms were proposed for computing moments of the image. Reeves [13, 14] presented two parallel algorithms for this problem using the Direct algorithm on mesh connected computers (MCCs). Chen [2] presented parallel algorithms for computing the 16 low order moments based on a so-called cascade-partial-sum technique on MCCs. Chen's algorithms run in $O(N)$ time on a linear array of size N and run in $O(\log N)$ time on a 2-D arrays of size $N \times N$, respectively. Recently, Cheng et al. [3] proposed two special VLSI architectures for computing the 2-D moments of order (p, q). For the 1-D structure, it took $O(\max(p, q) \times N)$ time using $N+1$ processors. For the 2-D structure, it took $O(N)$ time using $N(N+1)$ processors.

The main contribution of this paper is in designing an optimal speed-up algorithm for computing the 2-D moments on a 2-D AROB. We first derive the relationships between suffix sums and moments. Then using these relationships, an optimal speed-up algorithm for moment computation is developed on a 2-D AROB. When the domain of the image is $O(\log N)$-bit integer, for a constant c, $c \geq 1$, the proposed algorithm can be run in $O(1)$ time using $N \times N^{1+\frac{1}{c}}$ processors. To the best of our knowledge, there were no algorithms before which could solve this problem in constant time and/or provide such a performance on a 2-D array architecture. Clearly, this result is better than the previously known algorithms proposed in the literature [2, 3, 13, 14].

The rest of this paper is organized as follows. We give a brief introduction to the AROB model in Section 2. Section 3 describes some basic operations which will

*This work was supported by the National Science Council under the contract no. NSC-89-2213-E011-007.

be used in the parallel algorithm. Section 4 develops our algorithm for computing 2-D moments. Finally, some concluding remarks are included in the last section.

2 The Computation Model

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. Two related models using reconfigurable optical buses have been proposed, namely the array with reconfigurable optical buses (AROB) [10] and linear array with a reconfigurable pipelined bus system (LARPBS) [8, 9]. Due to unidirectional signal propagation and predictable delay of the signal per unit length, the optical buses enable synchronized concurrent access in a pipelined fashion.

The AROB model is a powerful computation model which incorporates some of the advantages and characteristics of reconfigurable meshes [1] and meshes with optical buses [10]. A linear array with pipelined optical buses (LAPPB) [4] of size N contains N processors connected to the optical bus with two couplers. One is used to write data on the upper (transmitting) segment of the bus and the other is used to read the data from the lower (receiving) segment of the bus. A 1-D AROB extends the capabilities of the LAPPB by permitting each processor to connect to the bus through a pair of switches. Each processor with a local memory is identified by a unique index denoted as P_{i_0}, $0 \leq i_0 < N$, and each switch can be set to cross or straight by the local processor. As for the properties of the optical buses, the message propagates unidirectionally from right to left on the upper segment and from left to right on the lower segment. Each processor uses a set of control registers to store information needed to control the transmission and reception of messages by that processor. An example for a linear AROB of size 5 is shown in Figure 1(a). Two interesting switch configurations derivable from a processor of an LAROB are also shown in Figure 1(b).

A 2-D AROB of size $M \times N$ contains $M \times N$ processors arranged in a 2-D grid. Each processor is identified by a unique 2-tuple index $(i_1,\ i_0)$, $0 \leq i_1 < M$, $0 \leq i_0 < N$. The processor with index $(i_1,\ i_0)$ is denoted by $P_{i_1,\ i_0}$. Each processor has four I/O ports, denoted by $-S_j$, $+S_j$, $0 \leq j < 2$, to be connected with a reconfigurable optical bus system. The interconnection among the four ports of a processor can be configured during the execution of algorithms. For more details

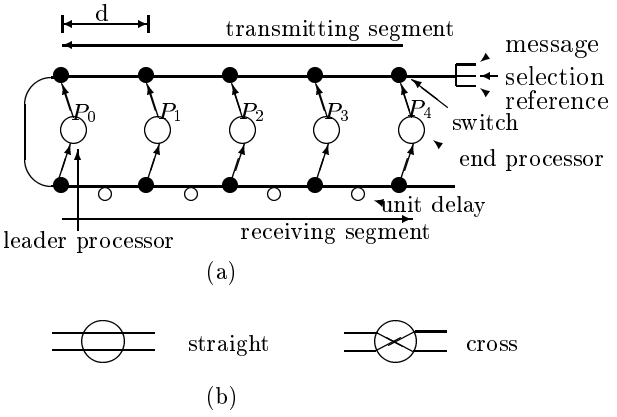

Figure 1: (a) An LAROB of size 5. (b) The switch states.

on the AROB, see [10].

