COMPUTER VISION AND IMAGE UNDERSTANDING
Vol. 61, No. 2, March, pp. 278–284, 1995

NOTE

Hough Transform on Reconfigurable Meshes*

KUO-LIANG CHUNG† AND HORN-YI LIN

Department of Information Management, National Taiwan Institute of Technology, No. 43, Section 4, Keelung Road, Taipei, Taiwan 10672, Republic of China

Received August 23, 1993; accepted July 25, 1994

The Hough transform is an important image processing operation. Given an $N \times N$ digital image, we first present a parallel algorithm for computing the Hough transform in $O((p/k) \log N)$ time on a reconfigurable mesh of $O(kN^2)$ processors, $1 \le k \le p$, where p is the number of angles to be considered. Then, on a 3-dimensional reconfigurable mesh using $O(kN^3)$ processors, the Hough transform can be computed in $O(p/k)$ time. When setting $k = O(p)$, a constant time algorithm is derived. Furthermore, a more general result is presented; the Hough transform can be computed in $O((p \log N)/(k \log M))$ time on a reconfigurable mesh of $O(kN^2M)$ processors, where $2 \le M \le N$. © 1995 Academic Press, Inc.

1. INTRODUCTION

The well-known Hough transform (HT) is an efficient method for detecting predefined features in digital images [3, 24]. Without loss of generality, we only consider the binary image, i.e., the pixel value is 0 or 1, and consider the HT for straight line detection using normal parameters as suggested by Duda and Hart [6]. The HT was brought to the attention of the mainstream image community by Rosenfeld [23]. In order to detect lines through large collinear subsets of a planar set of point $I(i, j)$ for $0 \le i, j \le N - 1$, each point is transformed into the sinusoidal curve in the (r, θ) plane which is defined by

$$r = i \cos \theta + j \sin \theta. \quad (1)$$

In Fig. 1, all points $I(i, j)$ on L satisfy (1). The HT utilizes (1) to detect straight lines or edges in an image. Given a projection angle θ ($-\pi/2 < \theta \le \pi/2$), let 1-pixel be the pixel with value 1; we first count the number of 1-pixels with value r. Setting a threshold value δ that depends on the practical demand, if the number of 1-pixels with value r is greater than or equal to δ, then the corresponding 1-pixels form a line. We try out a set $\{\theta_j | 0 \le j < p\}$ of p possible θ's. This is equivalent to trying out a set of p possible slopes for the lines to be detected.

Parallel computing for image processing and pattern recognition [15, 16, 31, 33, 37] has received considerable attention during the last few years. The progress of VLSI technology has made new parallel organizations possible. From the architecture viewpoint, mesh-connected computers (MCCs) are of particular interest to the image processing community. Some real machines such as the CLIP, GAPP, and the MPP [1, 5, 10] have been built for realizing the MCCs. Many parallel HT algorithms have been presented on MCCs [4, 8, 9, 25].

The main drawback of MCCs is the large communication diameter and MCCs tend to be slow when it comes to handling data transfer operations over long distances. In order to overcome this large diameter problem, MCCs have recently been enhanced by the addition of various bus systems. Some researchers [14, 16, 29–22, 28, 34–36] consider MCCs enhanced by a number of bus systems whose configuration can be changed dynamically. Such a bus system is referred to as a *reconfigurable* bus system. An MCC with a reconfigurable bus system is abbreviated as a reconfigurable mesh (RM). The parallel algorithm proposed by Kao *et al.* [11] computes the HT in $O(1)$ time on an RM using $O(pE^3)$ processors, where p is the number of angles to be considered and E denotes the number of edge pixels in the image. However, due to its input constraint being dominated by the number of edge pixels, their RM cannot be reused for different images because commonly the number of edge pixels for different images is hardly equivalent. A fixed large number of processors used is another limitation of Kao *et al.*'s result. Hence, the method proposed by Kao *et al.* [11] is still impractical to be implemented. Therefore, how to design new parallel HT algorithms on RMs but without the above two restrictions in Kao *et al.*'s result is the main motivation of this research. Another motivation is to reply to Olariu *et al.*'s conjecture [21]: whether the HT can be computed in $O(\log \log N)$ Time on RMs.

* This research is supported in part by the National Science Council of the Republic of China under Contract NSC82-0415-E011-180.

† To whom correspondence should be addressed.

1077-3142/95 $6.00
Copyright © 1995 by Academic Press, Inc.
All rights of reproduction in any form reserved.

