A PARALLEL ALGORITHM FOR COMPUTATION OF 2D IMAGE MOMENTS

WAFA KHADER ALDABABAT[1], MOHAMMAD HJOUJ BTOUSH[2], ADNAN I. ALRABEA[3]

Information Technology Department, Al-Balqa' Applied University
Al Salt-JORDAN

ABSTRACT

Moment functions of the two dimensional image intensity distributions are used as descriptors of shape in a variety of applications. In this work, an improvement on the moment computation time is considered by providing the design and implementation of a parallel algorithm that will speedup moment computation.

Keywords: *Third Order Moments, Parallel Algorithm, Digital Filter, Moment Invariants*.

I. INTRODUCTION

Image feature representation techniques using moment functions have been used in many applications, such as, invariant pattern recognition, object identification and image reconstruction. Low order moments are commonly used to find the location and orientation of an object. The zero through second order moments provide information about the area, center of gravity and about approximation of an object to an ellipse. High order moments are combined to form moment invariants to certain shape transformations, such as translation, rotation and scaling. Moment invariants have been used as features for pattern recognition.

Orthogonal moment functions can be defined as *continuous integrals* over a domain of normalized coordinates [5,3,9], but these moments involve a major source of error which is, the discrete approximation of the continuous integrals. *Discrete* orthogonal polynomials are used as basis functions for image moments, to solve the above problem [3,8]. Moment invariants were proposed by Hu [8], and have been used as a basis for many applications of image recognition. Regardless of the moment function used, the problem considered is that computation of moments requires a large computation time. Many researchers have proposed fast and efficient algorithms to reduce the computation time [2], [6], [7], [10], [11], [12], [14], [15], [16] and [17] .

In this paper, a parallel algorithm based on the improved filter algorithm [13] will be proposed and implemented on different parallel architectures using different partitioning methods.

The rest of this paper is organized as follows. Section 2 introduces 2D image moments, straightforward sequential and parallel algorithms for moment computation, and striped partitioning of images is discussed. Section 3 discusses third order moments and their fast computation using digital filter method [13]. In Section 4, we introduce our work by presenting a parallel algorithm for third order moments computation based on the digital filter method. Efficiency analysis for the parallel algorithm on different architectures is discussed in section 5. We end this paper by concluding some remarks in Section 6.

II. 2D IMAGE MOMENTS

A. Moment definition function

In this work, the moment definition function that will be considered is in equation *Eq.1*

$$M_{pq} = \sum_{x=1}^{N} \sum_{y=1}^{M} x^p y^q f(x,y) \qquad .. \quad (1)$$

Where:
p, q= 0,1,2,3,4,… (Positive integer numbers)
$f(x,y)$ is the two dimensional image intensity function.
$p+q$ = 0,1,2,3,4,… is the moment order, which is defined as the summation of p and q.
N is the vertical dimension of the image.
M is the horizontal dimension of the image.

A sequential algorithm for computation of the moments based on equation *Eq.1* is presented as given in algorithm.1.

Algorithm.1. A sequential algorithm for computation of moments of a 2D image

1. Input the values of p and q
2. $Moment_{p+q}=0$
3. For x=1 to x=N
4. For y=1 to y=M
5. Evaluate $temp=x^p*y^q*f(x,y)$
6. $Moment_{p+q}= Moment_{p+q}+temp$
7. Endfor y
8. Endfor x
9. Output $moment_{p+q}$

The above sequential algorithm requires N*M (additions, multiplications, (p-1) multiplications, (q-1) multiplications), this is a very large number of time-consuming doing multiplication and addition operations.

B. A Parallel Version of the Moment computation Function

Equation 2 can be derived from equation 1 as given in eq.2 without affecting the result of computation. Thus; x^p will be evaluated only N times instead of N*M times.

$$M_{pq} = \sum_{x=1}^{N} x^p \sum_{y=1}^{M} y^q f(x,y) \quad .. \quad (2)$$

Figure *Fig.1* shows the parallel approach of the moment function.