3 Basic Operations

For Boolean operation, there was a result derived by Pavel and Akl [11].

Lemma 1 *[11] Given N Boolean data each with either 0 or 1, the binary prefix sums of these N bits can be computed in $O(1)$ time on a 1-D N AROB.*

Next, we propose the efficient algorithms for computing integer suffix sums. Given N integers a_i, $0 \leq a_i < N$, $0 \leq i < N$, the suffix sums of these N integers is defined as

$$rps_j = \sum_{i=j}^{N-1} a_i,\ 0 \leq j < N. \tag{1}$$

Using the radix-ω system technique, rps_j of Eq. (1) can be computed more flexibly. Since $a_i < N$ and $0 \leq i < N$, each digit has a value ranging from 0 to $\omega - 1$ and a ω-ary representation $\cdots m_3 m_2 m_1 m_0$ is equal to $m_0\omega^0 + m_1\omega^1 + m_2\omega^2 + m_3\omega^3 \cdots$ in normal ω-ary representation. The maximum of rps_j is at most $N(N-1)$. With this approach, a_i and rps_j of Eq. (1) are equivalent to

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

$$rps_j = \sum_{l=0}^{\beta-1} n_{j,\ l}\ \omega^l, \tag{3}$$

where $\alpha = \lfloor \log_\omega N \rfloor + 1$, $0 \leq i < N$, $\beta = \lfloor \log_\omega N(N-1) \rfloor + 1$, and $0 \leq m_{i,\ k},\ n_{j,\ l} < \omega$.

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

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

where $0 \le k < \alpha$. Then rps_j can be also formulated as

$$rps_j = \sum_{k=0}^{\alpha-1} d_{j,\,k}\ \omega^k, \tag{5}$$

where $0 \le d_{j,\,k} \le (\omega-1)(N-j)$.
Let $cy_{j,\,0} = 0$ and $d_{j,\,u} = 0$, $\alpha \le u < \beta$. The relationship between Eqs. (3) and (5) is described by Eqs. (6)-(10).

$$sum_{j,\,t} = cy_{j,\,t} + d_{j,\,t},\ 0 \le t < \beta, \tag{6}$$

$$cy_{j,\,t+1} = sum_{j,\,t}\ \ \mathbf{div}\ \ \omega,\ 0 \le t < \beta, \tag{7}$$

$$n_{j,\,t} = sum_{j,\,t}\ \ \mathbf{mod}\ \ \omega,\ 0 \le t < \beta, \tag{8}$$

where $sum_{j,\,t}$ is the sum at the tth digit position and $cy_{j,\,t}$ is the carry to the tth digit position. Hence, $n_{j,\,t}$ is the coefficient of rps_j under the radix-ω system. Since $cy_{j,\,0} = 0$, then $cy_{j,\,1} \le (\omega-1)(N-j)/\omega \le N-j$, then $cy_{j,\,2} \le [(\omega-1)(N-j)/\omega + (\omega-1)(N-j)]/\omega \le N-j-(N-j)/\omega^2$, then $cy_{j,\,t+1} \le N-j-(N-j)/\omega^{t+1}$. Therefore, the carry to the $(t+1)^{th}$ digit position is not greater than $N-j$, we have $cy_{j,\,t+1} \le N-j$, $0 \le t < \beta$. Also $cy_{j,\,t+1}$ is no larger than $cy_{j+1,\,t+1}+1$ because of $m_{j,\,t} < \omega$. Let $q_{i,\,t+1}$ be a binary sequence representing the binary quotient sequence of $cy_{j,\,t+1}$. Then, $cy_{j,\,t+1}$ can be expressed as