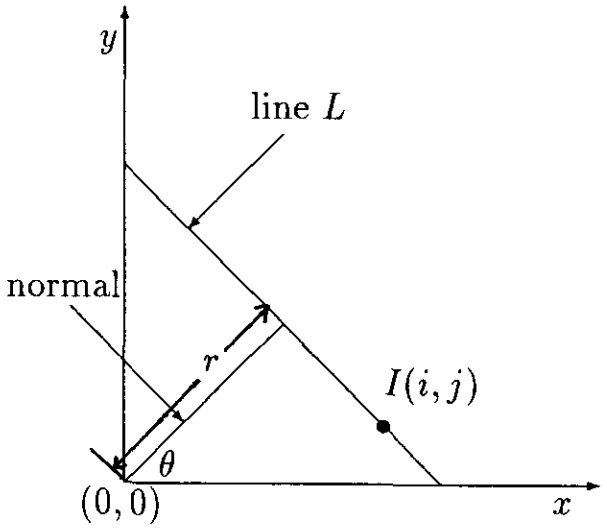


FIG. 1. (r, θ) plane.

This paper first presents a parallel algorithm for computing the HT in $O(p/k)$ log N) time on a 2-dimensional (2-D) RM of $O(kN^2)$ processors, $1 \le k \le p$. Then, on a 3-dimensional (3-D) RM with $O(kN^3)$ processors, the HT can be computed in $O(p/k)$ time. When setting $k = O(p)$, a constant time, i.e., $O(1)$ time, algorithm is derived. Furthermore, a more general result is presented; the HT can be computed in $O((p \log N)/(k \log M))$ time on an RM of $O(kN^2M)$ processors. When setting $k = O(p)$ and $M = O(\log N/\log \log N)$, we reply to Olariu *et al.*'s conjecture [21] that on an RM, the $O(\log \log N)$ algorithm for computing the HT is achievable. In addition, when setting $k = O(p)$ and $M = O(N^{1/c})$, where c is a constant, the HT can be computed in $O(1)$ time.

2. COMPUTATIONAL MODEL

The computational model used throughout this paper is the RM. An $N_1 \times N_2$ RM consists of $N_1 \times N_2$ identical processors positioned on a rectangular array with N_1 rows and N_2 columns. For example, an RM of size 4×5 shown in Fig. 2 contains four rows and five columns.

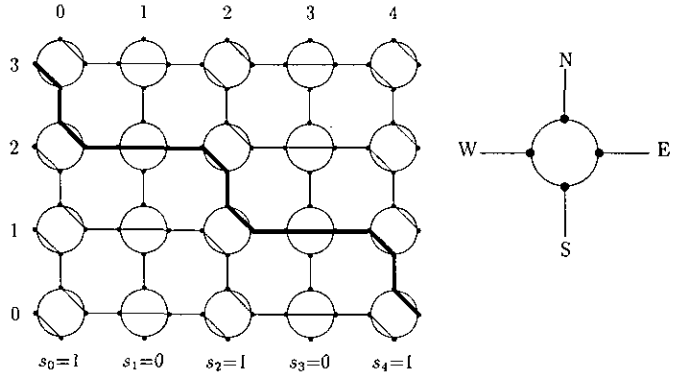


FIG. 2. An RM of size 4×5.

The processor located in row i and column j for $0 \le i \le N_1 - 1$ and $0 \le j \le N_2 - 1$ is referred to as $P(i, j)$. Every processor has four ports denoted by N, S, E, and W, respectively. In each processor, ports can be dynamically connected in pairs to fit computational needs. Our RM only allows two connections to be set in each processor. Furthermore, these two connections must involve disjoint pairs of ports. As shown in Fig. 2, we are given a binary sequence $s_0s_1s_2s_3\ s_4 = 10101$, where s_j is stored in $P(0, j)$ for $0 \le j \le 4$. If s_j is 0 then all processors $P(i, j)$, $0 \le i \le N_1 - 1$, connect their W and E ports; if s_j is 1 then all processors $P(i, j)$, $0 \le i \le N_1 - 1$, connect their W and S ports and their N and E ports. Now processor $P(3, 0)$ broadcasts a start signal, #, from its W port. It is easy to track down in a staircase manner. At this moment, $P(0, 4)$ saves the number of 1's in the sequence, say 3 $(= N_1 - 1 - i = 4 - 1 - 0)$, where i denotes the row number of the processor that receives # from its E port in the last column.

We assume that each processor has a constant number of registers and a set of some basic instructions. A processor can perform a standard arithmetic or boolean operation in unit time. We assume a SIMD model: in each time unit the same instruction is broadcasted to all processors, which execute it and wait for the next instruction. Each instruction can consist of setting local connections, performing an arithmetic or boolean operation, broadcasting a value on a bus, or receiving a value from a specific bus. At any given time, only one processor can broadcast a value onto a bus and processors, if instructed to do so, read the bus. It is assumed that communication along buses takes $O(1)$ time and the RM operates in a SIMD manner.