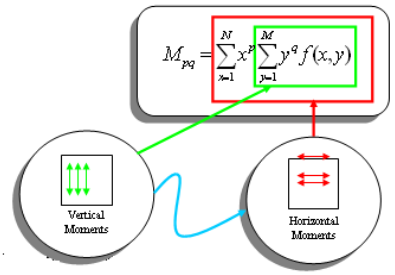

Fig.1. Moments Of a 2D Image

From Fig.1, the following were noticed about the parallel implementation of the moments: Vertical moments are chosen to be computed in parallel so that the term y^q will be evaluated only once by each processor since each processor will be responsible for one column (one value of y).

- Vertical moments of the image, as shown in the inner box, can be evaluated in parallel because they do not depend on each other; hence each one of them represents a value (or values) computed from one column of the matrix that implements the 2D image.
- Horizontal moments can then be evaluated for each row, shown in outer box, depending on the values of the vertical moments of that row. The final value of the moment can be evaluated by accumulating the horizontal moments.
- Computation of vertical and horizontal moments must be serialized, which means that a horizontal moment can't be evaluated until the vertical moments of that row have been computed.

So, depending on the previous analysis, Algorithm.2 is a parallel algorithm for computation of 2D image moments.

Algorithm.2. A parallel algorithm for computation of moments of a 2D image

1. Input the values of p and q
2. $Moment_{p+q}=0$
3. Evaluate in parallel the vertical moments of the image.
   $V_{x,y}=Y^q(F(x,y)) ,\forall y\in[1,M], x\in[1,N].$
4. Compute in parallel the horizontal moments of the image, depending on the values of the vertical moments that have been evaluated in 3.
   $H_x=X^p*(V_{x,y}) ,\forall y\in[1,M], x\in[1,N].$
5. $Moment_{p+q}= \sum H_x ,\forall\ x\in[1,N].$
6. Output $moment_{p+q}$

C. Striped Partitioning

Three approaches for data decomposition can be implemented: columnwise, rowwise and checkerboard partitioning. In our algorithm, the most suitable partitioning method is the column wise striped portioning.

Using columnwise striped portioning, the image is partitioned into column wise stripes, and different processors are assigned different stripes of the image. Assuming the number of processors equals the number of columns in the image (N=P), moments should be computed as shown in the following steps:

1. Each processor is assigned one stripe (column), assuming processor (P_1) is assigned column Y_1, processor (P_2) is assigned column Y_2… and processor P_N is assigned column Y_N.

2. All processors in parallel will compute the vertical moments of the column they own.

3. (A transposition of the vertical moments stored in each processor is made), so that each

processor will be assigned suitable vertical moment values, this can easily be achieved by all to all personalized communication between the processors.

4. All processors now concurrently estimate the horizontal moments.

5. The result then can easily be achieved by a single node accumulation of the horizontal moments computed in step 4.

Figure Fig.2 shows the entire steps of the proposed columnwise striped partitioning

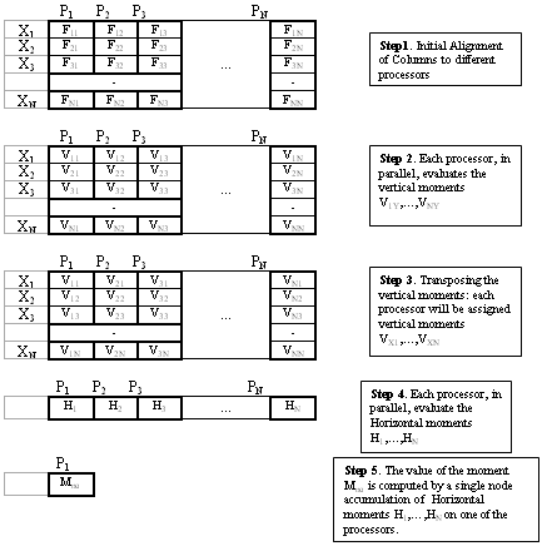

Fig.2. Striped Partitioning / Column Wise.