$$cy_{j,\,t+1} = \sum_{i=j}^{N-1} q_{i,\,t+1}. \tag{9}$$

From Eq. (9), we obtain

$$cy_{j,\,t+1} = cy_{j+1,\,t+1} + q_{j,\,t+1}. \tag{10}$$

That is, $q_{j,\,t+1} = cy_{j,\,t+1} - cy_{j+1,\,t+1}$. Since $rps_j \le N(N-1)$, the number of digits representing rps_j under radix-ω is not greater than β, where $\beta = \lfloor \log_\omega N(N-1) \rfloor + 1$. Therefore, instead of computing Eq. (1), we first compute the coefficient $m_{i,\,k}$ for each a_i. Then each $n_{j,\,t}$ can be computed by Eqs. (4), (6)-(10). Finally, rps_j can be computed by Eq. (3). For the sake of readability, let $P'_{i,\,j}$, $0 \le i < N$, $0 \le j < \omega$, denote the 2-D logical processor corresponding to the 1-D physical processor $P_{i\omega+j}$. The suffix sums of N integers each of size $(\log N)$-bit can be computed in a constant time on a 1-D ωN AROB. The detailed algorithm (SSA) is described in the following. Initially, a_i is allocated to the local variable $a(i,0)$ of processor $P'_{i,\,0}$, $0 \le i < N$. Finally, the suffix sums of each a_i is stored to the local variable $rps(i,0)$ of processor $P'_{i,\,0}$, $0 \le i < N$.

Procedure SSA$(a,\ rps)$

1: Processor $P'_{i,\,0}$, $0 \le i < N$, copies $a(i,0)$ (*i.e.*, a_i)to processor $P'_{i,\,j}$, $0 \le j < \omega$.

2: Processor $P'_{i,\,0}$, $0 \le i < N$, sets $q(i,0)[0] = 0$.

3: **for** $k = 0$ **to** β **do** Steps 3.1 - 3.6

3.1: Processor $P'_{i,\,j}$, $0 \le i < N$, $0 \le j < \omega$, computes $m(i,\ j)[k]$ from $a(i,j)$ by using Eq. (2).

3.2: Processor $P'_{i,\,\omega-j-1}$, $0 \le i < N$, $0 \le j < \omega$, sets $b(i,\ \omega-j-1) = 1$ if $j < m(i,\ \omega-j-1)[k]$; $b(i,\ \omega-j-1) = 0$, otherwise.

3.3: Processor $P'_{i,\,0}$, $0 \le i < N$, sets $b(i,\ 0) = q(i,0)[k]$.

3.4: Apply Lemma 1 to compute the binary suffix sums on $b(i,\ j)$ and store the result to the local variable $sum(i,\ 0)[k]$ of processor $P'_{i,\,0}$.

3.5: Processor $P'_{i,\,0}$ computes $cy(i,\ 0)[k+1]$ (*i.e.*, $cy_{i,\,k+1}$ in Eq. (7)) and $n(i,\ 0)[k]$ from $sum(i,\ 0)[k]$ by using Eqs. (7) and (8), respectively.

3.6: Processor $P'_{i,\,0}$, $1 \le i < N$, computes $q(i,\ 0)[k+1] := cy(i,\ 0)[k+1] - cy(i+1,\ 0)[k+1]$, where $cy(N,\ 0)[k+1] = 0$.

4: // Compute Eq. (3) to obtain the suffix sums. // $P'_{i,\,0}$ computes the final $rps(i,0)$ from $n(i,\ 0)[k]$ by using Eq. (3).

End{SSA}

Lemma 2 *Given N integers each of size $O(\log N)$-bit, the suffix sums of these N integers can be computed in $O(\beta)$ time on a 1-D ωN AROB for $\omega \ge 2$.*

Proof: The correctness of procedure SSA directly follows from Lemma 1 and Eqs. (1)-(10). Step 3.6 is used to implement Eqs. (9) and (10). The time complexity is analyzed as follows. Steps 1 and 2 each take $O(1)$ time. Each iteration of Step 3 takes $O(1)$ time, and the number of iterations will be repeated β times. Hence, the total time complexity of Step 3 is $O(\beta)$. Finally, Step 4 takes $O(\beta)$ time. Hence, the total time complexity of the proposed algorithm is $O(\beta)$. □

For simplicity, assume $\omega = N^{\frac{1}{c}}$, where c is a constant and $c \ge 1$. Then $\beta = \lfloor \log_\omega N(N-1) \rfloor + 1 = \lfloor 2c \rfloor + 1$ is also a constant. This leads to the following corollary.