3. HOUGH TRANSFORM ON RECONFIGURABLE MESHES

Given a binary image $I(i, j)$ for $0 \le i, j \le N - 1$, initially the image has been mapped onto an $N \times N$ RM such that $P(i, j)$ contains $I(i, j)$. For a specific projection angle θ $(-\pi/2 < \theta \le \pi/2)$, we first partition the RM into some parallel bands that are 1 pixel-width wide. A simple argument reveals that at most $N \sin \theta + N \cos \theta + 1$ bands intersect the $N \times N$ RM; thus there will be at most $\lfloor \sqrt{2}N \rfloor + 1$ bands [12]. For each band i, we attach its last processor a variable b_i that will count the number of 1-pixels contained in that band.

The normal length of the band is given by (1). For any θ_j, $0 \le j < p$, all image points that have the same normal length r lie on the same band. Note that the band with 1 pixel-width wide is uniquely defined by the angle θ_j and the length of the normal. In other words, given θ_j, processor $P(i, j)$ can be classified according to its normal length.

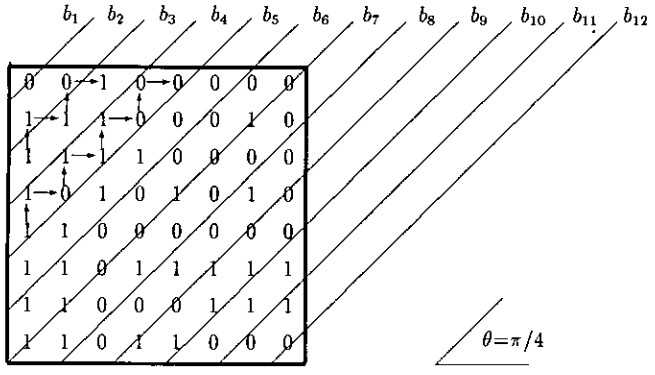


FIG. 3. A banded image for $\theta = \pi/4$.

As shown in Fig. 3, when $\theta = \pi/4$, the 1st band contains no processor, the 2nd band contains $P(0, 0)$, $P(1, 0)$, and $P(0, 1)$, . . . , the 11th band contains $P(7, 6)$, $P(7, 7)$, and $P(6, 7)$, and the 12th band contains no processor. For a specific projection angle θ_j, by (1) each processor can compute its normal length in $O(1)$ time in parallel. At this time, all processors belonging to the same band do have the same normal length.

The following theorem can confirm for us that each band can establish its bus in $O(1)$ time simultaneously.

THEOREM 1. *In the same band, each processor of the RM can visit its nearest neighbor within two steps.*

Proof. We prove it by contradiction. Give an angle θ $(-\pi/2 < \theta \leq \pi/2)$, one example is shown in Fig. 4. Processor A can visit processor B, processor C, or processor D within two steps. Suppose that in the band of Fig. 4, processor E is the nearest right neighbor of processor A, then it needs three steps to visit from A to E. We have that the band contains both processor A and processor E with bandwidth w $(< h < \overline{CD} = 1)$. Hence, it is a contradiction because each bandwidth is of 1 pixel-width. ■

By Theorem 1, we have the following result.

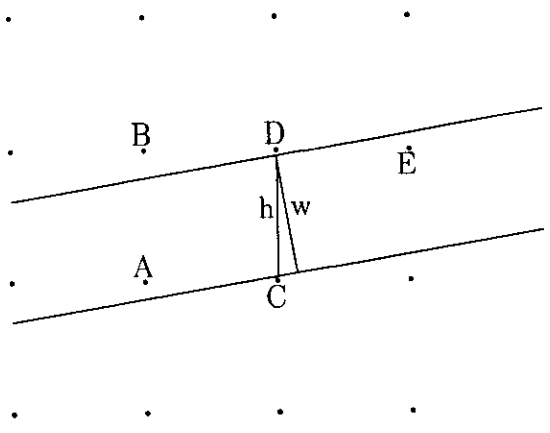


FIG. 4. One example.

COROLLARY 2. *In the same band, each processor of the RM can visit its nearest neighbor in $O(1)$ time.*

In addition, there is a notable property in Fig. 4 that for some θ, in the same band, there are at most two processors in the same column [4].

Algorithm 1 presents our first parallel HT algorithm to compute the number of 1-pixels contained in each band i $(1 \leq i \leq \lfloor\sqrt{2}N\rfloor + 1)$ in $O(\log N)$ time on an $N \times N$ RM. For simplicity, N is assumed to be a power of 2. In fact, it is not hard to extend this algorithm to the general case: N is not a power of 2, and we leave it to the readers. For simplicity, throughout the following three HT algorithms, we only present them for one band.

ALGORITHM 1. A parallel HT algorithm on the $N \times N$ RM.

Step 1. Establish a bus system for each band. First, each processor $P(i, j)$ computes its normal length by (1). Because there are at most two pixels within the same column, each processor then sums up the pixel value of itself and the pixel value stored in its upper-adjacent processor if both processors have the same r value. In order to avoid the collision when sharing processors, we duplicate the ports in each processor. One is used by the current bus while the duplicated port is used by the adjacent bus.