Scaling down the number of processors used can be made (number of processors less than number of columns in the image P<N). In this case, the stripe size will be N/P.

In step 1, each processor will be assigned N/P columns. Then, in step 2, each processor will evaluate the vertical moments of N/P columns. after that in step 3, the message size will be N/P.

In step 4, each processor will be responsible for computing the horizontal moments of N/P rows. Finally, in step 5, the message size during accumulation will be N/P.

III.THIRD ORDER MOMENTS OF A 2 DIMENSIONAL IMAGE

The third order moments of a 2D image are the moments with p+q=3. These include the moments m_{00}, m_{01}, m_{10}, m_{11}, m_{02}, m_{03}, m_{12}, m_{20}, m_{21} and m_{30}.

A. Improved digital filter method

One of the best known algorithms for computing third order moments of a 2D image is the *improved digital filter* [13]. Algorithm.3 presents this sequential algorithm, where the image size is NxM, and f is a two dimensional image intensity function.

Algorithm.3. A sequential algorithm for computation of 3rd order moments of a 2D image

$y_{00}=0$, $y_{01}=0$, $y_{02}=0$, $y_{03}=0$, $y_{10}=0$, $y_{11}=0$, $y_{12}=0$, $y_{20}=0$, $y_{21}=0$, $y_{30}=0$
$m_{00}=0$, $m_{01}=0$, $m_{10}=0$, $m_{11}=0$, $m_{02}=0$, $m_{03}=0$, $m_{12}=0$, $m_{20}=0$, $m_{21}=0$, $m_{30}=0$

```
For  j= M to 1 do  {
  y0=0; y1=0; y2=0; y3=0;
   For i = N to 1 do {
     y0=y0+f(i,j)
     y1=y1+y0
     y2=y2+y1
     y3=y3+y2          }
  y00=y00+y0
  y01=y01+y00
  y02=y02+y01
  y03=y03+y02
  y10=y10+y1
  y11=y11+y10
  y12=y12+y11
  y20=y20+y2
  y21=y21+y20
  y30=y30+y3     }
m00=y00
m01=y01
m10=y10
m11=y11
m02=y02+y02-y01
m03=6*(y03-y02)+y01
m12=y12+y12-y11
m20=y20+y20-y10
m21=y21+y21-y11
m30=6*(y30-y20)+y10
```

For an input pattern with NxM data, it takes (4N+10)*M additions to pass through the loops. After the completion of the outer loop, only 12 additions and two multiplications are needed to generate all the third order moments [13]. Moreover, since the multiplication operation is more time-consuming than the addition operation, the multiplications have been completely eliminated by replacing them with additions. So, the overall computation complexity will be ((4N + 10)*M+24) additions. Thus, the running time complexity $T_S(N,M)= O(NM)$.

B. Parallel version of the improved filter algorithm

The task graph of the improved filter algorithm for computation of third order moments, with input size (M=4, N=4) is shown in Fig.3.

Regarding the algorithm for moments computation presented in Alg.3., we have developed a parallel algorithm based on it. This can be done by following the steps presented in Alg.2. The image is decomposed into column stripes, these stripes are distributed on the processors. Then, all processors in parallel will compute the values(y0, y1, y2 and y3). The values of y0, y1, y2 and y3 from the previous step will be gathered in four different processors. Each of these processors will compute part of the horizontal moment values (y00, y01, y02, y03, y10, y11, y12, y20, y21 and y30). The horizontal moments are gathered into one processor for final evaluation of the moments (m00, m01, m10, m11, m02, m03, m12, m20, m21, and m30). This algorithm is presented in Algorithm.4. Assuming (P=M).

Algorithm.4.a. is a main program of the parallel algorithm for computing the third order moments based on improved filter algorithm. This program resides on main processor.