Corollary 1 *Given N integers each of size $O(\log N)$-bit, the suffix sums of these N integers can be computed in $O(1)$ time on a 1-D $N^{1+\frac{1}{c}}$ AROB for some fixed c and $c \geq 1$.*

Procedure SSA can be used to compute the suffix sums of N integers, each ranging from 0 to N^k. Such an integer can be represented as a sequence of (at most) k radix N digits. Thus, the problem can be reduced to repeat procedure SSA for every digit in the radix N representation, separately. By Corollary 1, this takes $O(k)$ time on a 1-D $N^{1+\frac{1}{c}}$ AROB. Hence, we have the following corollary.

Corollary 2 *Given N integers each of size $O(\log N^k)$-bit, the suffix sums of these N integers can be computed in $O(k)$ time on a 1-D $N^{1+\frac{1}{c}}$ AROB for some fixed c and $c \geq 1$.*

4 Computing 2-D Moments

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

$$m_{pq} = \sum_{x=1}^{N} \sum_{y=1}^{N} x^p y^q a(x, y), \tag{11}$$

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

For the sake of simplicity, we first consider a 1-D digital image $a(x)$; the moment of it with order p is

$$m_p = \sum_{x=1}^{N} x^p a(x). \tag{12}$$

Now we define the high order suffix sum functions S_i, $0 \leq i \leq p + 1$ as follows. Initially, the zero-order suffix sums is defined as

$$S_0(j) = a(j),\ 1 \leq j \leq N, \tag{13}$$

and the one-order suffix sums S_1 are defined as

$$\begin{aligned} S_1(j) &= \sum_{n=j}^{N} a(n) = \sum_{n=j}^{N} S_0(n) = a(j) + \sum_{n=j+1}^{N} a(n) \\ &= S_0(j) + S_1(j+1),\ 1 \leq j \leq N. \end{aligned} \tag{14}$$

Similarly, S_i, $2 \leq i < p + 1$, can be defined and recursively computed as

$$\begin{aligned} S_i(j) &= \sum_{n_1=j}^{N} \sum_{n_2=n_1}^{N} \cdots \sum_{n_i=n_{i-1}}^{N} a(n_i) = \sum_{n_1=j}^{N} S_{i-1}(n_1) \\ &= S_{i-1}(j) + \sum_{n_1=j+1}^{N} S_{i-1}(n_1) \\ &= S_{i-1}(j) + S_i(j+1),\ 1 \leq j \leq N, \end{aligned} \tag{15}$$

where $S_i(N + 1) = 0$. According to the above definitions, note that S_i can be obtained by recursively computing the suffix sums over the suffix sums S_{i-1}.

Next, we will use the suffix sums functions to compute moment m_p, $0 \leq p \leq 3$. Let $b_p(x)$, $1 \leq x \leq N$ represent the partial moment of order p and $b_p(1) = m_p$. In the following, we will show that $b_p(x)$ is a recurrence function, and it can be expressed as a linear combination of S_i as defined previously.

Lemma 3 *Let $b_0(x) = \sum_{i=x}^{N} a(i)$, then $m_0 = S_1(1)$ and $b_0(x) = S_1(x)$, $1 \leq x \leq N$, where m_0 is the zero-order moment.*

Proof: Let $b_0(1) = \sum_{i=1}^{N} a(i)$, $b_0(2) = \sum_{i=2}^{N} a(i)$, $\cdots$, $b_0(N) = a_N$. Then $b_0(x) = S_1(x)$ by Eq. (14) and $m_0 = b_0(1) = S_1(1)$. □

Lemma 4 *Let $b_1(x) = \sum_{i=x}^{N} (i-x+1)a(i)$, then $m_1 = S_2(1)$ and $b_1(x) = S_2(x)$, $1 \leq x \leq N$, where m_1 is the one-order moment.*