Step 2. For each band i, we want to compute $b_i = \sum_{j=0}^{N-1} p_{i,j}$, where $p_{i,j}$ denotes the pixel value stored in the processor in band i, $1 \leq i \leq \lfloor\sqrt{2}N\rfloor + 1$, and column j, $0 \leq j \leq N - 1$. We first sume up $p_{i,j}$'s in pairs as shown below:

$$p_{i,1} \leftarrow p_{i,0} + p_{i,1}$$
$$p_{i,3} \leftarrow p_{i,2} + p_{i,3}$$
$$\vdots$$
$$p_{i,N-1} \leftarrow p_{i,N-2} + p_{i,N-1}.$$

Then we want to compute $b_i = p_{i,1} + p_{i,3} + \ldots + p_{i,N-1}$. Repeat the above pairwise summations iteratively by using the bus systems dynamically, the value of each b_i can be determined in $O(\log N)$ stages in parallel and is stored in the last processor of band i.

Step 3. We compare all b_i's with the threshold value δ. If $b_i \geq \delta$, then we detect the corresponding lines. This comparison step takes $O(1)$ time.

For example, in Fig. 3, the bus system in the sixth band was constituted by $P(7, 0)$, $P(6, 0)$, $P(6, 1)$, $P(5, 1)$, $P(5, 2)$, $P(4, 2)$, $P(4, 3)$, $P(3, 3)$, $P(3, 4)$, $P(2, 4)$, $P(2, 5)$, $P(1, 5)$, $P(1, 6)$, $P(0, 6)$, and $P(0, 7)$. We have $p_{6,0} = 2$, $p_{6,1} = 2$, $p_{6,2} = 0$, $p_{6,3} = 0$, $p_{6,4} = 1$, $p_{6,5} = 0$, $p_{6,6} = 1$, and $p_{6,7} = 0$. Then we set those upper-adjacent processors to be bridges. The role of bridges is only transmitting the

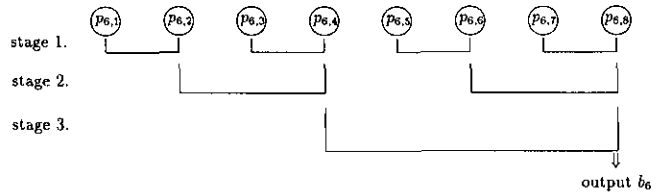


FIG. 5. A computational diagram for computing b_6.

data. Those bridges are $P(6, 0)$, $P(5, 1)$, $P(4, 2)$, $P(3, 3)$, $P(2, 4)$, $P(1, 5)$, and $P(0, 6)$. The bus established in the sixth band and the bus established in the seventh band share processors $P(6, 0)$, $P(5, 1)$, $P(4, 2)$, $P(3, 3)$, $P(2, 4)$, $P(1, 5)$, and $P(0, 6)$. By Corollary 2, Step 1 takes $O(1)$ time to establish a bus system for each band. Based on previous duplication strategy, in Step 2, b_i's can be performed in $O(\log N)$ time in a collision-free way. Figure 5 illustrates the computational diagram for computing b_6 of Fig. 3. It needs three stages to obtain b_6.

Combining the overall time spent in Steps 1–3, we have the following result.

THEOREM 3. *For p projection angles, the HT can be computed in $O((p/k)\log N)$ time on an RM with $O(kN^2)$ processors.*

On a 3-D RM, we next present a constant time HT algorithm. We construct a 3-D RM with a $N \times N$ base mesh denoted by layer L_0, and in total there are N layers, say L_0, L_1, . . . , and L_{N-1}. Initially, the image is mapped into L_0.

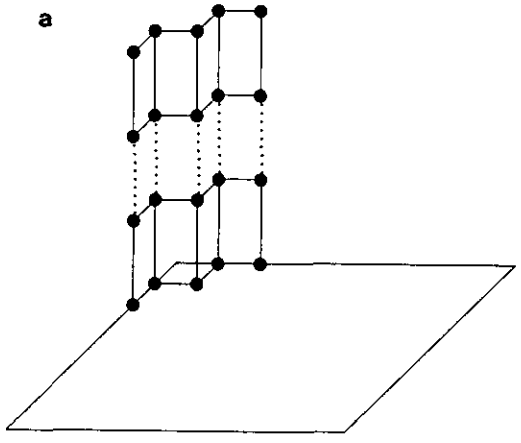
ALGORITHM 2. A parallel HT algorithm on the $N \times N \times N$ RM.