1. Decompose the image into M stripes (columns), each stripe of size N rows.
2. Distribute the M columns on M processors.
3. Receive the horizontal moment values y_{00}, y_{01}, y_{02}, y_{03} from processor P_1. y_{10}, y_{11}, y_{12} from processor P_2. y_{20}, y_{21} from processors P_3, and y_{30} from processor P_4.
4. Compute the values of the third order moments :

- m00=y00
- m01=y01
- m10=y10
- m11=y11
- m02=y02+y02-y01
- m03=6*(y03-y02)+y01
- m12=y12+y12-y11
- m20=y20+y20-y10
- m21=y21+y21-y11
- m30=6*(y30-y20)+y10

Algorithm.4.b. Slave program of the parallel algorithm for computing the third order moments based on improved filter algorithm.

1. Receive the image stripe of size N from the main processor.
2. All M processors compute in parallel the vertical moments y_0, y_1, y_2 and y_3.
   For i = N to 1 do {
   $y_0=y_0+f(i,j)$, *where j is the index of the column assigned to the processor*
   $y_1=y_1+y_0$
   $y_2=y_2+y_1$
   $y_3=y_3+y_2$ }
3. The M processors initiate four single node accumulations for the four vertical moments on four different processors.
   3.1. P_1 will gather the values of y_0 from the M processors in the array Y_0. $Y_0=\{ y_{0\_1}, y_{0\_2}, y_{0\_3},\ldots, y_{0\_M}\}$.
   3.2. P_2 will gather the values of y_1 from the M processors in the array Y_1. $Y_1=\{ y_{1\_1}, y_{1\_2}, y_{1\_3},\ldots, y_{1\_M}\}$.
   3.3 P_3 will gather the values of y_2 from the M processors in the array Y_2. $Y_2=\{ y_{2\_1}, y_{2\_2}, y_{2\_3},\ldots, y_{2\_M}\}$.
   3.4. P_4 will gather the values of y_3 from the M processors in the array Y_3. $Y_3=\{ y_{3\_1}, y_{3\_2}, y_{3\_3},\ldots, y_{3\_M}\}$.
4. Processors P_1, P_2, P_3 and P_4 compute in parallel the horizontal moments depending on the M values of the vertical moments gathered in step 3 in array Y.
   4.1. P_1 computes y_{00}, y_{01}, y_{02}, y_{03}.
   $y_{00}=0$, $y_{01}=0$, $y_{02}=0$, $y_{03}=0$
   For j= M to 1 do {
   $y_{00}=y_{00}+Y_0(j)$
   $y_{01}=y_{01}+y_{00}$
   $y_{02}=y_{02}+y_{01}$
   $y_{03}=y_{03}+y_{02}$ }
   4.2. P_2 calculates y_{10}, y_{11}, y_{12}.
   $y_{10}=0$, $y_{11}=0$, $y_{12}=0$
   For j= M to 1 do {
   $y_{10}=y_{10}+Y_1(j)$
   $y_{11}=y_{11}+y_{10}$
   $y_{12}=y_{12}+y_{11}$ }
   4.3. P_3 calculates y_{20}, y_{21}.
   $y_{20}=0$, $y_{21}=0$
   For j= M to 1 do {
   $y_{20}=y_{20}+Y_2(j)$
   $y_{21}=y_{21}+y_{20}$ }
   4.4. P_4 computes y_{30}.
   $y_{30}=0$
   For j= M to 1 do {
   $y_{30}=y_{30}+Y_3(j)$ }
5. Processors P_0, P_1, P_2 and P_3 sends the horizontal moments values computed in step 4 to the main program.

Accordingly; more speedup may be gained by using a pipeline for the computation of the vertical moments in each processor.