Proof: $b_1(1)$ can be rewritten as

$$\begin{aligned} b_1(1) &= \sum_{i=1}^{N} ia(i) = \sum_{i=2}^{N} (i-1)a(i) + \sum_{i=1}^{N} a(i) \\ &= \sum_{i=3}^{N} (i-2)a(i) + \sum_{i=2}^{N} a(i) + \sum_{i=1}^{N} a(i) \\ &= \cdots \\ &= \sum_{i=1}^{N} a(i) + \sum_{i=2}^{N} a(i) + \cdots + \sum_{i=N}^{N} a(i) \\ &= \sum_{i=1}^{N} \sum_{j=i}^{N} a(j) = \sum_{i=1}^{N} S_1(i) = S_2(1). \end{aligned} \tag{16}$$

From Eq. (16), observe that $\sum_{i=2}^{N} (i-1)a(i) = \sum_{i=1}^{N} S_1(i) - \sum_{i=1}^{N} a(i) = \sum_{i=2}^{N} S_1(i) = S_2(2)$. Consequently, $\sum_{i=x}^{N} (i-x+1)a(i) = S_2(x)$, $1 \leq x \leq N$. These imply that $b_1(x) = S_2(x)$. Then, we conclude that $m_1 = \sum_{i=1}^{N} ia(i) = b_1(1) = S_2(1)$. □

Lemma 5 *Let $b_2(x) = \sum_{i=x}^{N} (i-x+1)^2 a(i)$, then $m_2 = S_3(1) + S_3(2)$ and $b_2(x) = S_3(x) + S_3(x+1)$, $1 \leq x \leq N$, where m_2 is the two-order moment.*

Proof: Based on Lemma 4, $b_2(1)$ can be rewritten as

$$\begin{aligned} b_2(1) &= \sum_{i=1}^{N} i^2 a(i) = \sum_{i=2}^{N} (i-1)^2 a(i) + \sum_{i=2}^{N} (i-1)a(i) \\ &\quad + \sum_{i=1}^{N} ia(i) \end{aligned}$$

$$
\begin{aligned}
&= \cdots \\
&= \sum_{i=1}^{N} ia(i) + 2\sum_{i=2}^{N}(i-1)a(i) + 2\sum_{i=3}^{N}(i-2)a(i) \\
&\quad + \cdots + 2\sum_{i=N}^{N}(i-N+1)a(i) \\
&= S_2(1) + 2S_2(2) + 2S_2(3) + \cdots + 2S_2(N) \\
&= \sum_{i=1}^{N} S_2(i) + \sum_{i=2}^{N} S_2(i) \\
&= S_3(1) + S_3(2). \qquad (17)
\end{aligned}
$$

From Eq. (17), observe that $\sum_{i=2}^{N}(i-1)^2a(i) = \sum_{i=1}^{N} S_2(i) + \sum_{i=2}^{N} S_2(i) - \sum_{i=2}^{N}(i-1)a(i) - \sum_{i=1}^{N} ia(i) = S_3(1) + S_3(2) - S_2(2) - S_2(1) = S_3(2) + S_3(3)$. Consequently, $\sum_{i=x}^{N}(i-x+1)^2a(i) = S_3(x) + S_3(x+1)$, $1 \le x \le N$. These imply that $b_2(x) = S_3(x) + S_3(x+1)$. Then, we conclude that $m_2 = \sum_{i=1}^{N} i^2a(i) = b_2(1) = S_3(1) + S_3(2)$. □

Lemma 6 *Let* $b_3(x) = \sum_{i=x}^{N}(i-x+1)^3a(i)$, *then* $m_3 = S_4(1) + 4S_4(2) + S_4(3)$ *and* $b_3(x) = S_4(x) + 4S_4(x+1) + S_4(x+2)$, $1 \le x \le N$, *where* m_3 *is the three-order moment.*

Proof: The proof is similar to Lemma 5. □

From the above arguments, it is easy to see that $b_p(x)$ is a recurrence function with $b_p(1) = m_p$ and $m_p(N) = m_{p-1}(N) = a(N)$. So solving the recurrence function, we can find the relation between b_p and S_i. Similarly, the higher order moments ($p > 3$) can be computed using the suffix sum functions. According to the above derivations, we conclude that the p-order moment m_p can be expressed as a linear combination of the suffix sums S_i, $0 \le i \le p+1$. However, the noise of images of higher order moments are more sensitive than that of lower order moments. In this paper, we concern about the lower order moments up to order 3 since these lower order moments can provide sufficient information for pattern recognition [2, 6].