Step 1. Establish a vertical 2-D RM for each band. On L_0, by Corollary 2 we first establish the bus system for each band in $O(1)$ time; this part is similar to Step 1 of Algorithm 1. At this moment, each processor $P(0, i, j)$, $0 \le i, j \le N - 1$, on L_0 recognizes its adjacent neighbors and then acknowledges the processors $P(k, i, j)$ for $1 \le k \le N - 1$ on L_1, . . . , L_{N-2}, and L_{N-1}, to form a vertical 2-D RM. For an image, it needs $O(1)$ time to construct these $\lfloor\sqrt{2}N\rfloor + 1$ vertical 2-D RMs in parallel.

Step 2. Sum up the number of 1-pixels for each band by using the vertical 2-D RM. Thus, all b_i's are obtained.

Step 3. Finally, all b_i's values are stored in the first row and last column of processors on L_0. We compare them with the threshold value δ and determine whether they form the corresponding lines.

For example, in the third band of L_0, processors $P(0, 2, 0)$, $P(0, 1, 0)$, $P(0, 1, 1)$, $P(0, 0, 1)$, and $P(0, 0, 2)$ construct a vertical 2-D RM as shown in Fig. 6a, where processors $P(0, 1, 0)$ and $P(0, 0, 1)$ serve as the bridges and contribute 0-pixels since their coordinates are out of the third band. After straightening these zig-zag lines, the alternate vertical 2-D RM is shown in Fig. 6b. On one vertical 2-D RM in Fig. 6b, if $P(0, i, j)$ contains 0-pixels then all processors in the same column connect their W and E ports; if $P(0, i, j)$ contains 1-pixels then all processors in the same column connect their W and S ports and their N and E ports. Now processor $P(7, 2, 0)$ broadcasts a start signal, #, from its W port; it can be verified that the number $(N - 1) - l$, where l is the layer number of the



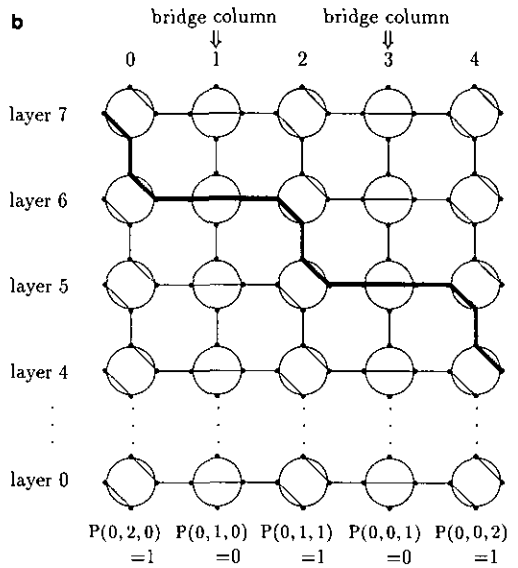


FIG. 6. Two vertical 2-D RMs for the third band of L_0. (a) Original vertical 2-D RM for the third band. (b) Vertical 2-D RM after straightening.

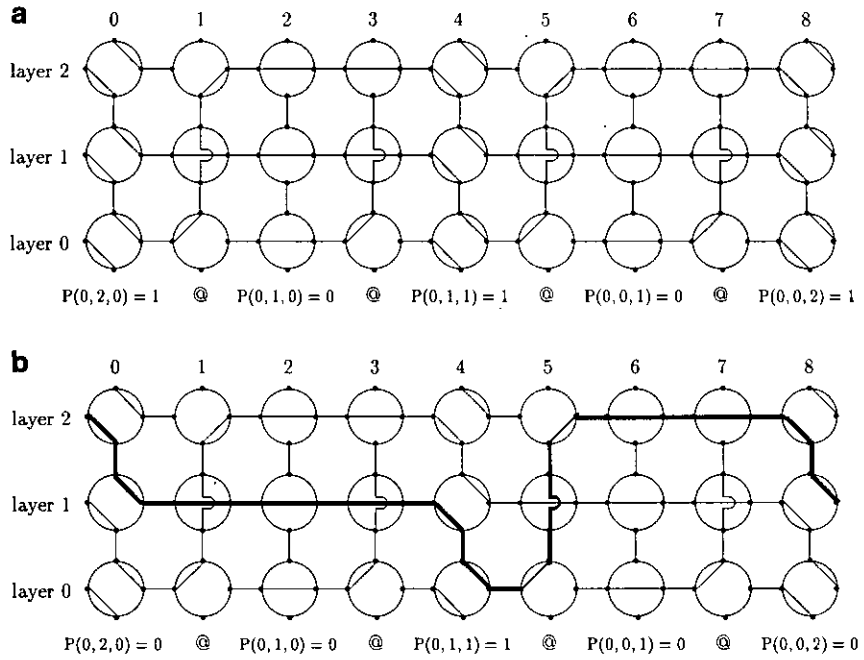


FIG. 7. (a) The augmented 2-D RM after setting up port-connections. (b) The augmented 2-D RM after pushing the start signal.

unique processor that receives # from its E port in the last column, equals the number of 1-pixels in the band. Based on the previous duplication strategy to avoid the collision when sharing processors, the work of summing up the number of 1-pixels in each band can be performed in $O(1)$ time in a collision-free way. As a result, all b_i's are obtained.