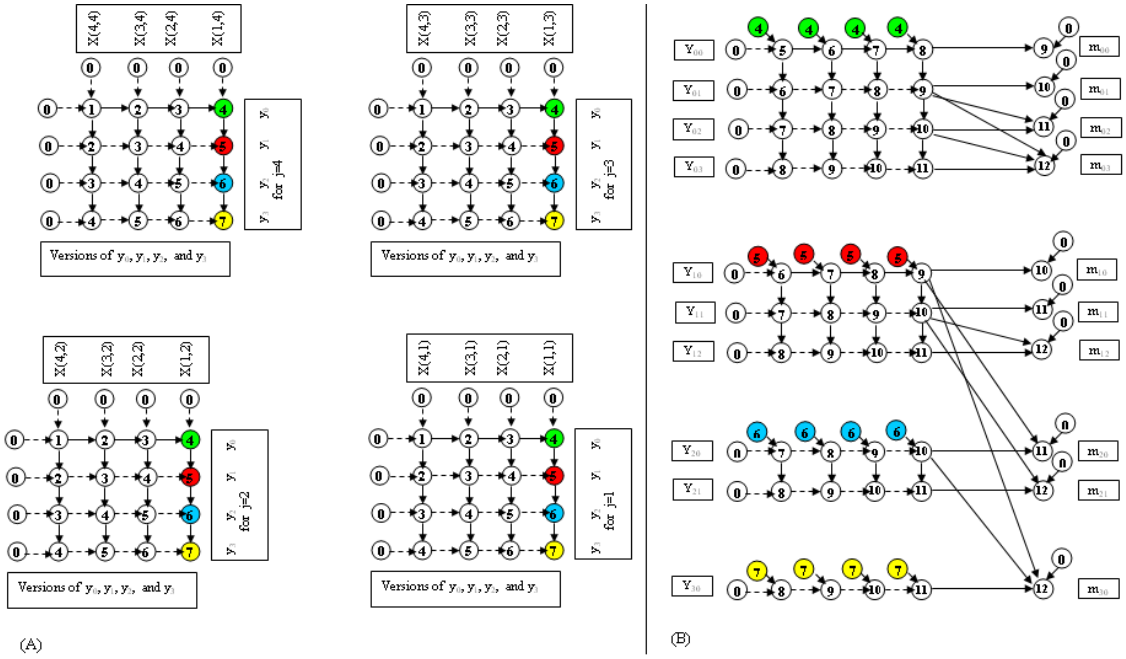

Fig.3. Task Graph of the Improved Filter Algorithm for image size 4X4 (A) Computing Y values B) computing the third order moments

IV. PERFORMANCE ANALYSIS OF THE PRESENTED PARALLEL ALGORITHM ON DIFFERENT ARCHITECTURES

The parallel algorithm presented in Algorithm.4. can be implemented on different parallel architectures [1], the architectures used in this work are the hypercube and the wraparound mesh interconnection networks, see fig.4 [1].

The following assumptions are used in the performance analysis:

- T_P : the parallel time of the algorithm
- E : the efficiency
- S : the speedup
- M : the number of columns in the image
- N : the number of rows in the image
- P : the number of processors
- T_w : the transmission time per element
- T_s : the start-up time

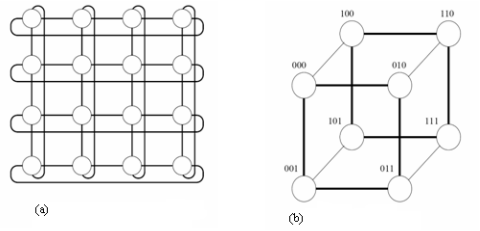

Fig.4. Interconnection networks. (a) 2D mesh with wraparound link (B) 3-D hypercube

A. Striped Partitioning.(P=M)

1) Hypercube Interconnection Network

Assuming that the image is already distributed on the processors, then (TP) will be composed of the following:

1. Evaluation of the vertical moments in all processors in parallel will take (4*(N-1)) additions.
2. Four single node gather for the values (y0, y1, y2 and y3) on four different processors, assuming SF routing is used, will take (Ts*Log2M + Tw*(M-1))*4.
3. The four processors in parallel will evaluate horizontal moments, this is limited

by the slowest processor (P1), which requires ((M-1)*4) additions.