As a further refinement of the 2-D moment computation, the double summation in Eq. (11) can be split into two separate summations:

$$m_{pq} = \sum_{x=1}^{N} x^p \sum_{y=1}^{N} y^q a(x,y) = \sum_{x=1}^{N} x^p m_{q,x}, \qquad (18)$$

where $m_{q,x}$ is the q-order row moment of row x, and it is defined as:

$$m_{q,x} = \sum_{y=1}^{N} y^q a(x,y). \qquad (19)$$

From the definition of the 2-D moments (Eq. (18)), the algorithm for computing 2-D moments m_{pq} can be decomposed into two phases. First compute the N 1-D row moments $m_{q,x}$, $1 \le x \le N$, for each row by applying Eq. (19) in parallel. Then the row moments $m_{q,x}$ are used to compute the 2-D moments m_{pq}. Instead of computing Eq. (19) directly, the 1-D moments can be computed by Lemmas 3-6. Thus, $m_{q,x}$, $0 \le q \le 3$, $1 \le x \le N$, can be computed by using the suffix sum functions as shown in Lemmas 3-6; m_{pq}, $0 \le p, q \le 3$, can be also computed by using the suffix sums functions by setting $S_0(j)$ to $m_{q,j}$, $0 \le j < N$, initially in Eq. (13).

Initially, assume that the gray-level image A is stored in the local variable $a(i,\ j)$ of processor $P_{i,\ j}$, $1 \le i,\ j \le N$. Finally, the results are stored in the local variable $m_{pq}(1,1)$ of processor $P_{1,1}$. Following the definitions of moments, the suffix sums, and the relationships between them, the detailed moment algorithm (MOMENT) is listed as follows.

Algorithm MOMENT;

1: For each row x, $1 \le x \le N$, apply Corollary 2 to compute the high order suffix sum functions S_i, $1 \le i \le 4$, by initializing $S_0(x,y) = a(x,y)$, $1 \le y \le N$. After that, the results S_i, $1 \le i \le 4$, are stored in local variable $S_i(x,y)$ in processor $P_{x,\ y}$.

2: For each row x, $1 \le x \le N$, compute the 1-D row moments $m_{q,x}$, $0 \le q \le 3$, according to Lemmas 3-6. After that, the 1-D row moments $m_{q,x}$, $0 \le q \le 3$, $1 \le x \le N$, are stored in the local variable $m_q(x,1)$ of processor $P_{x,\ 1}$.

3: Route $m_q(x,1)$, located on the first column processor $P_{x,\ 1}$, $1 \le x \le N$, to the $(q+1)$th row processor $P_{q+1,\ x}$.

4: For each row x, $1 \le x \le N$, apply Corollary 2 to compute the high order suffix sums functions S'_i, $1 \le i \le 4$, by initializing $S'_0(x+1,y) = m_q(x+1,y)$, $0 \le x \le 3$, $1 \le y \le N$. After that, the results S'_i, $1 \le i \le 4$, are stored in the local variable $S'_i(x+1,y)$ in processor $P_{x+1,\ y}$.

5: Compute the 2-D moments m_{pq}, $0 \le p, q \le 3$, according to Lemmas 3-6. After that, the 2-D moments m_{pq}, $0 \le p, q \le 3$, are stored in the local variable $m_{pq}(q+1,1)$ of processor $P_{q+1,\ 1}$.

Theorem 1 *Given an* $N \times N$ *gray level image* A, *the 2-D moments up to order 3 can be computed in* $O(1)$ *time on an* $N \times N^{1+\frac{1}{c}}$ *AROB for a constant* c *and* $c \ge 1$.