Since each above step needs $O(1)$ time, we have the following result.

THEOREM 4. *For p projection angles, the HT can be computed in $O(p/k)$ time on a 3-D RM with $O(kN^3)$ processors.*

Consider the case: the number of layers, say, **M**, in the 3-D RM is less than or equal to N. Following Algorithm 2 and the concept of *modulo* [22], in what follows, we present a more general result: the HT can be computed in $O((p \log N)/(k \log M))$ time on a RM of $O(kN^2M)$ processors, $2 \leq M \leq N$. We take the third band in Fig. 3 as an example to demonstrate our third algorithm.

ALGORITHM 3. A parallel HT algorithm on the RM with size $O(N^2M)$.

Step 1. Establish a vertical 2-D RM with height M for each band. Different from Step 1 of Algorithm 2, in the third band of L_0, processors $P(0, 2, 0)$, $P(0, 1, 0)$, $P(0, 1, 1)$, $P(0, 0, 1)$, and $P(0, 0, 2)$ construct an augmented vertical 2-D RM with height 3 as shown in Fig. 7(a). Recall that for any band, on the corresponding bus system, the processor contributes 0-pixels to the sequence if its coordinate is out of the band. Suppose that we have duplicated all the ports as described in Algorithm 2. If $P(0, i, j)$ contains 0-pixels then all processors in the same column connect their W and E ports; if $P(0, i, j)$ contains 1-pixels then all processors in the same column connect their W and S ports and their N and E ports. The augmented sequence of the third band is represented by 1 @ 0 @ 1 @ 0 @ 1, where the symbol @ denotes a dummy signal to set up the processors in the same column in order to transmit the start signal, #, to the adjacent processor in the next column. If processor $P(0, i, j)$ contains the @ signal then all processors $P(k, i, j)$, $1 \leq k \leq M - 2$, where M denotes the number of layers, connect their W and E ports and their N and S ports; $P(M - 1, i, j)$ connects its W and E ports when $P(0, i - 1, j)$ contains 0-pixels, but connects its S and E ports when $P(0, i - 1, j)$ contains 1-pixels; $P(0, i, j)$ connects its W and N ports by itself. Figure 7a illustrates the detailed port-connections of each processor for the third band. For each band, this construction step takes $O(1)$ time.

Step 2. Sum up the number of 1-pixels for each band by using the augmented 2-D RM. If we follow Step 2 of Algorithm 2, then we may not have sufficient height to perform the summation work. We modify Step 2 of Algorithm 2 by using the concept of *modulo* [22]. Now processor $P(2, 2, 0)$ broadcasts a start signal, #, from its W port. The # signal crashes to the bottom processor $P(0, 1, 1)$ and $P(0, 1, 1)$ is assigned the value 1 while the other bottom processors are assigned the values 0. Simulta-

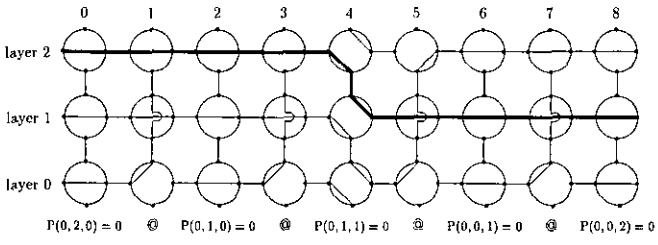


FIG. 8. The augmented 2-D RM after the second iteration.

neously, $P(1, 0, 2)$ receives the #signal from $P(2, 2, 0)$. At this moment, we obtain the first remainder $r_4^1 = 1 (= M - 1 - 1 = 3 - 1 - 1)$, which is stored in $P(1, 0, 2)$, and it satisfies $r_4^1 = f(10101) \bmod (M - 1)$, where $f(10101)$ denotes the number of 1's in the sequence 10101. Next, processor $P(1, 0, 2)$ sends the value of r_4^1 to b_4 which is stored in $P(0, 0, 2)$. Now, the associated sequence stored in the bottom processors is represented by 0 @ 0 @ 1 @ 0 @ 0 @ 0 which is shown in Fig. 7b. Basically, each iteration consists of two phases: setting up port-connections and pushing the start signal, respectively.

Once again, $P(2, 2, 0)$ broadcasts a start signal from its W port after setting up the port-connections based on the sequence 0 @ 0 @ 1 @ 0 @ 0. Then $P(1, 0, 2)$ receives the # signal and we obtain the second remainder $r_4^2 = 1$ which satisfies $r_4^2 = f(00100) \bmod (M - 1)$. Next, $P(0, 0, 2)$ receives r_4^2 from $P(1, 0, 2)$ and performs $b_4 = r_4^2 \times (M - 1) + r_4^1$. Now the sequence stored in the bottom processors is 0 @ 0 @ 0 @ 0 @ 0 which is shown in Fig. 8. Up to here, the summation work is finished.