4. Finally, single node gather of the 10 horizontal moments in one processor of the hypercube requires a maximum time of (Ts*Log24+Tw*(4-1))*4=((2Ts+3Tw)*4)= (8Ts+12Tw).
5. The single processor will now evaluate the moments sequentially; this requires 10 additions, and 2 multiplications.

Assuming that a basic communication operation time equals a single multiplication operation time, then T_P is the summation of all previous times

$T_P=((T_s*Log_2^M+T_w*(M-1))*4+(4*(N-1))+(M-1)*4+(8T_s+12T_w)+10+2)$,which is approximately $\Theta(N+M+ Log_2^M)$.

$S=N*M/(N+ M +Log_2^M)$.

$E= N*M/M(N+M+Log_2^M) = N/(N+M+Log_2^M)$, when $M\cong N$ $E=1/(2+ Log_2^M /N)$. for large values of M E is approximately $\Theta(1)$, meaning that the parallel system is cost optimal.

Assuming that the image is stored on the source processor, and number of processors equals number of columns (P=M), then additional time is needed for decomposing the image on the M processors, one to all personalized communication is needed with message size = N, the time required will be $(T_s*Log_2^M +T_w*N*(M-1))$. And in this case, the parallel time will be,

$T_P=((T_s*Log_2^M+T_w*(M-1))*4+(4*(N-1))+(M-1)*4+(8T_s+12T_w)+10+2+(T_s*Log_2^M +T_w*N*(M-1)))$.

$\Theta(N+M+ Log_2^M+N*M)$, which is apparently not cost optimal.

2) Wraparound Mesh Interconnection Network

Assuming that the image is already distributed on the processors, then (TP) will be composed of the following:

1. Evaluation of the vertical moments in all processors in parallel will take (4*(N-1)) additions.
2. Four single node gather for the values (y0, y1, y2 and y3) on four different processors, will take 4*(2Ts*(M1/2-1)+Tw*(M-1)).
3. The fours processors in parallel will evaluate horizontal moments, this is limited by the slowest processor (P1), which requires ((M-1)*4) additions.
4. Finally, single node gather of the 10 horizontal moments in the mesh requires a maximum time of four single node gather on a ring of 4 processors =((Ts+Tw)*(4-1) *4)= ((Ts+Tw)*12).
5. The single processor will now evaluate the moments sequentially; this requires 10 additions, and 2 multiplications.

When a basic communication operation time equals a single multiplication operation time, TP is the summation of all previous times.

$T_P=(4*(2T_s*(M^{1/2}-1)+T_w*(M-1)+4*(N-1)+(M-1)*4+(T_s+T_w)*12+10+2)$,which is approximately $\Theta(M^{1/2}+N+ M)$.

Then $S=N*M/(M^{1/2}+N+ M)$.

$E= N*M/M(M^{1/2}+N+ M) = N/(M^{1/2}+N+ M$ when $M\cong N$ $E=1/(2+ M^{1/2}/N)$. for large values of M E is approximately $\Theta(1)$, meaning that the parallel system is cost optimal.

Assuming that the image is stored on the source processor, and the number of processors equals the number of columns (P=M), then an extra time is needed for decomposing the image on the M processors, one to all personalized communication is needed with message size = N, the time needed will be $(2T_s*(M^{1/2}-1) +T_w*N*(M-1))$. And in this case, the parallel time will be,

$T_P=(4*(2T_s*(M^{1/2}-1)+T_w*(M-1)+4*(N-1)+(M-1)*4+(T_s+T_w)*12+10+2+T_w*N*(M-1)))$.

$\Theta(N+M+ M^{1/2}+N*M)$, which is apparently not cost optimal.

B. Striped Partitioning (P<M)

A scaling down of the number of processors used can be made (the number of processors is less than number of columns in the image P<M). And in this case the stripe size will be M/P. Computation of the moments will be the same as illustrated in Alg.4 , with the following differences: In step 1. Each processor will be assigned M/P columns instead of 1. In step 2. Each processor will evaluate the vertical moments of M/P columns. In step 3. The message size will be M/P instead of 1.