Proof : The correctness of this algorithm directly follows from Eqs. (11)-(19), Lemmas 3-6 and Corollary 2. Steps 1 and 4 implement Eqs. (13)-(15). Step 2 implements Eq. (19). Step 5 implements Eq. (18). The time complexity is analyzed as follows. Setting $k = 4$ in Corollary 2, Step 1 takes $O(1)$ time using $N \times N^{1+\frac{1}{c}}$ processors. Similarly, Setting $k = 8$ in Corollary 2, Step 4 also takes $O(1)$ time. Steps 2, 3 and 5 each take $O(1)$ time. Hence, the time complexity is $O(1)$. □

5 Concluding Remarks

In this paper, we first derive the relationships between the suffix sums and the moments. Then using these relationships, we propose an optimal algorithm for 2-D moment computation on a 2-D AROB. When the domain of the image is $O(\log N)$-bit integer, for a constant c, $c \geq 1$, the proposed algorithm can be run in $O(1)$ time using $N \times N^{1+\frac{1}{c}}$ processors. This result is better than any previously results derived in the literature [2, 3, 13, 14].

References

[1] Y. Ben-Asher, D. Peleg, R. Ramaswami, and A. Schuster, "The power of reconfiguration", *Journal of Parallel and Distributed Computing*, **13**, 1991, pp. 139–153.

[2] K. Chen, "Efficient parallel algorithms for computation of two-dimensional image moments", *Pattern Recognition*, **23**, 1990, pp. 109-119.

[3] H. D. Cheng, C. Y. Wu and D. L. Hung, "VLSI for moment computation and its application to breast cancer detection", *Pattern Recognition*, **31**, 1998, pp. 1391-1406.

[4] Z. Guo, R. G. Melhem, R. W. Hall, D. M. Chiarulli, and S. P. Levitan, "Pipelined communications in optically interconnected arrays", *Journal of Parallel and Distributed Computing*, **12**(3), 1991, pp. 269–282.

[5] M. Hatamian, "A real time two-dimensional moment generation algorithm and its single chip implementation", *IEEE Trans. ASPP*, **34**(3), 1986, pp. 546-553.

[6] M. K. Hu, "Visual pattern recognition by moment invariants", *IRE Trans. Inform. Theory*, **IT-8**, 1962, pp. 179-187.

[7] B. C. Li, "A new computation of geometric moments", *Pattern Recognition*, **26**, 1993, pp. 109-113.

[8] K. Li, Y. Pan and S. Q. Zheng, "Fast and processor efficient parallel matrix multiplication algorithms on a linear array with a reconfigurable pipelined bus system", *IEEE Trans. on Parallel and Distributed systems*, **9**, 1998, pp. 705-720.

[9] Y. Pan and K. Li, "Linear array with a reconfigurable pipelined bus system— concepts and applications", *Information Sciences – An International Journal*, **106**(3/4), 1998, pp. 237-258.

[10] S. Pavel and S. G. Akl, "On the power of arrays with reconfigurable optical bus", *Int. Conf. on Parallel and Distributed Processing Techniques and Applications*, 1996, pp. 1443-1454.

[11] S. Pavel and S. G. Akl, "Matrix operations using arrays with reconfigurable optical buses", *Parallel Algorithms and Applications*, **8**, 1996, pp. 223-242.

[12] W. Philips, "A new fast algorithm for moment computation", *Pattern Recognition*, **26**, 1993, pp. 1619-1621.

[13] A. P. Reeves, "A parallel mesh moment computer", in *Proc. 6th ICPR*, 1982, pp. 465-467.

[14] A. P. Reeves, "Parallel algorithms for real-time image processing", *Multicomputers and Image Processing, Algorithms and Programs*, pp. 7-18. Academic Press, New York, 1982.

[15] T. W. Shen, D. P. K. Lun and W. C. Siu, "On the efficient computation of 2-D image moments using the discrete Radon transform", *Pattern Recognition*, **31**, 1998, pp. 115-120.

[16] M. H. Singer, "A general approach to moment calculation for polygons and line segments", *Pattern Recognition*, **26**, 1993, pp. 1019-1028.

[17] L. Yang and F. Albregtsen, "Fast and exact computation of Cartesian geometric moments using discrete Green's theorem", *Pattern Recognition*, **29**, 1996, pp. 1061-1073.

[18] M. F. Zakaria, P. J. A. Zsombor-Murray and J. M. H. H. Kessel, "Fast algorithm for the computation of moment invariants", *Pattern Recognition*, **20**, 1987, pp. 639-643.