Generally speaking, the above process can be carried out iteratively, say, t iterations, for any band. Finally, the bottom processors contain the sequence 0 @ 0 @ 0 · · · 0 @ 0 @ 0 and we obtain $b_4 = r_4^t \times (M - 1)^{4-1} + \cdots + r_4^2 \times (M - 1) + r_4^1$. It is not hard to check that $t = O(\log N/\log M)$ and b_4 can be computed in $O(\log N/\log M)$ time.

Step 3. Similar to Step 3 of Algorithm 2.

Since Step 1 and Step 3 need $O(1)$ time and Step 2 needs $O(\log N/\log M)$ time, we have the following result.

THEOREM 5. *For p projection angles, the HT can be computed in $O(p \log N/(k \log M))$ time on an RM with $O(kN^3)$ processors. When setting $k = O(p)$ and $M = O(N^{1/c})$, where c is a constant, the HT can be computed in $O(1)$ time.*

4. REMARKS AND CONCLUSIONS

The well-known HT is an efficient method for detecting predefined features in digital image. In this paper we have presented simple algorithms for this problem which take only $O(\log N)$ time and $O(1)$ time on a 2-D RM of $O(N^2)$ processors and a 3-D RM of $O(N^3)$ processors, respectively, in which each processor has only a constant number of switches and registers and uses only a constant amount of memory. As far as the second algorithm is concerned, although the number of processors used to solve the HT problem is $O(N^3)$, we overcome the restriction in Kao *et al.*'s result which is dominated by the number of edge pixels. Finally, if we set $M = O(\log N/\log \log N)$ for our third algorithm, then we reply to Olariu *et al.*'s conjecture [21] that the $O(\log \log N)$ algorithm for the HT is achievable. In addition, when setting $k = O(p)$ and $M = O(N^{1/c})$, where c is a constant, the HT can be computed in $O(1)$ time. Although our algorithms seem ideally for detecting straight lines corresponding to the parallel bands, how to design efficient Hough transforms on RMs for the other parametric curves [2] such as circles and ellipses is our future research topic.

Finally, some implementation considerations are discussed. The YUPPIE system [17] and the configurable hardware [7] are the first two VLSI implementations to justify the enhancement of the 2-D RM. In fact, the claim of the $O(1)$ time broadcast delay is not true in the above two implementations. By employing precharged circuits, the broadcast delay has been shorten in the gated-connected network [30]. Furthermore, it has been shown that the $O(1)$ time claim is reasonable if the reconfigurable bus system is implemented by using optical fibers [29]. An empirical methodology for exploring RMs is presented in [13]. Especially, Maresca *et al.* [18] present a hierarchical node clustering scheme for packaging a class of RMs which uses circuit-switching-based router at each node to deliver a different topology at every instruction. This scheme makes it possible to obtain communication speed-up and automatic control, at the compiler level, over signal propagation delay.

REFERENCES

1. K. E. Batcher, Design of a massively parallel processor, *IEEE Trans. Comput. Ser. C* **29** 1980, 836–840.
2. D. H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognit.* **13**(2) 1981, 111–122.
3. D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice–Hall, Englewood Cliffs, NJ, 1982.
4. R. E. Cypher, J. L. C. Sanz, and L. Snyder, The Hough transform has $O(N)$ Complexity on $N \times N$ Mesh Connected Computers, *SIAM J. Comput.* **19**(5), 1990, 805–820.
5. M. J. B. Duff, Review of the CLIP image processing system, in National Computer Conference, *Anaheim, CA*, 1978.
6. R. O. Duda and P. E. Hart, Use of the Hough transformation to detect lines and curves in picture, *Commun. ACM* **15**(1), 1972, 11–15.
7. J. P. Gray and T. A. Kean, Configurable hardware: a new paradigm for computation, in *Proceedings, 10th Caltech. Conference on VLSI*, 1989, pp. 279–295.
8. C. Guerra and S. Hambruch, Parallel algorithms for line detection on a mesh, *J. Parallel Distrib. Comput.* **6**, 1989, 1–19.