1) Hypercube Interconnection Network

Assuming that the image is already distributed on the processors; then T_P will be composed of the following:

1- Evaluation of the vertical moments in all processors in parallel will take 4*(M/P)*(N-1) additions.

2- Four single node gather for the values (y0, y1, y2 and y3) on four different processors, assuming SF routing is used, will take (Ts*Log2P + Tw*M/P*(P-1))*4.

3- The fours processors in parallel will evaluate horizontal moments, this is limited by the slowest processor (P!), which requires ((M-1)*4) additions.

4- Finally, single node gather of the 10 horizontal moments in the hypercube requires a maximum time of (Ts*Log24 + Tw*M/P*(4-1)*4)= (8Ts+12Tw*M/P).

5- The single processor will now evaluate the moments sequentially; this requires 10 additions, and 2 multiplications.

Assuming that a basic communication operation time equals a single multiplication operation time, then T_P equals the summation of all previous times.

T_P=((T_s*Log_2^P + T_w*M/P*(P-1))*4+4*(N-1)*(M/P)+(M-1)*4+8T_s+12T_w*M/P+10+2),
which is approximately Θ((N*M/P+ M +Log_2^P).

Then S=N*M/(N*M/P+ M +Log_2^P).

And E= N*M/P(N*M/P+ M +Log_2^P)= 1/(1+(P/N)+P*(Log_2^P)/(N*M)), for P<<M, it will be Θ(1), meaning that the parallel system is cost optimal.

Assuming that the image is stored on the source processor, and the number of processors is less than the number of columns (P<M), then an extra time is needed for decomposing the image on the P processors, one to all personalized communication is needed with message size = N*M/P, the time needed will be (T_s*Log_2^P +T_w* N*M/P *(P-1)). And in this case, the parallel time will be,

T_P= ((T_s*Log_2^P + T_w*M/P*(P-1)) *4+ (N-1)*(M/P) + (M-1)*4+((T_s+T_w)*8)+10+2+(T_s*Log_2^P +T_w* N*M/P *(P-1))), which is Θ(N*M/P+ M +Log_2^P+N*M).

S= N*M/(N*M/P+ M +Log_2^P+N*M).

E= N*M/P(N*M/P+ M +Log_2^P+N*M) = 1/(1+(P/N)+(P*(Log_2^P)/(N*M))+P), for P<<M, it will be Θ(1/(1+P)) < Θ(1), meaning that the parallel system is not cost optimal.

2) Wraparound Mesh Interconnection Network

Assuming that the image is already distributed on the processors, let T_P be the parallel time of the algorithm; then T_P will be composed of the following:

1. Evaluation of the vertical moments in all processors in parallel will 4*(M/P)*(N-1)additions.
2. Four single node gather for the values (y0, y1, y2 and y3) on four different processors, assuming CT routing is used, will take 4*(2Ts*(P1/2-1)+Tw*M/P*(P-1)).
3. The fours processors in parallel will evaluate horizontal moments, this is limited by the slowest processor (P1), which requires ((M-1)*4) additions.
4. Single node gather of the 10 horizontal moments in the mesh requires a maximum time of a four single node gather on a ring of 4 processors =((Ts+Tw)*(4-1) *4)= ((Ts+Tw)*12).
5. The single processor will now evaluate the moments sequentially; this requires 10 additions, and 2 multiplications.

T_P=(4*(2T_s*($P^{1/2}$-1)+T_w*M/P*(P-1))+4*(N-1)*(M/P)+(M-1)*4+((T_s+T_w)*12)+10+2), which is approximately Θ((N*M/P+ M + $P^{1/2}$).

S=N*M/(N*M/P+ M + $P^{1/2}$).