9. C. S. Kannan and Y. H. Chuang, Fast Hough transform on a mesh connected processor array, *Inform. Process. Lett.* **33,** 1990, 243–248.
10. NCR Microelectronics Division, Product Description ncr45cg72, NCR Corporation, Dayton, OH, 1984.
11. T. W. Kao, S. J. Horng, Y. L. Wang, and K. L. Chung, A constant time algorithm for computing Hough transform, *Pattern Recognit.* **26**(2), 1993, 277–286.
12. F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes, Kaufmann, Los Altos, CA, 1992.*
13. W. B. Ligon III and U. Ramachandran, An empirical methodology for exploring reconfigurable architectures, *J. Parallel Distrib. Comput.* **19,** 1993, 323–337.
14. H. W. Li and Q. F. Stout, Reconfigurable SIMD massively parallel computers, *Proc. IEEE* **79**(4), 1991, 429–443.
15. M. Maresca, M. Lavin, and H. W. Li, Parallel architectures for vision, *Proc. IEEE* **76,** Aug. 1988, 970–981.
16. M. Maresca, H. W. Li, and M. Shen, Parallel computer vision on polymorphic-torus architecture, *Int. J. Comput. Vision Appl.*, Nov. 1989.
17. M. Maresca and H. W. Li, Connection autonomy and SIMD computers: A VLSI implementation, *J. Parallel Distrib. Comput.* **7,** 1989, 302–320.
18. M. Maresca, H. W. Li, and P. Baglietto, Hardware support for fast reconfigurability in processor arrays, Meshes with reconfigurable buses, in Proceedings of the International Conference on Parallel Processing, *1993,* Vol. 1, pp. 282–289.
19. R. Miller, V. K. Prasanna Kumar, D. Reisis, and Q. F. Stout, Meshes with reconfigurable buses, in Proceedings of the International Conference on Parallel Processing, *1988,* Vol. 1. 205–208.
20. R. Miller, V. K. Prasanna Kumar, D. Reisis, and Q. F. Stout, Data movement operations and applications on reconfigurable VLSI arrays, in *Proceedings of the Fifth MIT Conference on Advanced Research in VLSI, 1988,* pp. 163–178.
21. S. Olariu, J. L. Schwing, and J. Zhang, Fast computer vision algorithms on reconfigurable meshes, *Image Vision Comput. J.* **10,** 1992, 610–616.
22. S. Olariu, J. L. Schwing, and J. Zhang, Fundamental data movement of reconfigurable meshes, in *Proceedings*, International Phoenix Conference on Computers and Communication, Scottsdle, AZ, 1992, pp. 472–478.
23. A. Rosenfeld, *Picture Processing by Computer,* Academic Press, New York, 1969.
24. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, 2nd ed., Academic Press, San Diego, 1986.
25. A. Rosenfeld, J. Ornelas, Jr., and Y. Hung, Hough transform algorithms for mesh-connected SIMD parallel processors, *Comput. Vision Graphics Image Process.* **41,** 1988, 293–305.
26. J. Rothstein, On the ultimate limitations of parallel processing, in *Proceedings, International Conference on Parallel Processing., St. Charles, MO, 1976,* pp. 206–212.
27. J. Rothstein and A. Davis, Parallel recognition of parabolic and conic patterns by bus automata, in *Proceedings, International Conference on Parallel Processing, St. Charles, MO, 1979,* pp. 288–297.
28. J. Rothstein, Bus automata, brains, and mental models, *IEEE Trans. Syst. Man Cybernetics* **18,** 1988, 522–531.
29. A. Schuster and Y. Ben-Asher, Algorithms and optic implementation for reconfigurable networks, in *Proceedings 5th Jerusalem Conference on Information Technology, 1990.*
30. D. B. Shu, L. W. Chow, and J. G. Nash, A content-addressable, bit-serial associative processor, in *Proceedings IEEE Workshop on VLSI Signal Processing, CA, 1988.*
31. H. J. Siegel, J. B. Armstrong, and D. W. Watson, Mapping computer-vision-related tasks onto reconfigurable parallel processing systems, *IEEE Computer.* **25**(2), 1992, 54–62.
32. Q. F. Stout, Meshes with multiple buses in *Proceedings 27th IEEE Symposium on the Foundations of Computer Science, 1986,* pp. 264–273.
33. S. L. Tanimoto, A pyramidal approach to parallel processing, in *Proceedings Internatonal Symposium on Computer Architecture, 1983,* pp. 372–378.
34. P. Thangavel and V. P. Muthuswamy, Parallel algorithms for addition and multiplication on processor arrays with reconfigurable bus systems, *Inform. Process. Lett.* **46,** 1993, 89–94.
35. B. F. Wang, G. H. Chen, and F. C. Lin, Constant time sorting on a processing array with a reconfigurable bus system, *Inform. Process. Lett.* **34**(4), 1990, 187–192.
36. B. F. Wang and G. H. Chen, Constant time algorithms for the transitive closure and some related graph problems on processor arrays with a reconfigurable bus system, *IEEE Trans. Parallel Distrib. Syst.* **1,** 1990, 500–507.
37. C. C. Weems, S. P. Levitan, A. R. Hanson, E. M. Riseman, D. B. Shu, and J. G. Nash, The image understanding architecture, *Int. J. Comput. Vision* **2,** 1989, 251–282.