E= N*M/P(N*M/P+ M + $P^{1/2}$)= 1/(1+(P/N)+P*($P^{1/2}$)/(N*M)), for P<<M, it will be Θ(1), meaning that the parallel system is cost optimal.

Assuming that the image is stored on the source processor, and number of processors equals number of columns (P<M), then an extra time is needed for decomposing the image on the P processors, one to all personalized communication is needed with message size = N*M/P, the time needed will be (2T_s*($P^{1/2}$-1) +T_w*N*M/P*(P-1)). And in this case, the parallel time will be T_P=(4*(2T_s*($P^{1/2}$-1)+T_w*M/P*(P-1))+(N-1)*(M/P)+(M-1)*4+((T_s+T_w)*8)+10)+(2T_s*($P^{1/2}$-1)+T_w*N* M/P*(P-1))).

Θ(N+M+ Log_2^P+ $P^{1/2}$+N*M).

E=(N*M)/ P(N+M+$P^{1/2}$+N*M), for P<<M, it will be Θ(1/(1+P)) < Θ(1), which is apparently not cost optimal.

V. CONCLUSION

This work has proposed a parallel algorithm for computation of third order moments of a 2D image by showing the architectures on which this algorithm can be parallelized with optimal costs in comparison with these on which the algorithm is costly.

The parallel algorithm analysis presented in this paper can benefit programmers and algorithm developers by presenting a systematic way to tackle a sequential problem, implementing and analysing it using a parallel algorithm on different architectures.

REFERENCES

[1] Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar. 2003. Introduction to Parallel Computing. Addison-Wesley.

[2] M. Hatamian, A real time two-dimensional moment generation algorithm and its single chip implementation, IEEE Transactions on ASSP 34 (1986), pp. 546–553.

[3] Patrick C Hew, Michael D Alder, Zernike Or Orthogonal Fourier Mellon Moments For Representing And Recognising Printed Digits, 1996.

[4] M.K. Hu, Visual pattern recognition by moment invariants, IRE Transactions on Information Theory 8 (1962), pp. 179–187.

[5] Drew E. Kornreich, Anthony B. Davis, and Charles A. Rohde, "Analysis of the I3RC Two-Dimensional Step Cloud Problem with TWODANT",1995.

[6] B.C. Li, High-order moment computation of grey-level images, IEEE Transactions on Image Processing 4 (1995), pp. 502–505.

[7] J. Martinez and F. Thomas, Efficient computation of local geometric moments, IEEE Transactions on Image Processing 11 (2002), pp. 1102–1112.

[8] Mukundan R, S.H.Ong, P.A.Lee, Discrete Orthogonal Moment Features using Chebyshev Polynomials, Image and Vision Computing, NewZealand NOV 2000, pp. 20-25.

[9] R. Mukundan, S.H.Ong, P.A. Lee, Discrete vs. Continuous Orthogonal Moments in Image Analysis",Intnl. Conf. on Imaging Science, Systems and Technology-CISST'01, Las Vegas, (25-28 June 2001) pp. 23-29.

[10] W. Philips, A new fast algorithm for moment computation, Pattern Recognition 26 (1993), pp. 1619–1621.

[11] A.P. Reeves, A parallel mesh moment computer, in: Proc. 6th ICPR, 1982, pp. 465–467.

[12] 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.

[13] W.H.Wong, W.C.Siu, Improved Digital Filter Structure For Fast Moments Computation, Image Signal Process, Vol. 146, No. 2, April 1999.

[14] C.H. Wu, S.J. Horng and P.Z. Lee, A new computation of shape moments via quadtree decomposition, Pattern Recognition 34 (2001), pp. 1319–1330

[15] C.H. Wu and S.J. Horng, Run-length chain coding and scalable computation of a shape's moments using reconfigurable optical buses, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 34 (2004), pp. 845–855

[16] C.H. Wu , S.J. Horng, C.F. Wen and Y.R Wang, Fast and scalable computations of 2D image moments, Image and Vision Computing, Volume 26, Issue 6, 2 June 2008, pp. 799-811

[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.